FREE EBOOK

oliveboard

Tamil Nadu Textbook

Class 12 - English Medium

Computer Science

For all TN Exams

GOVERNMENT OF TAMILNADU

HIGHER SECONDARY
SECOND YEAR

COMPUTER SCIENCE

A publication under Free Textbook Programme of Government of Tamil Nadu

Department of School Education

Untouchability is Inhuman and a Crime

XII Std - CS EM Introduction Pages.indd 1 1/24/2020 10:14:29 AM

Government of Tamil Nadu

First Edition - 2019

Revised Edition - 2020

(Published under New syllabus)

NOT FOR SALE

Content Creation

State Council of Educational Research and Training
© SCERT 2019

Printing & Publishing

Tamil NaduTextbook and Educational Services Corporation

www.textbooksonline.tn.nic.in

XII Std - CS EM Introduction Pages.indd 2 1/24/2020 10:14:30 AM

PREFACE

The tremendous effect of the computer and computing technology is in shaping the modern society for the betterment of mankind. Human civilization achieved the highest peak with the development of computer known as "**Internet Era**".

Python being a high level language is good for beginners to learn due to its easy syntax and powerful memory management. For any internet applications, it is good to learn as python being the better choice.

The user of this textbook being acquainted with the knowledge of C++ in std XI will have no difficulty in studying python language. No substantial knowledge nor experience, is required as the examples illustrated in the book are easy to follow

HOW TO USE THE BOOK?

- This book does not require prior knowledge in computer Technology
- Each unit comprises of simple activities and demonstrations which can be done by the teacher and also students.
- Technical terminologies are listed in glossary for easy understanding
- The " Do you know?" boxes enrich the knowledge of reader with additional information
- Workshops are introduced to solve the exercises using software applications
- QR codes are used to link supporting additional
- materials in digital form

How to get connected to QR Code?

- Download the QR code scanner from the google play store/ apple app store into your smartphone
- Open the QR code scanner application
- Once the scanner button in the application is clicked, camera opens and then bring it closer to the QR code in the textbook.
- Once the camera detects the QR code, a URL appears in the screen. Click the URL and go to the content page.

XII Std - CS EM Introduction Pages.indd 3 1/24/2020 10:14:30 AM

CAREER GUIDANCE AFTER 12TH

COMPUTER SCIENCE

Computer Knowledge Hub

- Centre for Development of Advanced Computing https://www.cdac.in/
- Indian Computing Olympiad https://www.iarcs.org.in/inoi
- International Olympiad of Informatics https://ioinformatics.org
- Microsoft certification exams https://www.microsoft.com/en-in/learning/exam-list.aspx
- National Cyber Olympiad http://www.sofworld.org/nco
- National Institute of Electronics & Information Technology http://www.nielit.gov.in/
- National Programme on Technology Enhanced Learning https://nptel.ac.in/

After Completing +2

Arts & Science BSc Courses | Animation & Multimedia
Computer Science | Geography
Journalism / Mass Media Communication | Library Science
Maths / Physics / Chemistry / Statistics | Psychology/ Sociology
Social Work | Visual Communication | B.C.A.

Technical Diploma

Diploma in Engineering

Professional Degree & Entrance Exams

- Hotel & Catering Institute http://www.ihmchennai.org/ihmchennai/
- AIEEE- All India Engineering Entrance Exam
- B.E/B.Tech/ B.Arch (JEE, AIEEE in IITs and NITs)
- Fashion Technology & Design http://nift.ac.in
- GATE-Graduate Aptitude Test in Engineering www gate.iitm.ac.in
- Indian Navy – 10+2 BTech Entry Scheme
- Institute of Chartered Accountants of India https://icai.org
- Institute of Company Secretary https://www.icsi.edu
- Institute of Cost Accountants of India http://icmai.in
- Institute of Banking Personal Selection IBPS https://ibps.in
- JEE-Joint Entrance Examination https://jeemain.nic.in/
- JEST- Joint Entrance Screening Test https://www.jest.org.in
- Law 5-Year Integrated Course http://tndalu.ac.in
- National Defence Academy https://www.nda.nic.in
- NET- National Eligibility Test (CSIR and UGC) http://cbsenet.nic.in
- TamilNadu Dr. Ambedkar Law University http://tndalu.ac.in
- Technical Entry Scheme – Army/Navy/Airforce
- TIFR GS - Tata Institute of Fundamental Research Graduate School http://univ.tifr.res.in
- Visual Arts Degree http://www.tndipr.gov.in/television-traininginstitute.aspx

PG in Computer Science

Artificial Intelligence
Information Management and Data Analytics
Algorithms
Applied Computer Science
Big Data Analytics
Bio Informatics
Cloud Computing
Compiler Design & Construction
Computer and Network Security
Computer Networks
Computer Security
Computer Simulation
Cyber Security
Data Analytics
Data Mining
Database Management Systems
Fundamentals of Algorithms
Graphics
Machine Learning
Mobile and Web Computing
Mobile Device Programming
Modern Programming Practices
Modern Web Applications
Operating Systems
Parallel Programming
Robotics
Software Engineering
Software Engineering
Software Testing
Systems Analysis and Design
Web Application Architecture
Web Application Programming

Scholarships for graduate and post graduate courses

- DST – INSPIRE Fellowships (for Ph.D)
- DST – INSPIRE Scholarships (for UG and PG)
- In addition various fellowships for SC/ST/PWD,
- Indira Gandhi Fellowship for Single girl child (for UG and PG)
- Moulana Azad Fellowship for minorities (for Ph.D)
- UGC National Fellowship (for Ph.D)
- International Olympiad: for getting stipend for Higher Education in Science and Mathematics
- OBC etc are available.
- Visit website of University Grants Commission

XII Std - CS EM Introduction Pages.indd 4 1/24/2020 10:14:30 AM

CAREER GUIDANCE

After PG courses

MPhil –Computer Science | PhD – Computer Science

Following are the latest topics for PhD in computer science:

BioInformatics and Computational Biology

Capturing and Visualizing Persona Through Faces

Comprehensive analysis of RNA sequencing experiments

Designing and Evaluating Information Gathering Robots

Digital Pathology: Diagnostic Errors, Viewing Behavior and Image Characteristics

Embedded Systems | Game Theory | Graph Theory

Graphics and Visualization | Human Computer Interaction – HCI

Improving Fault Tolerance and Performance of Data Center Networks

Increasing Access to Computer Science for Blind Students

In-situ Semantic 3D Modeling

Intelligent Crowdsourcing for Natural Language Learning and Other AI Applications

Learning Robust Tractable Models for Vision

Manipulators And Manipulation In High Dimensional Spaces

New Algorithmic Tools for Distributed Similarity Search

Reproducible measurements of web security and privacy

The Security and Privacy of Web and Mobile Advertising

Towards More Practical Reinforcement Learning, with Applications to Educational Games

Topics for Research after PG in Computer Studies

Architectures, Compiler Optimization, and Embedded Systems.

Bioinformatics and Computational Biology | Cloud Computing

Data Mining, Databases, and Geographical Information Systems.

Graphics and Visualization | High Performance Computing.

Human Computer Interaction | Internet of Things

Natural Language Processing | Networks, Distributed Systems, and Security.

Competitive Exams for Govt. Jobs

Airforce Common Admission Test – AFCAT | Army Education Officer Entry- AEC

Combined Defence Services –CDS | Defence Service Staff College, Nilgiris

Indian Defence Services | Indian Military Academy

Judge Advocate General Department-JAG | NCC Entry | Railway Board Examination

SSC NAVY (Pilot/Observer) | SSC Tech Entry – Officers Training School

Staff Selection Examination | Tamil Nadu Public Service Commission

Teacher Recruitment Board | Technical Graduate Course –TGC

Territorial Army | Union Public Service Examination | Women Special Entry Scheme

COMPUTER SCIENCE

Computer Related Jobs

Applications Software Developer | Big Data Analysts

Cloud Computing Programmer | College/ University Faculty

Computer Programmer | Computer Teacher

Computer Vocational Instructor | Computer Information Research Scientist

Computer Information Systems Manager | Computer Network Architect

Computer Support Specialist | Computer System Analysts

Data Mining Specialist | Database Administrator

Information Security Analysts | Market Research Analysts

Network & Computer System Administrator | Research Assistant

Systems Software Developer | User Interface Designer | Web Developer

Research Institutions in various areas of science

- Bhaba Atomic Research centre (BARC) Mumbai www.barc.gov.in
- BITS Pilani, https://www.bits-pilani.ac.in
- Central Universities www.ugc.ac.in
- Chennai Mathematical Institute (CMI) Chennai www.cmi.ac.in
- Delhi University, Delhi www.du.ac.in
- Hyderabad central university, Hyderabad www.uohyd.ac.in
- IISER Educational Institutions www.iiseradmission.in
- Indian Institute of Technology in various places (IIT's) www.iitm.ac.in
- Institute of Mathematical Sciences (IMSc) Chennai www.imsc.res.in
- Jawaharlal Nehru University (JNU) www.jnu.ac.in
- Mumbai University, Mumbai www.mu.ac.in
- National Institute of Science Education and Research (NISER), www.niser.ac.in
- National Institute of Technology (NITs) www.nitt.edu
- SavithiribaiPhule Pune university, Pune www.unipune.ac.in
- State Universities https://www.ugc.ac.in
- Tata Institute of Fundamental Research (TIFR) Mumbai www.tifr.res.in

XII Std - CS EM Introduction Pages.indd 5 1/24/2020 10:14:30 AM

Table of Contents

Computer Science-II Year				
UNIT NO.	CHAPTER	COMPUTER SCIENCE	PAGE NO	MONTH
UNIT- I Problem Solving Techniques	1	Function	1	June
	2	Data Abstraction	11	June
	3	Scoping	21	June
	4	Algorithmic Strategies	31	June
UNIT- II Core Python	5	Python -Variables and Operators	47	July
	6	Control Structures	67	July
	7	Python functions	89	July
	8	Strings and String manipulation	114	Aug
UNIT-III Modularity and OOPS	9	Lists, Tuples, Sets and Dictionary	132	Aug
	10	Python Classes and objects	170	Aug
UNIT-IV Database concepts and MySql	11	Database Concepts	189	Oct
	12	Structured Query Language (SQL)	205	Oct
	13	Python and CSV files	232	Oct
UNIT-V Integrating Python with MySql and C++	14	Importing C++ programs in Python.	267	Oct
	15	Data manipulation through SQL	291	Nov
	16	Data visualization using pyplot: line chart, pie chart and bar chart	320	Nov
		Glossary	335	
Annexure-1		List of Python functions	341	
Annexure-2		Installation of MySQL and Creating a Database	345	
Annexure-3		Installation of MinGW	346	
Annexure-4		Installation of pip	351	

2ZQ6DE

E - book

3F91EQ

Assessment

B274_12_CS_EM

DIGI links

XII Std - CS EM Introduction Pages.indd 6 1/24/2020 10:14:30 AM

Unit I

CHAPTER 1

FUNCTION

4A1UFQ

Learning Objectives

After the completion of this chapter, the student will be able to:

- Understand Function Specification.
- Parameters (and arguments).
- Interface Vs Implementation.
- Pure functions.
- Side - effects (impure functions).

1.1 Introduction

The most important criteria in writing and evaluating the algorithm is the time it takes to complete a task. To have a meaningful comparison of algorithms, the duration of computation time must be independent of the programming language, compiler, and computer used. As you aware that algorithms are expressed using statements of a programming language. If a bulk of statements to be repeated for many numbers of times then subroutines are used to finish the task.

Subroutines are the basic building blocks of computer programs. Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly. In Programming languages these subroutines are called as Functions.

1.2 Function with respect to Programming language

A function is a unit of code that is often defined within a greater code structure. Specifically, a function contains a set of code that works on many kinds of inputs, like variants, expressions and produces a concrete output.

1.2.1 Function Specification

Let us consider the example *a:= (24)*. *a:= (24)* has an expression in it but (24) is not itself an expression. Rather, it is a function definition. Definitions bind values to names, in this case the value 24 being bound to the name *'a'*. Definitions are not expressions, at the same time expressions are also not treated as definitions. Definitions are distinct syntactic blocks. Definitions can have expressions nested inside them, and vice-versa.

1.2.2 Parameters (and arguments)

Parameters are the variables in a function definition and arguments are the values which are passed to a function definition.

1. Parameter without Type

Let us see an example of a function definition:

XII Std - CS EM Chapter-1.indd 1 24-01-2020 12:38:55

```
(requires: b>=0 )
(returns: a to the power of b)
let rec pow a b:=
        if b=0 then 1
        else a * pow a (b-1)
```

In the above function definition variable *'b'* is the parameter and the value which is passed to the variable *'b'* is the argument. The precondition ***(requires)*** and postcondition ***(returns)*** of the function is given. Note we have not mentioned any types: ***(data types)***. Some language compiler solves this type ***(data type)*** inference problem algorithmically, but some require the type to be mentioned.

In the above function definition if expression can return ***1*** in the then branch, shows that as per the ***typing*** rule the entire if expression has type ***int***. Since the if expression is of type ***'int'***, the function's return type also be *'int'*. ***'b'*** is compared to *0* with the equality operator, so ***'b'*** is also a type of *'int'*. Since ***'a'*** is multiplied with another expression using the * operator, ***'a'*** must be an int.

2. Parameter with Type

Now let us write the same function definition with types for some reason:

```
(requires: b>  0 )
(returns: a to the power of b )
let rec pow (a: int) (b: int) : int :=
        if b=0 then 1
        else a * pow a (b-1)
```

When we write the type annotations for ***'a'*** and ***'b'*** the parentheses are mandatory. Generally we can leave out these annotations, because it's simpler to let the compiler infer them. There are times we may want to explicitly write down types. This is useful on times when you get a type error from the compiler that doesn't make sense. Explicitly annotating the types can help with debugging such an error message.

The syntax to define functions is close to the mathematical usage: the definition is introduced by the keyword ***let***, followed by the ***name*** of the function and its *arguments*; then the formula that computes the image of the argument is written after an = sign. If you want to define a recursive function: use ***"let rec" instead of "let".***

Syntax: The syntax for function definitions:

```
let rec fn a1 a2 ... an := k
```

Here the ***'fn'*** is a variable indicating an identifier being used as a function name. The names ***'a1'*** to ***'an'*** are variables indicating the identifiers used as parameters. The keyword ***'rec'*** is required if ***'fn'*** is to be a recursive function; otherwise it may be omitted.

Note

A function definition which call itself is called recursive function.

For example: let us see an example to check whether the entered number is even or odd.

XII Std - CS EM Chapter-1.indd 2 24-01-2020 12:38:55

```
(requires: x>= 0)
        let rec even x :=
        x=0 || odd (x-1)
        return 'even'
(requires: x>= 0)
        let odd x :=
        x<>0 && even (x-1)
        return 'odd'
```

The syntax for function types:

$x \rightarrow y$

$x1 \rightarrow x2 \rightarrow y$

$x1 \rightarrow \ldots \rightarrow xn \rightarrow y$

The ***'x'*** and ***'y'*** are variables indicating types. ***The type x → y is the type of a function that gets an input of type 'x' and returns an output of type 'y'.*** Whereas ***x1 → x2 → y*** is a type of a function that takes two inputs, the first input is of type ***'x1'*** and the second input of type ***'x2'***, and returns an output of type ***'y'***. Likewise ***x1 → … → xn → y*** has type ***'x'*** as input of n arguments and ***'y'*** type as output.

Note

All functions are static definitions. There is no dynamic function definitions.

1.3 Interface Vs Implementation

An interface is a set of action that an object can do. For example when you press a light switch, the light goes on, you may not have cared how it splashed the light. In Object Oriented Programming language, an Interface is a description of all functions that a class must have in order to be a new interface. In our example, anything that ***"ACTS LIKE"*** a light, should have function definitions like turn_on () and a turn_off (). The purpose of interface is to allow the computer to enforce the properties of the class of ***TYPE T*** *(whatever the interface is)* must have functions called *X, Y, Z,* etc.

A class declaration combines the external interface *(its local state)* with an implementation of that interface *(the code that carries out the behaviour)*. An object is an instance created from the class.

The interface defines an object's visibility to the outside world.

The difference between interface and implementation is

Interface	Implementation
Interface just defines what an object can do, but won't actually do it	Implementation carries out the instructions defined in the interface

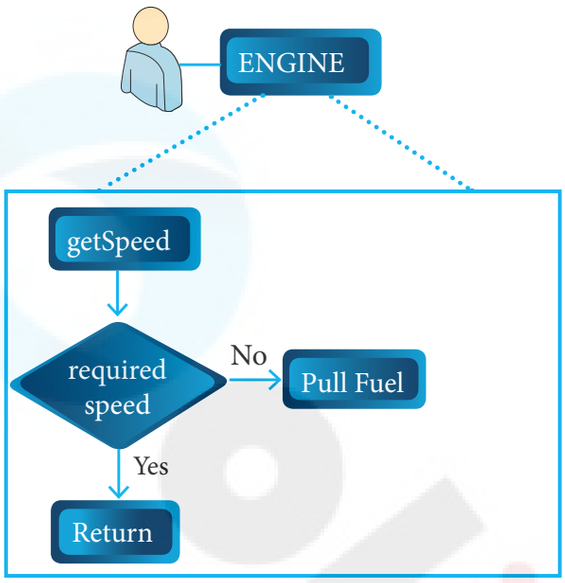
In object oriented programs classes are the interface and how the object is processed and executed is the implementation.

1.3.1 Characteristics of interface

- The class template specifies the interfaces to enable an object to be created and operated properly.
- An object's attributes and behaviour is controlled by sending functions to the object.

For example, let's take the example of increasing a car's speed.

XII Std - CS EM Chapter-1.indd 3 24-01-2020 12:38:55



The person who drives the car doesn't care about the internal working. To increase the speed of the car he just presses the accelerator to get the desired behaviour. Here the accelerator is the interface between the driver *(the calling / invoking object)* and the engine *(the called object)*.

In this case, the function call would be Speed (70): This is the interface.

Internally, the engine of the car is doing all the things. It's where fuel, air, pressure, and electricity come together to create the power to move the vehicle. All of these actions are separated from the driver, who just wants to go faster. Thus we separate interface from implementation.

Let us see a simple example, consider the following implementation of a function that finds the minimum of its three arguments:

```
let min 3 x y z :=
        if x < y then
                if x < z then x else z
        else
                if y < z then y else z
```

1.4 Pure functions

Pure functions are functions which will give exact result when the same arguments are passed. For example the mathematical function sin (0) always results **0**. This means that every time you call the function with the same arguments, you will always get the same result. A function can be a pure function provided it should not have any external variable which will alter the behaviour of that variable.

Let us see an example

```
let square x
        return: x * x
```

The above function square is a pure function because it will not give different results for same input.

There are various theoretical advantages of having pure functions. One advantage is that if a function is pure, then if it is called several times with the same arguments, the compiler only needs to actually call the function once. Lt's see an example

```
let i: = 0;
        if i <strlen (s) then
        -- Do something which doesn't affect s
        ++i
```


XII Std - CS EM Chapter-1.indd 4 24-01-2020 12:38:55

If it is compiled, ***strlen (s)*** is called each time and strlen needs to iterate over the whole of ***'s'***. If the compiler is smart enough to work out that strlen is a pure function and that ***'s'*** is not updated in the loop, then it can remove the redundant extra calls to strlen and make the loop to execute only one time. From these what we can understand, strlen is a pure function because the function takes one variable as a parameter, and accesses it to find its length. This function reads external memory but does not change it, and the value returned derives from the external memory accessed.

Note

Evaluation of pure functions does not cause any side effects to its output

1.4.1 Impure functions

The variables used inside the function may cause side effects though the functions which are not passed with any arguments. In such cases the function is called impure function. When a function depends on variables or functions outside of its definition block, you can never be sure that the function will behave the same every time it's called. For example the mathematical function random() will give different outputs for the same function call.

```
let Random number
let a := random()
    if a > 10 then
      return: a
    else
      return: 10
```

Here the function Random is impure as it is not sure what will be the result when we call the function.

1.4.2 Side-effects (Impure functions)

As you are aware function has side effects when it has observable interaction with the outside world. There are situations our functions can become impure though our goal is to make our functions pure. Just to clarify remember that side effect is not a necessary bad thing.Sometimes they are useful *(especially outside functional programming paradigm)*.

Modify variable outside a function

One of the most popular groups of side effects is modifying the variable outside of function.

For example

```
let y: = 0
(int) inc (int) x:
      y: = y + x;
      return (y)
```

In the above example the value of y get changed inside the function definition due to which the result will change each time. The side effect of the inc () function is it is changing the data of the external visible variable ***'y'***. As you can see some side effects are quite easy to spot and some of them may tricky. A good sign that our function impure *(has side effect)* is that it doesn't take any arguments and it doesn't return any value.

From all these examples and definitions what we can understand about the main differences between pure and impure functions are

Pure Function	Impure Function
The return value of the pure functions solely depends on its arguments passed. Hence, if you call the pure functions with the same set of arguments, you will always get the same return values. They do not have any side effects.	The return value of the impure functions does not solely depend on its arguments passed. Hence, if you call the impure functions with the same set of arguments, you might get the different return values For example, random(), Date().
They do not modify the arguments which are passed to them	They may modify the arguments which are passed to them

Now let's see the example of a pure function to determine the greatest common divisor *(gcd)* of two positive integer numbers.

```
let rec gcd a b :=
    if b <> 0 then gcd b (a mod b) else return a;

output
    gcd 13 27;;
    - : int = 1
    gcd 20536 7826;;
    - : int = 2
```

In the above example program *'gcd'* is the name of the function which recursively called till the variable *'b'* becomes *'0'*. Remember *b* and *(a mod b)* are two arguments passed to *'a'* and *'b'* of the gcd function.

1.4.3 Chameleons of Chromeland problem using function

Recall the In the Chameleons of Chromeland problem what you have studied in class XI. suppose two types of chameleons are equal in number. Construct an algorithm that arranges meetings between these two types so that they change their color to the third type. In the end, all should display the same color.

Let us represent the number of chameleons of each type by variables *a, b* and *c,* and their initial values by *A, B* and *C,* respectively. Let a = b be the input property.

The input – output relation is a = b = 0 and c = A + B + C. Let us name the algorithm monochromatize. The algorithm can be specified as

monochromatize (a, b, c)

-- inputs	:	a = A, b = B, c = C, a = b
-- outputs	:	a = b = 0, c = A+B+C

In each iterative step, two chameleons of the two types *(equal in number)* meet and change their colors to the third one. For example, if A, B, C = 4, 4, 6, then the series of meeting will result in

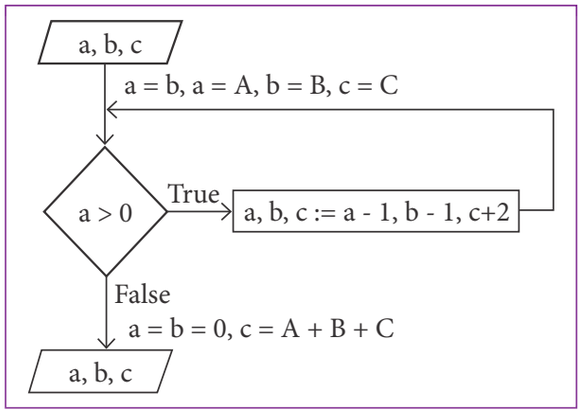
iteration	a	b	c
0	4	4	4
1	3	3	8
2	2	2	10
3	1	1	12
4	0	0	14

XII Std - CS EM Chapter-1.indd 6 24-01-2020 12:38:55

In each meeting, ***a*** and ***b*** each decreases by ***1***, and ***c*** increases by ***2***. The solution can be expressed as an iterative algorithm.

```
monochromatize (a, b,  c)
      -- inputs : a = A, b=B, c=C, a=b
      -- outputs : a = b = 0, c = A+B+C
while a>0
        a, b, c := a-1, b-1, c+2
```

The algorithm is depicted in the flowchart as below



Now let us write this algorithm using function

```
let rec monochromatize a  b  c :=
      if a > 0 then
        a, b, c := a-1, b-1, c+2
      else
        a:=0, b:=0, c:= a + b + c
        return c
```

Points to remember:

- Algorithms are expressed using statements of a programming language
- Subroutines are small sections of code that are used to perform a particular task that can be used repeatedly
- A function is a unit of code that is often defined within a greater code structure
- A function contains a set of code that works on many kinds of inputs and produces a concrete output
- Definitions are distinct syntactic blocks
- Parameters are the variables in a function definition and arguments are the values which are passed to a function definition through the function definition.
- When you write the type annotations the parentheses are mandatory in the function definition
- An interface is a set of action that an object can do
- Interface just defines what an object can do, but won't actually do it
- Implementation carries out the instructions defined in the interface
- Pure functions are functions which will give exact result when the same arguments are passed
- The variables used inside the function may cause side effects though the functions which are not passed with any arguments. In such cases the function is called impure function

XII Std - CS EM Chapter-1.indd 7 24-01-2020 12:38:55

Hands on Practice

1. Write algorithmic function definition to find the minimum among 3 numbers.
2. Write algorithmic recursive function definition to find the sum of n natural numbers.

4PJPH2

Evaluation

Part - I

Choose the best answer **(1 Mark)**

1. The small sections of code that are used to perform a particular task is called

   (A) Subroutines (B) Files (C) Pseudo code (D) Modules

2. Which of the following is a unit of code that is often defined within a greater code structure?

   (A) Subroutines (B) Function (C) Files (D) Modules

3. Which of the following is a distinct syntactic block?

   (A) Subroutines (B) Function (C) Definition (D) Modules

4. The variables in a function definition are called as

   (A) Subroutines (B) Function (C) Definition (D) Parameters

5. The values which are passed to a function definition are called

   (A) Arguments (B) Subroutines (C) Function (D) Definition

6. Which of the following are mandatory to write the type annotations in the function definition?

   (A) Curly braces (B) Parentheses (C) Square brackets (D) indentations

7. Which of the following defines what an object can do?

   (A) Operating System (B) Compiler (C) Interface (D) Interpreter

8. Which of the following carries out the instructions defined in the interface?

   (A) Operating System (B) Compiler (C) Implementation (D) Interpreter

9. The functions which will give exact result when same arguments are passed are called

   (A) Impure functions (B) Partial Functions

   (C) Dynamic Functions (D) Pure functions

XII Std - CS EM Chapter-1.indd 8 24-01-2020 12:38:56

10. The functions which cause side effects to the arguments passed are called

(A) impure function (B) Partial Functions

(C) Dynamic Functions (D) Pure functions

Part - II

Answer the following questions **(2 Marks)**

1. What is a subroutine?
2. Define Function with respect to Programming language.
3. Write the inference you get from X:=(78).
4. Differentiate interface and implementation.
5. Which of the following is a normal function definition and which is recursive function definition

   i) let rec sum x y:

   return x + y

   ii) let disp :

   print 'welcome'

   iii) let rec sum num:

   if (num!=0) then return num + sum (num-1)

   else

   return num

Part - III

Answer the following questions **(3 Marks)**

1. Mention the characteristics of Interface.
2. Why strlen is called pure function?
3. What is the side effect of impure function. Give example.
4. Differentiate pure and impure function.
5. Wha happens if you modify a variable outside the function? Give an example.

XII Std - CS EM Chapter-1.indd 9 24-01-2020 12:38:56

Part - IV

Answer the following questions **(5Marks)**

1. What are called Parameters and write a note on

   (i) Parameter without Type (ii) Parameter with Type

2. Identify in the following program

```
let rec gcd a b :=
    if b <> 0 then gcd b (a mod b) else return a
```

   i) Name of the function

   ii) Identify the statement which tells it is a recursive function

   iii) Name of the argument variable

   iv) Statement which invoke the function recursively

   v) Statement which terminates the recursion

3. Explain with example Pure and impure functions.

4. Explain with an example interface and implementation.

REFERENCES

1. *Data Structures and Algorithms in Python By Michael T.Goodrich, RobertoTamassia and Michael H. Goldwasser.*
2. *Data Structure and Algorithmic Thinking in Python By Narasimha Karumanchi*
3. *https://www.python.org*

XII Std - CS EM Chapter-1.indd 10 24-01-2020 12:38:56

Unit I

CHAPTER 2

DATA ABSTRACTION

5YVDJD

Learning Objectives

After the completion of this chapter, the student will be able to Understand

- what is Abstract Data structures.
- Abstract data type.
- Difference between concrete and abstract implementation.
- Pairs.
- Data Abstration in Structure.

2.1 Data Abstraction- Introduction

Data abstraction is a powerful concept in computer science that allows programmers to treat code as objects — for example, car objects, pencil objects, people objects, etc. Programmers need not to worry about how code is implemented — they have to just know what it does.

This is especially important when several people are doing a project. Here project refers to the programming .With data abstraction, your group members won't have to read through every line of your code to understand. They can just assume that it does work.

Abstraction provides modularity (modularity means splitting a program in to many modules). Classes (structures) are the representation for "Abstract Data Types", (ADT)

2.2 Abstract Data Types

Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of value and a set of operations.

The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations. It is called "abstract" because it gives an implementation independent view. ***The process of providing only the essentials and hiding the details is known as abstraction***.

You can see that these definitions do not specify how these ADTs will be represented and how the operations will be carried out. There can be different ways to implement an ADT, for example, the List ADT can be implemented using singly linked list or doubly linked list. Similarly, stack ADT and Queue ADT can be implemented using lists.

Data abstraction replicate how we think about the world. For example, when you want to drive a car, you don't need to know how the engine was built or what kind of material the tires are made of. You just have to know how to turn the wheel and press the gas pedal. ***To facilitate data abstraction, you will need to create two types of functions: constructors and selectors.***

XII Std - CS EM Chapter-2.indd 11 24-01-2020 12:39:50

2.3 constructors and selectors

Constructors are functions that build the abstract data type. Selectors are functions that retrieve information from the data type.

For example, say you have an abstract data type called city. This city object will hold the city's name, and its latitude and longitude. To create a city object, you'd use a function like

city = makecity (name, lat, lon)

To extract the information of a city object, you would use functions like

- getname(city)
- getlat(city)
- getlon(city)

The following pseudo code will compute the distance between two city objects:

```
distance(city1, city2):
    lt1, lg1 := getlat(city1), getlon(city1)
    lt2, lg2 := getlat(city2), getlon(city2)
    return ((lt1 - lt2)**2 + (lg1 - lg2)**2))^1/2
```

In the above code read distance(), getlat() and getlon() as functions and read lt as latitude and lg longitude. Read := as "assigned as" or "becomes"

lt1, lg1 := getlat(city1), getlon(city1)

is read as lt1 becomes the value of getlat(city1) and lg1 becomes the value of getlon (city1).

Notice that you don't need to know how these functions were implemented. You are assuming that someone else has defined them for us.

It's okay if the end user doesn't know how functions were implemented. However, the functions still have to be defined by someone.

Let us identify the constructors and selectors in the above code

As you already know that Constructors are functions that build the abstract data type. In the above pseudo code the function which creates the object of the city is the constructor.

city = makecity (name, lat, lon)

Here makecity (name, lat, lon) is the constructor which creates the object city.

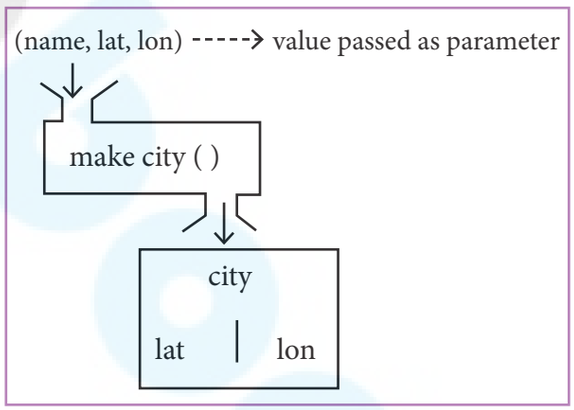


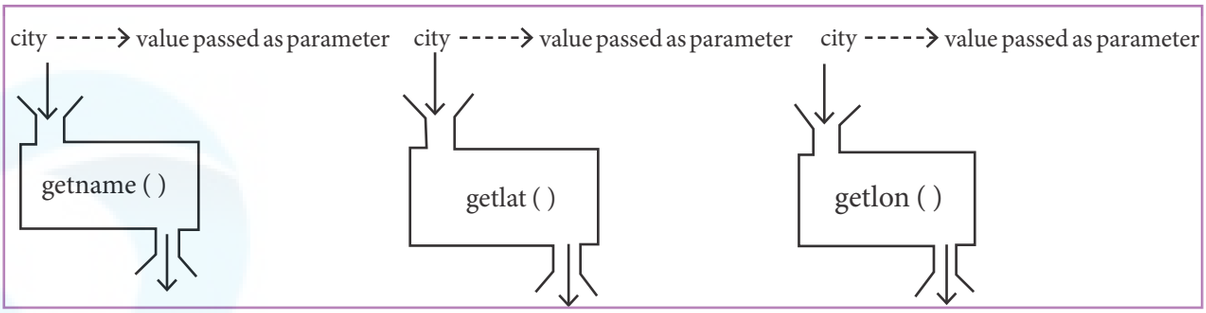
Fig 1 constructor

Selectors are nothing but the functions that retrieve information from the data type. Therefore in the above code

- getname(city)
- getlat(city)
- getlon(city)

are the selectors because these functions extract the information of the city object

XII Std - CS EM Chapter-2.indd 12 24-01-2020 12:39:50



Now let us consider one more example to identify the constructor and selector for a slope.Read - - as comments.

```
- - constructor
makepoint(x, y):
    return x, y
- - selector
 xcoord(point):
    return point[0]
- -selector
ycoord(point):
    return point[1]
```

Note

Data abstraction is supported by defining an abstract data type (ADT), which is a collection of constructors and selectors. Constructors create an object, bundling together different pieces of information, while selectors extract individual pieces of information from the object.

2.4 Representation of Abstract datatype using Rational numbers

The basic idea of data abstraction is to structure programs so that they operate on abstract data. That is, our programs should use data in such a way, as to make as few assumptions about the data as possible. At the same time, a concrete data representation is defined as an independent part of the program.

Note

In concrete data representation, a definition for each function is known

Any program consist of two parts. **The two parts of a program are, the part that operates on abstract data and the part that defines a concrete representation**, is connected by a small set of functions that implement abstract data in terms of the concrete representation. To illustrate this technique, let us consider an example to design a set of functions for manipulating rational numbers.

Example

A rational number is a ratio of integers, and rational numbers constitute an important sub-class of real numbers. A rational number such as 8/3 or 19/23 is typically written as:

<numerator>/<denominator>

where both the <numerator> and <denominator> are placeholders for integer values. Both parts are needed to exactly characterize the value of the rational number. Actually dividing integers produces a float approximation, losing the exact precision of integers.

8/3 =2.6666666666666665

However, you can create an exact representation for rational numbers by combining together the numerator and denominator.

As we know from using functional abstractions, we can start programming productively before you have an implementation of some parts of our program. Let us begin by assuming that you already have a way of constructing a rational number from a numerator and a denominator. You also assume that, given a rational number, you have a way of selecting its numerator and its denominator component. Let us further assume that the constructor and selectors are also available.

We are using here a powerful strategy for designing programs: **'wishful thinking'**. We haven't yet said how a rational number is represented, or how the constructor and selectors should be implemented.

Note

Wishful Thinking is the formation of beliefs and making decisions according to what might be pleasing to imagine instead of by appealing to reality.

Example: An ADT for rational numbers

- - constructor

- - constructs a rational number with numerator n, denominator d

rational(n, d)

- - selector

numer(x) → returns the numerator of rational number x

denom(y) → returns the denominator of rational number y

Now you have the operations on rational numbers defined in terms of the selector functions numer and denom, and the constructor function rational, but you haven't yet defined these functions. What you need is some way to glue together a numerator and a denominator into a compound value.

The pseudo code for the representation of the rational number using the above constructor and selector is

```
x,y:=8,3
rational(n,d)
  numer(x)/denom(y)
 - - output :  2.6666666666666665
```

2.5 Lists,Tuples

To enable us to implement the concrete level of our data abstraction, Some languages like Python provides a compound structure called Pair which is made up of list or Tuple. The first way to implement pairs is with the List construct.

2.5.1 List

List is constructed by placing expressions within square brackets separated by commas. Such an expression is called a list literal. List can store multiple values. Each value can be of any type and can even be another list.

XII Std - CS EM Chapter-2.indd 14 24-01-2020 12:39:51

Example for List is [10, 20].

The elements of a list can be accessed in two ways. The first way is via our familiar method of multiple assignment, which unpacks a list into its elements and binds each element to a different name.

```
lst := [10, 20]

x, y := lst
```

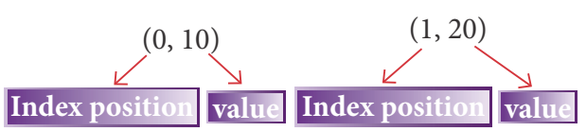
In the above example ***x*** will become10 and ***y*** will become 20.

A second method for accessing the elements in a list is by the element selection operator, also expressed using square brackets. Unlike a list literal, a square-brackets expression directly following another expression does not evaluate to a list value, but instead selects an element from the value of the preceding expression.

```
lst[0]
10
lst[1]
20
```

In both the example mentioned above mathematically we can represent list similar to a set.

lst[(0, 10), (1, 20)] - where



Any way of bundling two values together into one can be considered as a pair. Lists are a common method to do so. Therefore List can be called as Pairs.

Representing Rational Numbers Using List

You can now represent a rational number as a pair of two integers in pseudo code : a numerator and a denominator.

```
rational(n, d):
    return [n, d]
numer(x):
    return x[0]
denom(x):
    return x[1]
```

2.5.2 Tuple

Remember, a pair is a compound data type that holds two other pieces of data. So far,we have provided you with two ways of representing the pair data type. The first way is using List construct and the second way to implement pairs is with the tuple construct.

A tuple is a comma-separated sequence of values surrounded with parentheses. Tuple is similar to a list. The difference between the two is that you cannot change the elements of a tuple once it is assigned whereas in a list, elements can be changed.

Example colour= ('red', 'blue', 'Green')

Representation of Tuple as a Pair

```
nums := (1, 2)
nums[0]
1
nums[1]
2
```

Note the square bracket notation is used to access the data you stored in the pair. The data is zero indexed, meaning you access the first element with nums[0] and the second with nums[1].

XII Std - CS EM Chapter-2.indd 15
24-01-2020 12:39:51

2.6 Data Abstraction in Structure

As you already know that List allow data abstraction in that you can give a name to a set of memory cells. For instance, in the game Mastermind, you must keep track of a list of four colors that the player guesses. Instead of using four separate variables (color1, color2, color3, and color4) you can use a single variable 'Predict', e.g.,

Predict =['red', 'blue', 'green', 'green']

What lists do not allow us to do is name the various parts of a multi- item object. In the case of a Predict, you don't really need to name the parts:

using an index to get to each color suffices.

But in the case of something more complex, like a person, we have a multi- item object where each 'item' is a named thing: the firstName, the lastName, the id, and the email. One could use a list to represent a person:

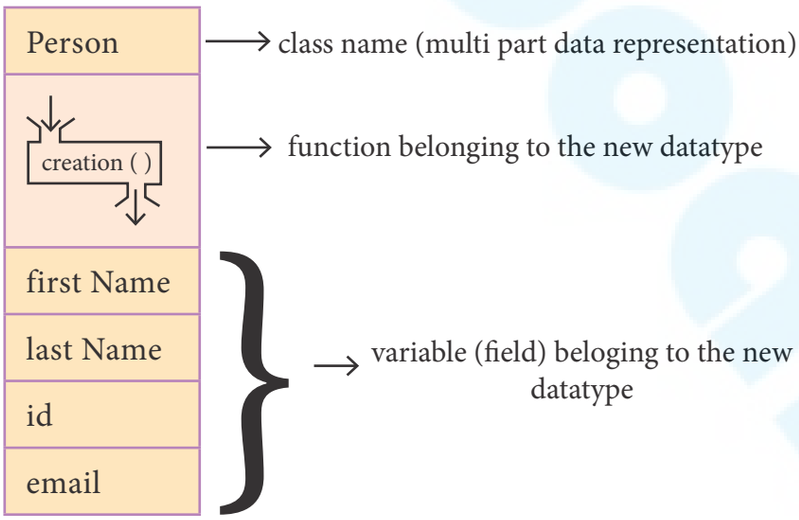
person=['Padmashri', 'Baskar', '994-222-1234', 'compsci@gmail.com']

but such a representation doesn't explicitly specify what each part represents.

For this problem instead of using a list, you can use the structure construct (In OOP languages it's called class construct) to represent multi-part objects where each part is named (given a name). Consider the following pseudo code:

```
class Person:
    creation( )
    firstName := " "
    lastName := " "
    id := " "
    email := " "
```

The new data type Person is pictorially represented as



XII Std - CS EM Chapter-2.indd 16 24-01-2020 12:39:51

Let main() contains	
p1:=Person()	statement creates the object.
firstName := " Padmashri "	setting a field called firstName with value Padmashri
lastName :="Baskar"	setting a field called lastName with value Baskar
id :="994-222-1234"	setting a field called id value 994-222-1234
email="compsci@gmail.com"	setting a field called email with value compsci@gmail.com
- - output of firstName : Padmashri	

The class (structure) construct defines the form for multi-part objects that represent a person. Its definition adds a new data type, in this case a type named Person. Once defined, we can create new variables (instances) of the type. In this example ***Person is referred to as a class or a type, while p1 is referred to as an object or an instance.*** You can think of class Person as a cookie cutter, and p1 as a particular cookie. Using the cookie cutter you can make many cookies. Same way using class you can create many objects of that type.

So far, you've seen how a class defines a data abstraction by grouping related data items. A class is not just data, it has functions defined within it. We say such functions are subordinate to the class because their job is to do things with the data of the class, e.g., to modify or analyze the data of a Person object.

Therefore we can define ***a class as bundled data and the functions that work on that data***. From All the above example and explanation one can conclude the beauty of data abstraction is that we can treat complex data in a very simple way.

Points to remember:

- Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of value and a set of operations.
- The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented.
- ADT does not specify how data will be organized in memory and what algorithms will be used for implementing the operations
- Constructors are functions that build the abstract data type.
- Selectors are functions that retrieve information from the data type.
- Concrete data types or structures (CDT's) are direct implementations of a relatively simple concept.
- Abstract Data Types (ADT's) offer a high level view (and use) of a concept independent of its implementation.

Points to remember:

- A concrete data type is a data type whose representation is known and in abstract data type the representation of a data type is unknown
- Pair is a compound structure which is made up of list or Tuple
- List is constructed by placing expressions within square brackets separated by commas
- The elements of a list can be accessed in two ways. The first way is via multiple assignment and the second method is by the element selection operator
- Bundling two values together into one can be considered as a pair
- List does not allow to name the various parts of a multi-item object.

6EE8KP

Evaluation

Part - I

Choose the best answer **(1 Mark)**

1. Which of the following functions that build the abstract data type ?

   (A) Constructors (B) Destructors (C) recursive (D)Nested

2. Which of the following functions that retrieve information from the data type?

   (A) Constructors (B) Selectors (C) recursive (D)Nested

3. The data structure which is a mutable ordered sequence of elements is called

   (A) Built in (B) List (C) Tuple (D) Derived data

4. A sequence of immutable objects is called

   (A) Built in (B) List (C) Tuple (D) Derived data

5. The data type whose representation is known are called

   (A) Built in datatype (B) Derived datatype

   (C) Concrete datatype (D) Abstract datatype

6. The data type whose representation is unknown are called

   (A) Built in datatype (B) Derived datatype

   (C) Concrete datatype (D) Abstract datatype

7. Which of the following is a compound structure?

   (A) Pair (B) Triplet (C) single (D) quadrat

XII Std - CS EM Chapter-2.indd 18 24-01-2020 12:39:51

8. Bundling two values together into one can be considered as

(A) Pair (B) Triplet (C) single (D) quadrat

9. Which of the following allow to name the various parts of a multi-item object?

(A) Tuples (B) Lists (C) Classes (D) quadrats

10. Which of the following is constructed by placing expressions within square brackets?

(A) Tuples (B) Lists (C) Classes (D) quadrats

Part - II

Answer the following questions **(2 Marks)**

1. What is abstract data type?
2. Differentiate constructors and selectors.
3. What is a Pair? Give an example.
4. What is a List? Give an example.
5. What is a Tuple? Give an example.

Part - III

Answer the following questions **(3 Marks)**

1. Differentiate Concrete data type and abstract datatype.
2. Which strategy is used for program designing? Define that Strategy.
3. Identify Which of the following are constructors and selectors?

   (a) N1=number() (b) accetnum(n1) (c) displaynum(n1)

   (d) eval(a/b) (e) x,y= makeslope (m), makeslope(n)

   (f) display()
4. What are the different ways to access the elements of a list. Give example.
5. Identify Which of the following are List, Tuple and class ?

   (a) arr [1, 2, 34] (b) arr (1, 2, 34) (c) student [rno, name, mark]

   (d) day= ('sun', 'mon', 'tue', 'wed') (e) x= [2, 5, 6.5, [5, 6], 8.2]

   (f) employee [eno, ename, esal, eaddress]

XII Std - CS EM Chapter-2.indd 19 24-01-2020 12:39:51

Part - IV

Answer the following questions **(5Marks)**

1. How will you facilitate data abstraction. Explain it with suitable example
2. What is a List? Why List can be called as Pairs. Explain with suitable example
3. How will you access the multi-item. Explain with example.

Reference Books

1. *Data structure and algorithmic thinking with python by narasimha karumanchi*
2. *sign and analysis of algorithms by s sridhar*
3. *Data Structures and Algorithms in Python by Goodrich, Tamassia & Goldwasser*
4. *https://www.tutorialspoint.com*

XII Std - CS EM Chapter-2.indd 20 24-01-2020 12:39:51

Unit I

CHAPTER 3

SCOPING

7NPWN1

Learning Objectives

After the completion of this chapter, the student will be able to

- Understand what is Scoping
- Able to implement the LEGB rule
- Understand what is module
- Understand the implementation of access control in programming language

3.1 Introduction

Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program. In other words, which parts of your program can see or use it. Normally, every variable defined in a program has global scope. Once defined, every part of your program can access that variable. But it is a good practice to limit a variable's scope to a single definition. This way, changes inside the function can't affect the variable on the outside of the function in unexpected ways.

3.2 Variable Scope

To understand the scope of variables in a programming language, it is important to learn about what variables really are. Essentially, they're addresses (references, or pointers), to an object in memory. When you assign a variable with := to an instance (object), you're binding (or mapping) the variable to that instance. Multiple variables can be mapped to the same instance.

Note

The process of binding a variable name with an object is called mapping. : = (colon equal to sign) is used in programming languages to map the variable and object.

Programming languages keeps track of all these mappings with namespaces. ***Namespaces are containers for mapping names of variables to objects***. You can think of them as dictionaries, containing list of words and its meanings. The words are mapped with its meaning in dictionaries whereas names are mapped with objects (name : = object) in programming language. This allows access to objects by names you choose to assign to them.

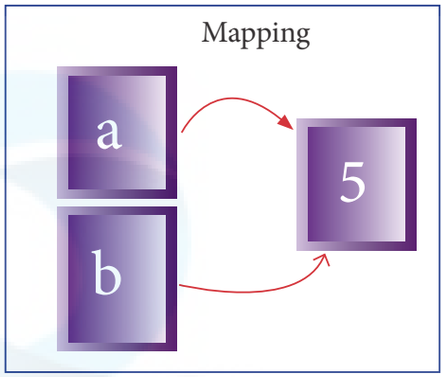
In the following example, ***a*** is first mapped to the integer ***5***. In this case, ***a*** is the variable name, while the integer value ***5*** is the object.

Then, ***b*** is set equal to ***a***. This actually means that ***b*** is now bound to the same integer value as ***a***, which is ***5***.

1. a:=5

2. b:=a

XII Std - CS EM Chapter-3.indd 21 24-01-2020 12:41:18

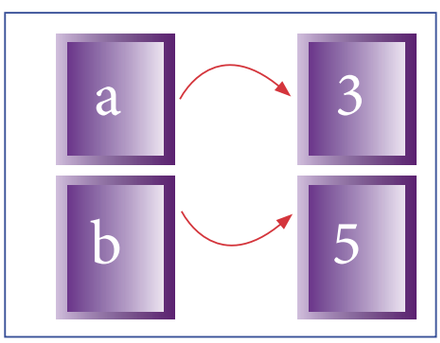


Now consider the following extension of the above statements

1. a:=5
2. b:=a
3. a:=3

If you then change ***a*** to be equal to ***3***, a budding programmer might expect ***b*** also be equal to 3, but that is not the case. ***b*** is still mapped (or pointing) to the integer value of 5. The only thing that changed is ***a***, which is now mapped to the integer value 3.

Mapping after changing the value of ***a***



The scope of a variable is that part of the code where it is visible. Actually, to refer to it, you don't need to use any prefixes then. Let's take an example,

1. **Disp**():
2. a:=7

When you try to display the value of ***a*** outside the procedure the program flags the error ***"name 'a' is not defined"***. This is because the lifetime of the variable is only till the end of the procedure. Also, ***the duration for which a variable is alive is called its 'life time'.***

3.3 LEGB rule

Scope also defines the order in which variables have to be mapped to the object in order to obtain the value. Let us take a simple example as shown below:

1. x:= 'outer x variable'
2. **display():**
3. x:= 'inner x variable'
4. print x
5. display()

When the above statements are executed the statement (4) and (5) display the result as

Output

outer x variable

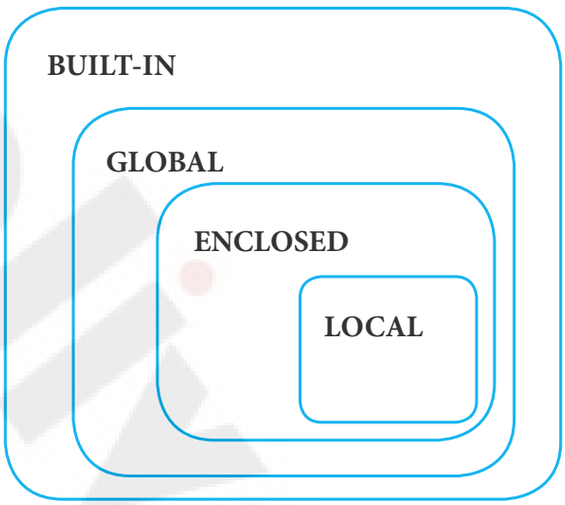
inner x variable

Above statements give different outputs because the same variable name ***x*** resides in different scopes, one inside the function display() and the other in the upper level. The value ***'outer xvariable'*** is printed when ***x*** is referenced outside the function definition. Whereas when display() gets executed, ***'inner x variable'*** is printed which is the ***x*** value inside the function definition. From the above example, we can guess that there is a rule followed, in order to decide from which scope a variable has to be picked.

The ***LEGB*** rule is used to decide the order in which the scopes are to be searched for scope resolution. The scopes are listed below in terms of hierarchy (highest to lowest).

XII Std - CS EM Chapter-3.indd 22 24-01-2020 12:41:18

Local(**L**)	Defined inside function/class
Enclosed(**E**)	Defined inside enclosing functions (Nested function concept)
Global(**G**)	Defined at the uppermost level
Built-in (**B**)	Reserved names in built-in functions (modules)



3.4 Types of Variable Scope

There are 4 types of Variable Scope, let's discuss them one by one:

3.4.1 Local Scope

Local scope refers to variables defined in current function. Always, a function will first look up for a variable name in its local scope. Only if it does not find it there, the outer scopes are checked.

Look at this example

1. **Disp():** 2. a:=7 3. print a 4. Disp()	Entire program Disp( ): a:=7 print a Disp ( )	Output of the Program 7

On execution of the above code the variable ***a*** displays the value 7, because it is defined and available in the local scope.

XII Std - CS EM Chapter-3.indd 23 24-01-2020 12:41:18

3.4.2 Global Scope

A variable which is declared outside of all the functions in a program is known as global variable. This means, global variable can be accessed inside or outside of all the functions in a program. Consider the following example

1. a:=10 2. **Disp()**: 3. a:=7 4. print a 5. Disp() 6. print a	Entire program a:=10 Disp( ) a:=7 print a Disp1( ): print a	Output of the Program 7 10

On execution of the above code the variable ***a*** which is defined inside the function displays the value 7 for the function call Disp() and then it displays 10, because ***a*** is defined in global scope.

3.4.3 Enclosed Scope

All programming languages permit functions to be nested. A function (method) with in another function is called nested function. ***A variable which is declared inside a function which contains another function definition with in it, the inner function can also access the variable of the outer function. This scope is called enclosed scope***.

When a compiler or interpreter search for a variable in a program, it first search Local, and then search Enclosing scopes. Consider the following example

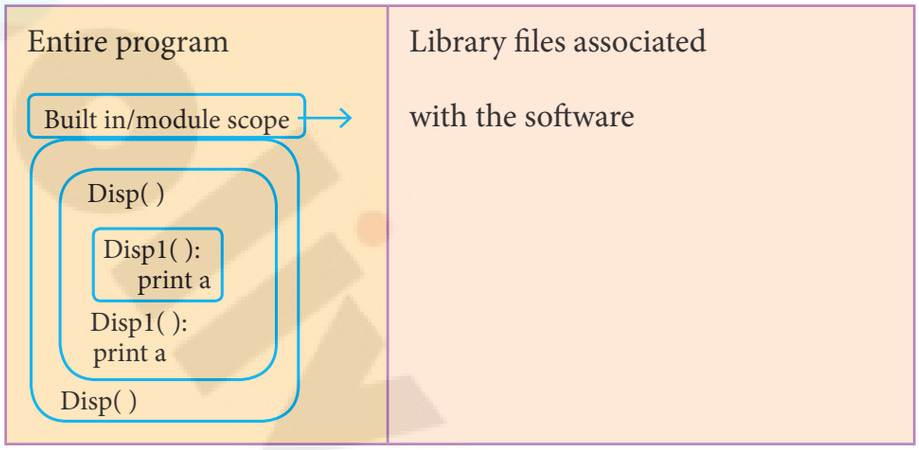
1. Disp(): 2. a:=10 3. Disp1(): 4. print a 5. Disp1() 6. print a 7. Disp()	Entire program Disp( ) a:=10 Disp 1( ): print a Disp 1( ): print a Disp( )	Output of the Program 10 10

XII Std - CS EM Chapter-3.indd 24 24-01-2020 12:41:18

In the above example Disp1() is defined with in Disp(). The variable 'a' defined in Disp() can be even used by Disp1() because it is also a member of Disp().

3.4.4 Built-in Scope

Finally, we discuss about the widest scope. The built-in scope has all the names that are pre-loaded into the program scope when we start the compiler or interpreter. Any variable or module which is defined in the library functions of a programming language has Built-in or module scope. They are loaded as soon as the library files are imported to the program.



Normally only Functions or modules come along with the software, as packages. Therefore they will come under Built in scope.

3.5 Module

A module is a part of a program. Programs are composed of one or more independently developed modules. A single module can contain one or several statements closely related each other. Modules work perfectly on individual level and can be integrated with other modules. A software program can be divided into modules to ease the job of programming and debugging as well. A program can be divided into small functional modules that work together to get the output. The process of subdividing a computer program into separate sub-programs is called Modular programming. Modular programming enables programmers to divide up the work and debug pieces of the program independently. The examples of modules are procedures, subroutines, and functions.

3.5.1 Characteristics of Modules

The following are the desirable characteristics of a module.

1. Modules contain instructions, processing logic, and data.
2. Modules can be separately compiled and stored in a library.
3. Modules can be included in a program.
4. Module segments can be used by invoking a name and some parameters.
5. Module segments can be used by other modules.

XII Std - CS EM Chapter-3.indd 25
24-01-2020 12:41:18

3.5.2 The benefits of using modular programming include

- Less code to be written.
- A single procedure can be developed for reuse, eliminating the need to retype the code many times.
- Programs can be designed more easily because a small team deals with only a small part of the entire code.
- Modular programming allows many programmers to collaborate on the same application.
- The code is stored across multiple files.
- Code is short, simple and easy to understand.
- Errors can easily be identified, as they are localized to a subroutine or function.
- The same code can be used in many applications.
- The scoping of variables can easily be controlled.

3.5.3 Access Control

Access control is a security technique that regulates who or what can view or use resources in a computing environment. It is a fundamental concept in security that minimizes risk to the object. In other words access control is a selective restriction of access to data. IN Object oriented programming languages it is implemented through access modifiers. Classical object-oriented languages, such as C++ and Java, control the access to class members by public, private and protected keywords. Private members of a class are denied access from the outside the class. They can be handled only from within the class.

Public members (generally methods declared in a class) are accessible from outside the class. The object of the same class is required to invoke a public method. ***This arrangement of private instance variables and public methods ensures the principle of data encapsulation***.

Protected members of a class are accessible from within the class and are also available to its sub-classes. No other process is permitted access to it. This enables specific resources of the parent class to be inherited by the child class.

Python doesn't have any mechanism that effectively restricts access to any instance variable or method. Python prescribes a convention of prefixing the name of the variable or method with single or double underscore to emulate the behaviour of protected and private access specifiers.

All members in a Python class are public by default, whereas by default in C++ and java they are private. Any member can be accessed from outside the class environment in Python which is not possible in C++ and java.

XII Std - CS EM Chapter-3.indd 26 24-01-2020 12:41:18

Points to remember:

- Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.
- The process of binding a variable name with an object is called mapping. : = (colon and equal to sign) is used in programming languages to map the variable and object.
- Namespaces are containers for mapping names of variables to objects.
- The scope of a variable is that part of the code where it is visible.
- The LEGB rule is used to decide the order in which the scopes are to be searched for scope resolution.
- Local scope refers to variables defined in current function.
- A variable which is declared outside of all the functions in a program is known as global variable.
- A function (method) with in another function is called nested function.
- A variable which is declared inside a function which contains another function definition with in it, the inner function can also access the variable of the outer function. This scope is called enclosed scope.
- Built-in scope has all the names that are pre-loaded into program scope when we start the compiler or interpreter.
- A module is a part of a program. Programs are composed of one or more independently developed modules.
- The process of subdividing a computer program into separate sub-programs is called Modular programming.
- Access control is a security technique that regulates who or what can view or use resources in a computing environment. It is a fundamental concept in security that minimizes risk to the object.
- Public members (generally methods declared in a class) are accessible from outside the class.
- Protected members of a class are accessible from within the class and are also available to its sub-classes
- Private members of a class are denied access from the outside the class. They can be handled only from within the class.
- Python prescribes a convention of prefixing the name of the variable/method with single or double underscore to emulate the behaviour of protected and private access specifiers.
- C++ and Java, control the access to class members by public, private and protected keywords
- All members in a Python class are public by default whereas by default in C++ and java all members are private.

XII Std - CS EM Chapter-3.indd 27 24-01-2020 12:41:18

Hands on Practice

1. Observe the following diagram and Write the pseudo code for the following

```
sum( )
  num1:=20
  sum1( )
  num1:=num1 + 10
    sum2( )
    num1: = num1 + 10
  sum2( )
sum1( )
num1:=10
sum( )
print num1
```

8IHQP1

Evaluation

Part - I

Choose the best answer **(1 Mark)**

1. Which of the following refers to the visibility of variablesin one part of a program to another part of the same program.

   (A) Scope (B) Memory (C) Address (D) Accessibility

2. The process of binding a variable name with an object is called

   (A) Scope (B) Mapping (C) late binding (D) early binding

3. Which of the following is used in programming languages to map the variable and object?

   (A) :: (B) := (C) = (D) ==

4. Containers for mapping names of variables to objects is called

   (A) Scope (B) Mapping (C) Binding (D) Namespaces

5. Which scope refers to variables defined in current function?

   (A) Local Scope (B) Global scope

   (C) Module scope (D) Function Scope

XII Std - CS EM Chapter-3.indd 28 24-01-2020 12:41:18

6. The process of subdividing a computer program into separate sub-programs is called

(A) Procedural Programming (B) Modular programming

(C)Event Driven Programming (D) Object oriented Programming

7. Which of the following security technique that regulates who canuse resources in a computing environment?

(A) Password (B)Authentication

(C) Access control (D) Certification

8. Which of the following members of a class can be handled only from within the class?

(A) Public members (B)Protected members

(C) Secured members (D) Private members

9. Which members are accessible from outside the class?

(A) Public members (B)Protected members

(C) Secured members (D) Private members

10. The members that are accessible from within the class and are also available to its sub-classes is called

(A) Public members (B)Protected members

(C) Secured members (D) Private members

Part - II

Answer the following questions **(2 Marks)**

1. What is a scope?
2. Why scope should be used for variable. State the reason.
3. What is Mapping?
4. What do you mean by Namespaces?
5. How Python represents the private and protected Access specifiers?

Part - III

Answer the following questions **(3 Marks)**

1. Define Local scope with an example.
2. Define Global scope with an example.
3. Define Enclosed scope with an example.

XII Std - CS EM Chapter-3.indd 29 24-01-2020 12:41:18

4. Why access control is required?

5. Identify the scope of the variables in the following pseudo code and write its output

```
color:= Red
mycolor():
  b:=Blue
  myfavcolor():
      g:=Green
      printcolor, b, g
  myfavcolor()
  printcolor, b
mycolor()
print color
```

Part - IV

Answer the following questions **(5Marks)**

1 Explain the types of scopes for variable or LEGB rule with example.

2. Write any Five Characteristics of Modules.

3. Write any five benefits in using modular programming.

REFERENCES

1. *Data Structures and Algorithms in Python By Michael T.Goodrich, RobertoTamassia and Michael H. Goldwasser.*
2. *Data Structure and Algorithmic Thinking in Python By Narasimha Karumanchi*
3. *https://www.python.org*

XII Std - CS EM Chapter-3.indd 30 24-01-2020 12:41:18

Unit I

CHAPTER 4

ALGORITHMIC STRATEGIES

8Y1KQC

Learning Objectives

At the end of this chapter the students will be able to:

- Know the basics and technical perspective of algorithms.
- Understand the efficiency, time and space complexity of an algorithm.
- Develop and analyze algorithms for searching and sorting.
- Learn about dynamic programming through algorithmic approach.

4.1 Introduction to Algorithmic strategies

An algorithm is a finite set of instructions to accomplish a particular task. It is a step-by-step procedure for solving a given problem. An algorithm can be implemented in any suitable programming language.

Algorithms must have input, output and should satisfy the following characteristics such as definiteness, correctness and effectiveness. Data are maintained and manipulated effectively through data structures. Algorithms can be developed to store, manipulate and retrieve data from such data structures. Examples for data structures are arrays, structures, list, tuples, dictionary etc.

Search	To search an item in a data structure using linear and binary search.
Sort	To sort items in a certain order using the methods such as bubble sort, insertion sort, selection sort, etc.
Insert	To insert an item (s) in a data structure.
Update	To update an existing item (s) in a data structure.
Delete	To delete an existing item (s) in a data structure.

The way of defining an algorithm is called algorithmic strategy. For example to calculate factorial for the given value **n** then it can be done by defining the function to calculate factorial once for the iteration-1 then it can be called recursively until the number of required iteration is reached.

DO YOU KNOW?

The word Algorithm comes from the name of a Persian author, Abu Jafar Mohammed ibn Musa al Khowarizmi(c. 825 AD(CE)), who wrote a textbook on mathematics. The word Algorithm has come to refer to a method to solve a problem.

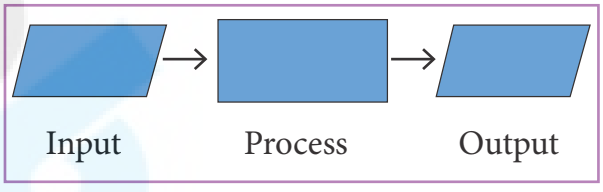
4.1.1 Characteristics of an Algorithm

An algorithm should have the following characteristics:

Input	Zero or more quantities to be supplied.
Output	At least one quantity is produced.
Finiteness	Algorithms must terminate after finite number of steps.
Definiteness	All operations should be well defined. For example operations involving division by zero or taking square root for negative number are unacceptable.
Effectiveness	Every instruction must be carried out effectively.
Correctness	The algorithms should be error free.
Simplicity	Easy to implement.
Unambiguous	Algorithm should be clear and unambiguous. Each of its steps and their inputs/outputs should be clear and must lead to only one meaning.
Feasibility	Should be feasible with the available resources.
Portable	An algorithm should be generic, independent of any programming language or an operating system able to handle all range of inputs.
Independent	An algorithm should have step-by-step directions, which should be independent of any programming code.

4.1.2 Writing an Algorithm

Algorithms are generic and not limited to computer alone. It can be used in various real time activities also. Knowingly or unknowingly we perform many algorithms in our daily life such as packing books in school bag, finding shortest path to search a place, scheduling day-to-day activities, preparation for examination, etc. As we know that all programming languages share basic code constructs like conditions and iterations can be used to write an algorithm. A typical algorithm is shown in the following Figure 4.1.



A Typical Algorithm

Example

Consider the example of Coffee preparation. To make coffee, we need to have the following ingredients: Water, milk, coffee powder and sugar. These ingredients are the inputs of an algorithm. Preparing a cup of coffee is called process. The output of this process is coffee.

The procedure for preparing coffee is as follows:

1. Take a bowl with coffee powder
2. Boil the water and pour it into the bowl

3. Filter it
4. Boil milk
5. Mix sugar and filtered coffee along with boiled milk
6. Pour the coffee into the cup to serve

This kind of procedure can be represented using an algorithm. Thus, the algorithm consists of step-step-by instructions that are required to accomplish a task and helps the programmer to develop the program.

Problem: Design an algorithm to find square of the given number and display the result.

The algorithm can be written as:

Step 1 – start the process

Step 2 – get the input x

Step 3 –calculate the square by multiplying the input value ie., square ← x* x

Step 4 – display the result square

Step 5 – stop

Algorithm could be designed to get a solution of a given problem. A problem can be solved in many ways. Among many algorithms the optimistic one can be taken for implementation.

DO YOU KNOW?

An algorithm that yields expected output for a valid input is called an algorithmic solution.

Algorithm	Program
• Algorithm helps to solve a given problem logically and it can be contrasted with the program	• Program is an expression of algorithm in a programming language
• Algorithm can be categorized based on their implementation methods, design techniques etc	• Algorithm can be implemented by structured or object oriented programming approach
• There is no specific rules for algorithm writing but some guidelines should be followed.	• Program should be written for the selected language with specific syntax
• Algorithm resembles a pseudo code which can be implemented in any language	• Program is more specific to a programming language

Table 4.1 Algorithm Vs Program

4.1.3. Analysis of Algorithm

Computer resources are limited. Efficiency of an algorithm is defined by the utilization of time and space complexity.

Analysis of an algorithm usually deals with the running and execution time of various operations involved. The running time of an operation is calculated as how many programming instructions executed per operation.

Analysis of algorithms and performance evaluation can be divided into two different phases:

1. **A Priori estimates:** This is a theoretical performance analysis of an algorithm. Efficiency of an algorithm is measured by assuming the external factors.

2. **A Posteriori testing:** This is called performance measurement. In this analysis, actual statistics like running time and required for the algorithm executions are collected.

DO YOU KNOW?

An estimation of the time and space complexities of an algorithm for varying input sizes is called algorithm analysis.

4.2 Complexity of an Algorithm

Suppose A is an algorithm and n is the size of input data, the time and space used by the algorithm A are the two main factors, which decide the efficiency of A.

Time Factor -Time is measured by counting the number of key operations like comparisons in the sorting algorithm.

Space Factor - Space is measured by the maximum memory space required by the algorithm.

The complexity of an algorithm f (n) gives the running time and/or the storage space required by the algorithm in terms of n as the size of input data.

4.2.1 Time Complexity

The Time complexity of an algorithm is given by the number of steps taken by the algorithm to complete the process.

4.2.2. Space Complexity

Space complexity of an algorithm is the amount of memory required to run to its completion. The space required by an algorithm is equal to the sum of the following two components:

A fixed part is defined as the total space required to store certain data and variables for an algorithm. For example, simple variables and constants used in an algorithm.

A variable part is defined as the total space required by variables, which sizes depends on the problem and its iteration. For example: recursion used to calculate factorial of a given value n.

4.3 Efficiency of an algorithm

Computer resources are limited that should be utilized efficiently. The efficiency of an algorithm is defined as the number of computational resources used by the algorithm. An algorithm must be analyzed to determine its resource usage. The efficiency of an algorithm can be measured based on the usage of different resources.

For maximum efficiency of algorithm we wish to minimize resource usage. The important resources such as time and space complexity cannot be compared directly,

so time and space complexity could be considered for an algorithmic efficiency.

4.3.1 Method for determining Efficiency

The efficiency of an algorithm depends on how efficiently it uses time and memory space.

The time efficiency of an algorithm is measured by different factors. For example, write a program for a defined algorithm, execute it by using any programming language, and measure the total time it takes to run. The execution time that you measure in this case would depend on a number of factors such as:

- Speed of the machine
- Compiler and other system Software tools
- Operating System
- Programming language used
- Volume of data required

However, to determine how efficiently an algorithm solves a given problem, you would like to determine how the execution time is affected by the nature of the algorithm. Therefore, we need to develop fundamental laws that determine the efficiency of a program in terms of the nature of the underlying algorithm.

DO YOU KNOW?

A way of designing algorithm is called algorithmic strategy

4.3.2 Space-Time tradeoff

A space-time or time-memory tradeoff is a way of solving in less time by using more storage space or by solving a given algorithm in very little space by spending more time.

To solve a given programming problem, many different algorithms may be used. Some of these algorithms may be extremely time-efficient and others extremely space-efficient.

Time/space trade off refers to a situation where you can reduce the use of memory at the cost of slower program execution, or reduce the running time at the cost of increased memory usage.

DO YOU KNOW?

The best algorithm to solve a given problem is one that requires less space in memory and takes less time to execute its instructions to generate output.

4.3.3 Asymptotic Notations

Asymptotic Notations are languages that uses meaningful statements about time and space complexity. The following three asymptotic notations are mostly used to represent time complexity of algorithms:

(i) Big O

Big O is often used to describe the worst-case of an algorithm.

(ii) Big Ω

Big Omega is the reverse Big O, if Bi O is used to describe the upper bound (worst - case) of a asymptotic function, Big Omega is used to describe the lower bound (best-case).

(iii) Big Θ

When an algorithm has a complexity with lower bound = upper bound, say that an algorithm has a complexity O (n log n) and Ω (n log n), it's actually has the complexity Θ (n log n), which means the running time of that algorithm always falls in n log n in the best-case and worst-case.

4.3.4 Best, Worst, and Average ease Efficiency

Let us assume a list of n number of values stored in an array. Suppose if we want to search a particular element in this list, the algorithm that search the key element in the list among n elements, by comparing the key element with each element in the list sequentially.

The best case would be if the first element in the list matches with the key element to be searched in a list of elements. The efficiency in that case would be expressed as O(1) because only one comparison is enough.

Similarly, the worst case in this scenario would be if the complete list is searched and the element is found only at the end of the list or is not found in the list. The efficiency of an algorithm in that case would be expressed as O(n) because n comparisons required to complete the search.

The average case efficiency of an algorithm can be obtained by finding the average number of comparisons as given below:

Minimum number of comparisons = 1

Maximum number of comparisons = n

If the element not found then maximum number of comparison = n

Therefore, average number of comparisons = (n + 1)/2

Hence the average case efficiency will be expressed as O (n).

4.4 Algorithm for Searching Techniques

4.4.1 Linear Search

Linear search also called sequential search is a sequential method for finding a particular value in a list. This method checks the search element with each element in sequence until the desired element is found or the list is exhausted. In this searching algorithm, list need not be ordered.

Pseudo code

1. Traverse the array using for loop
2. In every iteration, compare the target search key value with the current value of the list.

- If the values match, display the current index and value of the array
- If the values do not match, move on to the next array element.

3. If no match is found, display the search element not found.

To search the number 25 in the array given below, linear search will go step by step in a sequential order starting from the first element in the given array if the search element is found that index is returned otherwise the search is continued till the last index of the array. In this example number 25 is found at index number 3.

index	0	1	2	3	4
values	10	12	20	25	30

Example 1:

Input: values[] = {5, 34, 65, 12, 77, 35}
target = 77
Output: 4

Example 2:

Input: values[] = {101, 392, 1, 54, 32, 22, 90, 93}
target = 200
Output: -1 (not found)

4.4.2. Binary Search

Binary search also called half-interval search algorithm. It finds the position of a search element within a sorted array. The binary search algorithm can be done as divide-and-conquer search algorithm and executes in logarithmic time.

Pseudo code for Binary search

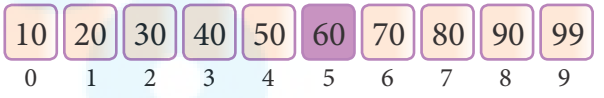
1. Start with the middle element:

- If the search element is equal to the middle element of the array i.e., the middle value = number of elements in array/2, then return the index of the middle element.
- If not, then compare the middle element with the search value,
- If the search element is greater than the number in the middle index, then select the elements to the right side of the middle index, and go to Step-1.
- If the search element is less than the number in the middle index, then select the elements to the left side of the middle index, and start with Step-1.

2. When a match is found, display success message with the index of the element matched.
3. If no match is found for all comparisons, then display unsuccessful message.

Binary Search Working principles

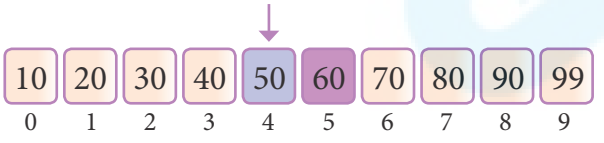
List of elements in an array must be sorted first for Binary search. The following example describes the step by step operation of binary search. Consider the following array of elemnts, the array is being sorted so it enables to do the binary search algorithm. Let us assume that the search element is 60 and we need to search the location or index of search element 60 using binary search.



First, we find index of middle element of the array by using this formula :

mid = low + (high - low) / 2

Here it is, 0 + (9 - 0) / 2 = 4 (fractional part ignored). So, 4 is the mid value of the array.



Now compare the search element with the value stored at mid value location 4. The value stored at location or index 4 is 50, which is not match with search element. As the search value 60 is greater than 50.

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

Now we change our low to mid + 1 and find the new mid value again using the formula.

low = mid + 1

mid = low + (high - low) / 2

Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

The value stored at location or index 7 is not a match with search element, rather it is more than what we are looking for. So, the search element must be in the lower part from the current mid value location

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

The search element still not found. Hence, we calculated the mid again by using the formula.

high = mid -1

mid = low + (high - low)/2

Now the mid value is 5.

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

Now we compare the value stored at location 5 with our search element. We found that it is a match.

10	20	30	40	50	60	70	80	90	99
0	1	2	3	4	5	6	7	8	9

We can conclude that the search element 60 is found at lcoation or index 5. For example if we take the search element as 95, For this value this binary search algorithm return unsuccessful result.

4.5 Sorting Techniques

4.5.1 Bubble sort algorithm

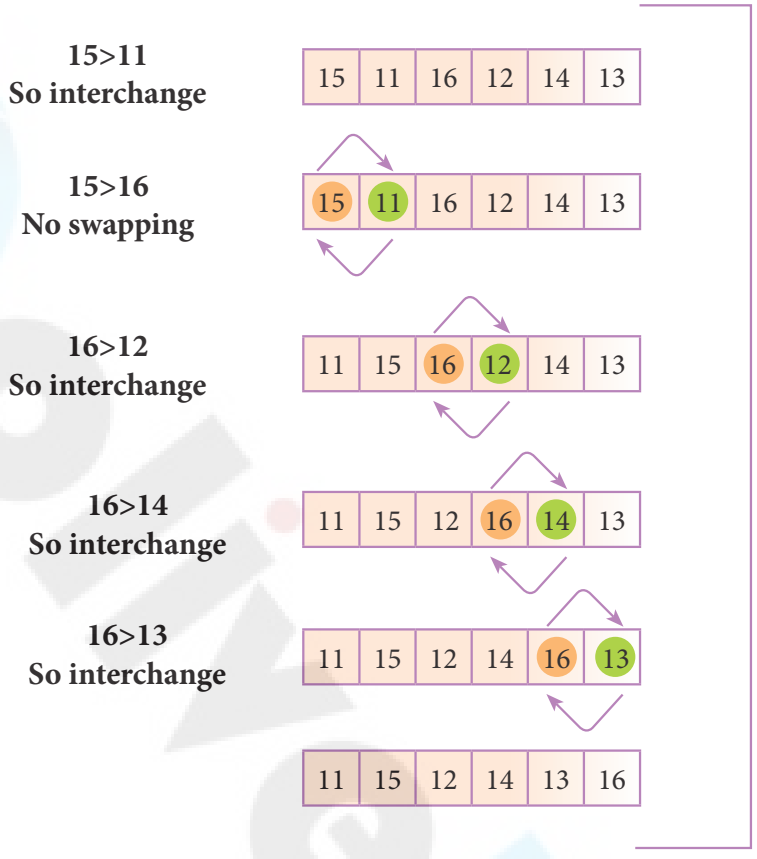
Bubble sort is a simple sorting algorithm. The algorithm starts at the beginning of the list of values stored in an array. It compares each pair of adjacent elements and swaps them if they are in the unsorted order. This comparison and passed to be continued until no swaps are needed, which indicates that the list of values stored in an array is sorted. The algorithm is a comparison sort, is named for the way smaller elements "bubble" to the top of the list. Although the algorithm is simple, it is too slow and less efficient when compared to insertion sort and other sorting methods.

Assume list is an array of n elements. The swap function swaps the values of the given array elements.

Pseudo code

1. Start with the first element i.e., index = 0, compare the current element with the next element of the array.
2. If the current element is greater than the next element of the array, swap them.
3. If the current element is less than the next or right side of the element, move to the next element. Go to Step 1 and repeat until end of the index is reached.

Let's consider an array with values {15, 11, 16, 12, 14, 13} Below, we have a pictorial representation of how bubble sort will sort the given array.



The above pictorial example is for iteration-1. Similarly, remaining iteration can be done. The final iteration will give the sorted array.

At the end of all the iterations we will get the sorted values in an array as given below:

11	12	13	14	15	16

4.5.2 Selection sort

The selection sort is a simple sorting algorithm that improves on the performance of bubble sort by making only one exchange for every pass through the list. This algorithm will first find the smallest elements in array and swap it with the element in the first position of an array, then it will find the second smallest element and swap that element with the element in the second position, and it will continue until the entire array is sorted in respective order.

This algorithm repeatedly selects the next-smallest element and swaps in into the right place for every pass. Hence it is called selection sort.

Pseudo code

1. Start from the first element i.e., index-0, we search the smallest element in the array, and replace it with the element in the first position.

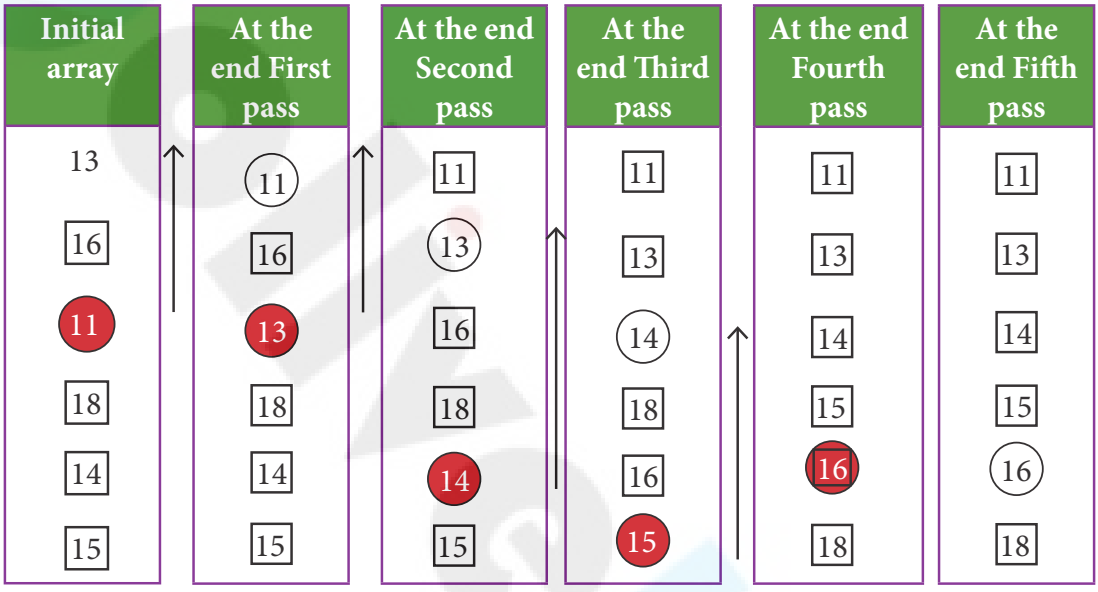
2. Now we move on to the second element position, and look for smallest element present in the sub-array, from starting index to till the last index of sub - array.

3. Now replace the second smallest identified in step-2 at the second position in the or original array, or also called first position in the sub array.

4. This is repeated, until the array is completely sorted.

Let's consider an array with values {13, 16, 11, 18, 14, 15}

Below, we have a pictorial representation of how selection sort will sort the given array.



In the first pass, the smallest element will be 11, so it will be placed at the first position.

After that, next smallest element will be searched from an array. Now we will get 13 as the smallest, so it will be then placed at the second position.

Then leaving the first element, next smallest element will be searched, from the remaining elements. We will get 13 as the smallest, so it will be then placed at the second position.

Then leaving 11 and 13 because they are at the correct position, we will search for the next smallest element from the rest of the elements and put it at third position and keep doing this until array is sorted.

Finally we will get the sorted array end of the pass as shown above diagram.

4.5.3 Insertion sort

Insertion sort is a simple sorting algorithm. It works by taking elements from the list one by one and inserting then in their correct position in to a new sorted list. This algorithm builds the final sorted array at the end. This algorithm uses n-1 number of passes to get the final sorted list as per the pervious algorithm as we have discussed.

Pseudo for Insertion sort

Step 1 – If it is the first element, it is already sorted.

Step 2 – Pick next element

Step 3 – Compare with all elements in the sorted sub-list

Step 4 – Shift all the elements in the sorted sub-list that is greater than the value to be sorted

Step 5 – Insert the value

Step 6 – Repeat until list is sorted

44	16	83	07	67	21	34	45	10	Assume 44 is a sorted list of 1 item
16	44	83	07	67	21	34	45	10	inserted 16
16	44	83	07	67	21	34	45	10	inserted 83
07	16	44	83	67	21	34	45	10	inserted 07
07	16	44	67	83	21	34	45	10	inserted 67
07	16	21	44	67	83	34	45	10	inserted 21
07	16	21	34	44	67	83	45	10	inserted 34
07	16	21	34	44	45	67	83	10	inserted 45
07	10	16	21	34	44	45	67	83	inserted 10

At the end of the pass the insertion sort algorithm gives the sorted output in ascending order as shown below:

07	10	16	21	34	44	45	67	83

4.6. Dynamic programming

Dynamic programming is an algorithmic design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions. Dynamic programming approach is similar to divide and conquer. The given problem is divided into smaller and yet smaller possible sub-problems.

Dynamic programming is used whenever problems can be divided into similar sub-problems. so that their results can be re-used to complete the process. Dynamic programming approaches are used to find the solution in optimized way. For every inner sub problem, dynamic algorithm will try to check the results of the previously solved sub-problems. The solutions of overlapped sub-problems are combined in order to get the better solution.

Steps to do Dynamic programming

- The given problem will be divided into smaller overlapping sub-problems.
- An optimum solution for the given problem can be achieved by using result of smaller sub-problem.
- Dynamic algorithms uses Memoization.

Note

Memoization or memoisation is an optimization technique used primarily to speed up computer programs by storing the results of expensive function calls and returning the cached result when the same inputs occur again.

4.6.1 Fibonacci Series – An example

Fibonacci series generates the subsequent number by adding two previous numbers. Fibonacci series starts from two numbers – Fib 0 & Fib 1. The initial values of Fib 0 & Fib 1 can be taken as 0 and 1.

Fibonacci series satisfies the following conditions :

$Fibn = Fib_{n-1} + Fib_{n-2}$

Hence, a Fibonacci series for the n value 8 can look like this

Fib_8 = 0 1 1 2 3 5 8 13

4.6.2 Fibonacci Iterative Algorithm with Dynamic programming approach

The following example shows a simple Dynamic programming approach for the generation of Fibonacci series.

Initialize f0=0, f1 =1

step-1: Print the initial values of Fibonacci f0 and f1

step-2: Calculate fibanocci fib ← f0 + f1

step-3: Assign f0← f1, f1← fib

step-4: Print the next consecutive value of fibanocci fib

step-5: Goto step-2 and repeat until the specified number of terms generated

For example if we generate fibobnacci series upto 10 digits, the algorithm will generate the series as shown below:

The Fibonacci series is : 0 1 1 2 3 5 8 13 21 34 55

Points to remember:

- An algorithm is a finite set of instructions to accomplish a particular task.
- Algorithm consists of step-step-by instructions that are required to accomplish a task and helps the programmer to develop the program.
- Program is an expression of algorithm in a programming language.
- Algorithm analysis deals with the execution or running time of various operations involved.
- Space complexity of an algorithm is the amount of memory required to run to its completion.
- Big Oh is often used to describe the worst-case of an algorithm.
- Big Omega is used to describe the lower bound which is best way to solve the space complexity.
- The Time complexity of an algorithm is given by the number of steps taken by the algorithm to complete the process.
- The efficiency of an algorithm is defined as the number of computational resources used by the algorithm.

Points to remember:

- A way of designing algorithm is called algorithmic strategy.
- A space-time or time-memory tradeoff is a way of solving a problem or calculation in less time by using more storage space.
- Asymptotic Notations are languages that uses meaningful statements about time and space complexity.
- Bubble sort is a simple sorting algorithm. It compares each pair of adjacent items and swaps them if they are in the unsorted order.
- The selection sort improves on the bubble sort by making only one exchange for every pass through the list.
- Insertion sort is a simple sorting algorithm that builds the final sorted array or list one item at a time. It always maintains a sorted sublist in the lower positions of the list.
- Dynamic programming is used when the solutions to a problem can be viewed as the result of a sequence of decisions.

Evaluation

9DJFRN

Part - I

Choose the best answer: **(1 Marks)**

1. The word comes from the name of a Persian mathematician Abu Ja'far Mohammed ibn-i Musa al Khowarizmi is called?

   (A) Flowchart (B) Flow (C) Algorithm (D) Syntax

2. From the following sorting algorithms which algorithm needs the minimum number of swaps?

   (A) Bubble sort (B) Quick sort (C) Merge sort (D) Selection sort

3. Two main measures for the efficiency of an algorithm are

   (A) Processor and memory (B) Complexity and capacity

   (C) Time and space (D) Data and space

4. The complexity of linear search algorithm is

   (A) O(n) (B) O(log n) (C) O(n2) (D) O(n log n)

5. From the following sorting algorithms which has the lowest worst case complexity?

   (A) Bubble sort (B) Quick sort (C) Merge sort (D) Selection sort

6. Which of the following is not a stable sorting algorithm?

   (A) Insertion sort (B) Selection sort (C) Bubble sort (D) Merge sort

7. Time complexity of bubble sort in best case is

   (A) θ (n) (B) θ (nlogn) (C) θ (n2) (D) θ (n(logn) 2)

8. The Θ notation in asymptotic evaluation represents

   (A) Base case (B) Average case (C) Worst case (D) NULL case

9. If a problem can be broken into subproblems which are reused several times, the problem possesses which property?

   (A) Overlapping subproblems (B) Optimal substructure

   (C) Memoization (D) Greedy

10. In dynamic programming, the technique of storing the previously calculated values is called ?

   (A) Saving value property (B) Storing value property

   (C) Memoization (D) Mapping

Part - II

Answer the following questions **(2 Marks)**

1. What is an Algorithm?
2. Define Pseudo code.
3. Who is an Algorist?
4. What is Sorting?
5. What is searching? Write its types.

Part - III

Answer the following questions **(3 Marks)**

1. List the characteristics of an algorithm.
2. Discuss about Algorithmic complexity and its types.
3. What are the factors that influence time and space complexity.
4. Write a note on Asymptotic notation.
5. What do you understand by Dynamic programming?

Part - IV

Answer the following questions **(5Marks)**

1. Explain the characteristics of an algorithm.
2. Discuss about Linear search algorithm.
3. What is Binary search? Discuss with example.
4. Explain the Bubble sort algorithm with example.
5. Explain the concept of Dynamic programming with suitable example.

Reference Books

1. *Fundamentals Computer Algorithms, Ellis Horowitz, Sartaj Sahni, Sanguthevar, Rajasekaran, Second Edition, University press (India) Limited, 2013.*
2. *Design and Analysis of Algorithms, S. Sridhar, Oxford University Press, 2015*

Web References

www.wickipedia.org

CASE STUDY/ STUDENT'S ACTIVITY

1. Create an algorithm for grading systems of your class student's Quarterly examination marks by satisfying all necessary conditions.

Why Python in standard XII curriculum?

This is the question that is fretting the minds of teachers and students.

The present book is organized in such a way that even a novice reader can grasp and work on python programming.

Testimonies

- "Python has been an important part of Google since the beginning and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language." -- **Peter Norvig, director of search quality at Google, Inc.**
- "Python is fast enough for our site and allows us to produce maintainable features in record times, with a minimum of developers," -- **Cuong Do, Software Architect, YouTube.com**

Python's popularity has seen a steady and unflagging growth over the recent years. Today, familiarity with Python is an advantage for every programmer, as Python has infiltrated every niche and has useful roles to play in any software solution.

Python has experienced an impressive growth as compared to the other languages. The **IEEE Spectrum magazine** published by **the Institute of Electrical & Electronics Engineers, New York**, ranks Python as the top language for 2018, for the second consecutive year.

The above statistical data has maintained its grip with Python scoring 100 and C++ language stands second nipping at its heels with a 99.7 score.

Python being popular is used by a number of tech giants like **Google, Instagram, Pinterest, Yahoo, Disney, IBM, Nokia etc.**

1. Python	100.0
2. C++	99.7
3. Java	97.5
4. C	96.7
5. C#	89.4
6. PHP	84.9
7. R	82.8
8. JavaScript	82.6
9. Go	76.4
10. Assembly	74.1

Many businesses are advised to choose Python for the following reasons:-

- Easy syntax and readability
- High level scripting language with oops
- famous for enormous functions, add-on modules, libraries, frameworks and tool-kits.
- Built-in functions supports scientific computing.

With the advent of computers, there have been significant changes in the way we work in almost all the fields. The computerization has helped to improve productivity and accelerate decision making in every organization. Even for individuals, be it engineers, doctors, chartered accountants or homemakers, the style of working has changed drastically. So as we go, let us accept the change and move towards a brighter day ahead.

Unit II

CHAPTER 5

PYTHON – VARIABLES AND OPERATORS

7972LP

Learning Objectives

After studying this lesson, students will be able to:

- Appreciate the use of Graphical User Interface (GUI) and Integrated Development Environment (IDE) for creating Python programs.
- Work in Interactive & Script mode for programming.
- Create and assign values to variables.
- Understand the concept and usage of different data types in Python.
- Appreciate the importance and usage of different types of operators (Arithmetic, Relational and Logical)
- Creating Python expression (s) and statement (s).

5.1 Introduction

Python is a general purpose programming language created by Guido Van Rossum from CWI (Centrum Wiskunde & Informatica) which is a National Research Institute for Mathematics and Computer Science in Netherlands. The language was released in I991. Python got its name from a BBC comedy series from seventies- "Monty Python's Flying Circus". Python supports both Procedural and Object Oriented programming approaches.

5.2 Key features of Python

- ✓ It is a general purpose programming language which can be used for both scientific and non-scientific programming.
- ✓ It is a platform independent programming language.
- ✓ The programs written in Python are easily readable and understandable.

The version 3.x of Python **IDLE** (**I**ntegrated **D**evelopment **L**earning **E**nvironment) is used to develop and run Python code. It can be downloaded from the web resource **www.python.org.**

5.3 Programming in Python

In Python, programs can be written in two ways namely **Interactive mode** and **Script mode.** The Interactive mode allows us to write codes in Python command prompt (>>>) whereas in script mode programs can be written and stored as separate file with the extension **.py** and executed. Script mode is used to create and edit python source file.

5.3.1 Interactive mode Programming

In interactive mode Python code can be directly typed and the interpreter displays the result(s) immediately. The interactive mode can also be used as a **simple calculator.**

(i) Invoking Python IDLE

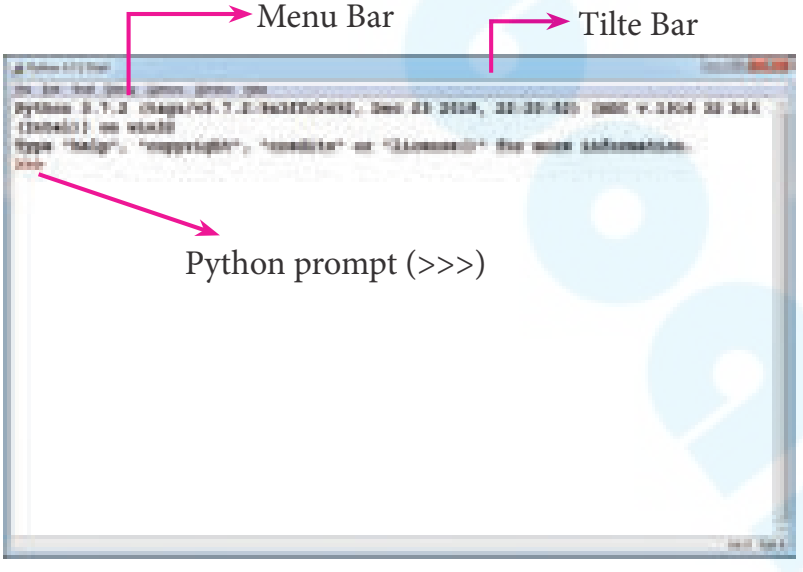
The following command can be used to invoke Python IDLE from Window OS.

Start → All Programs → Python 3.x → IDLE (Python 3.x)

(Or)

Click python Icon on the Desktop if available.

Now **Python IDLE** window appears as shown in the **Figure 5.1**



Python IDLE Window

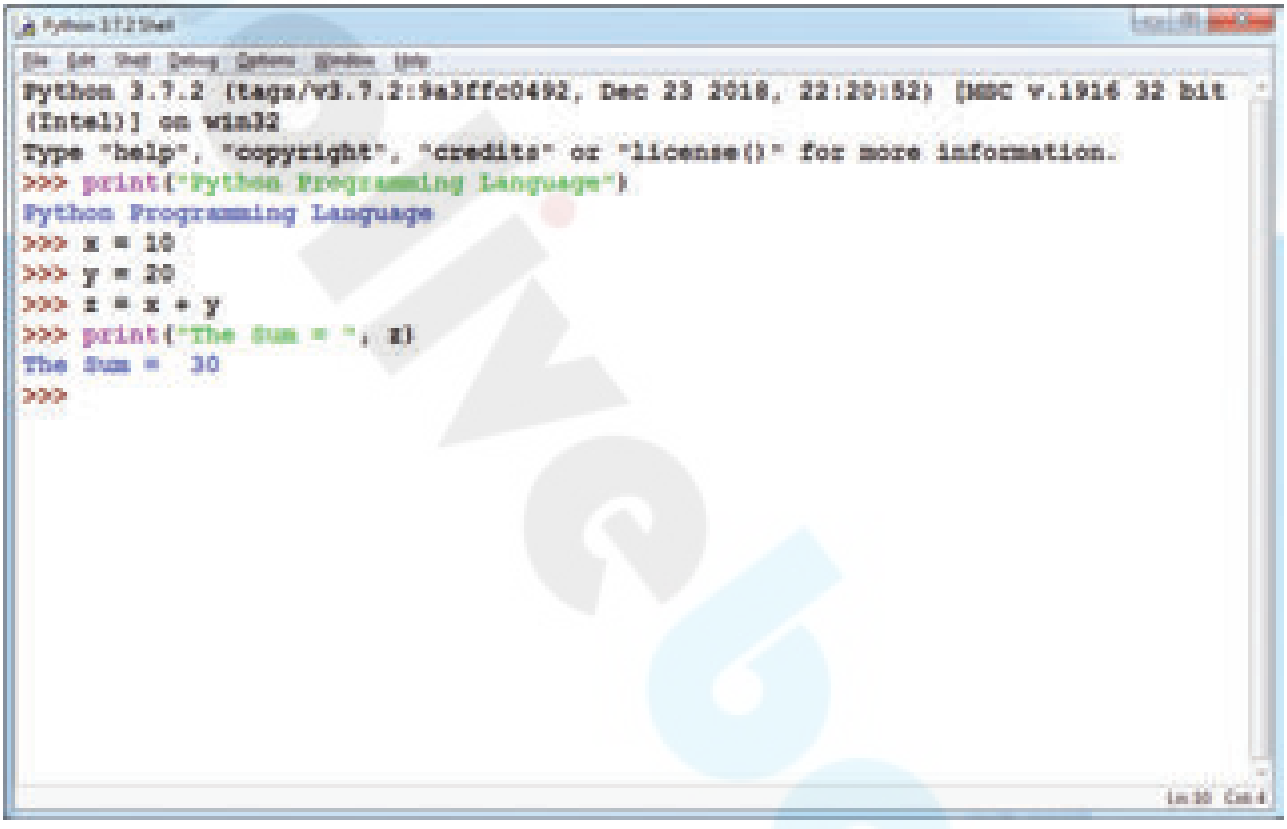
The prompt (>>>) indicates that Interpreter is ready to accept instructions. Therefore, the prompt on screen means **IDLE** is working in interactive mode. Now let us try as a simple calculator by using a simple mathematical expressions.

Example 1:

```
>>> 5 + 10
15
>>> 5 + 50 *10
505
>>> 5 ** 2
25
```

Example 2:

```
>>>print ("Python Programming Language")
Python Programming Language
>>>x=10
>>>y=20
>>>z=x + y
>>>print ("The Sum", z)
The Sum = 30
```



Python Interactive Window

5.3.2 Script mode Programming

Basically, a script is a text file containing the Python statements. Python Scripts are reusable code. Once the script is created, it can be executed again and again without retyping. The Scripts are editable.

(i) Creating Scripts in Python

1. Choose **File** → **New File** or press **Ctrl + N** in Python shell window.

A8C9SN

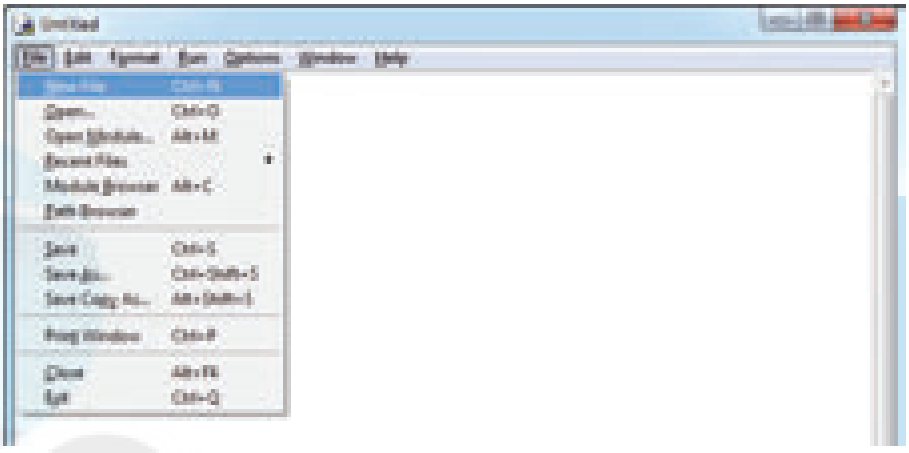


Figure 5.3 – To create new File

2. An **untitled** blank script text editor will be displayed on screen as shown in **Figure 5.3(a)**

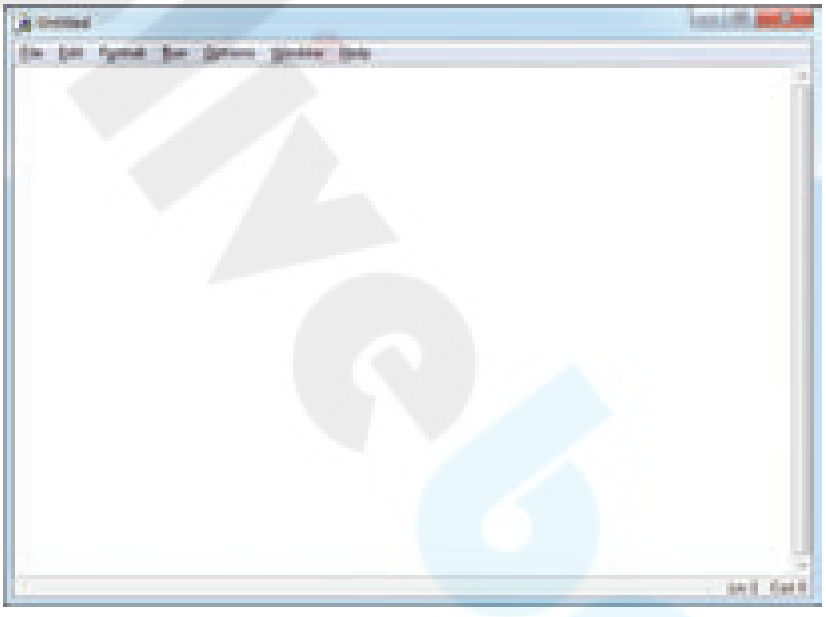


Figure 5.3(a) Untitled, blank Python script editor

3. Type the following code in Script editor

```
a =100

b = 350

c = a+b

print ("The Sum=", c)
```

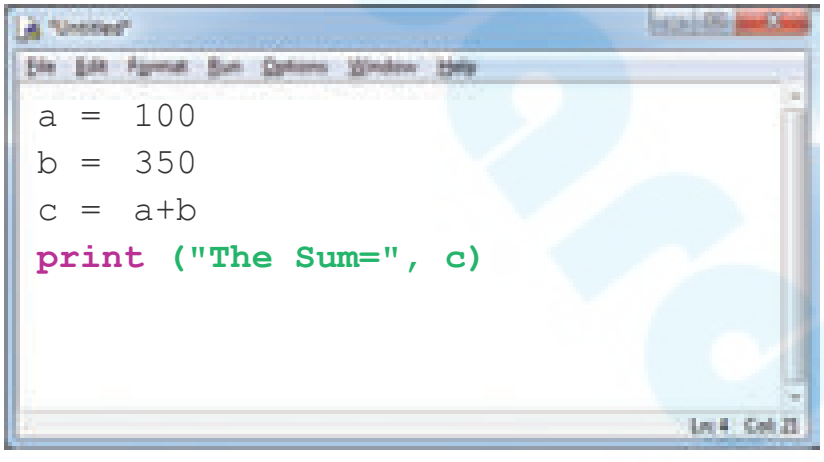


Figure 5.4 – Python Sample code

(ii) Saving Python Script

(1) Choose **File** → **Save** or Press **Ctrl + S**

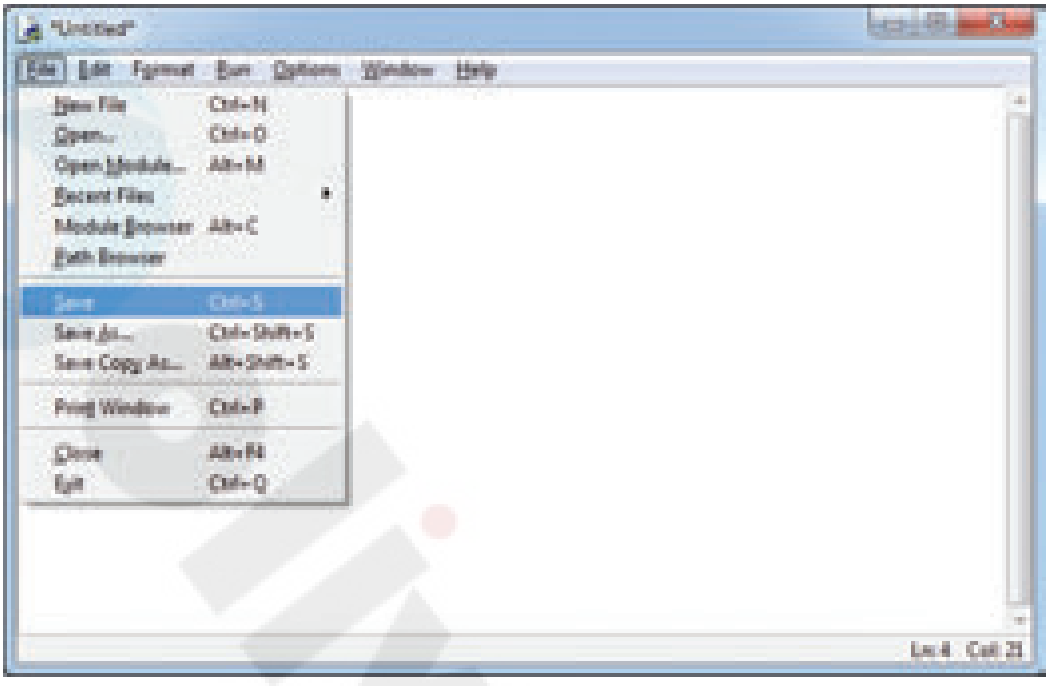


Figure 5.5 – To Save the file First time

(2) Now, **Save As** dialog box appears on the screen as shown in the **Figure 5.6**

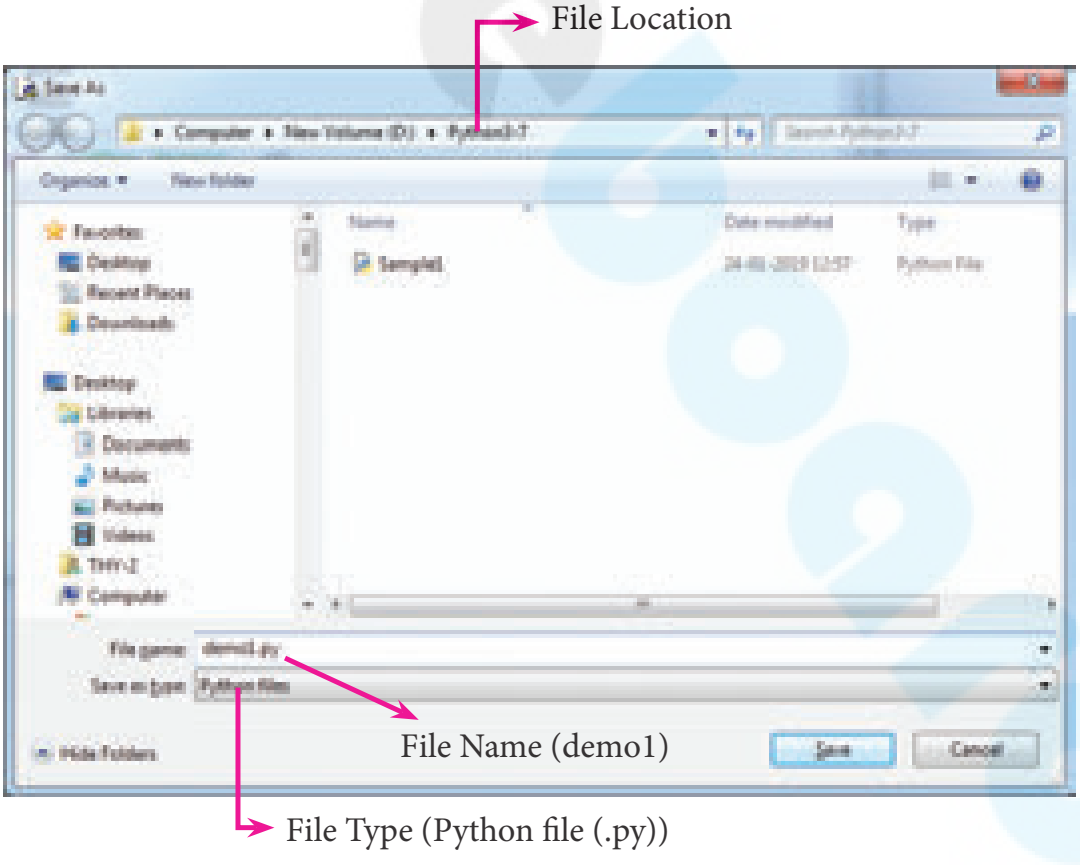


Figure 5.6 – Save As Dialog Box

(3) In the **Save As** dialog box, select the location where you want to save your Python code, and type the file name in **File Name** box. Python files are by default saved with extension **.py.** Thus, while creating Python scripts using Python Script editor, no need to specify the file extension.

(4) Finally, click **Save** button to save your Python script.

(iii) Executing Python Script

(1) Choose **Run** → **Run Module** or Press **F5**

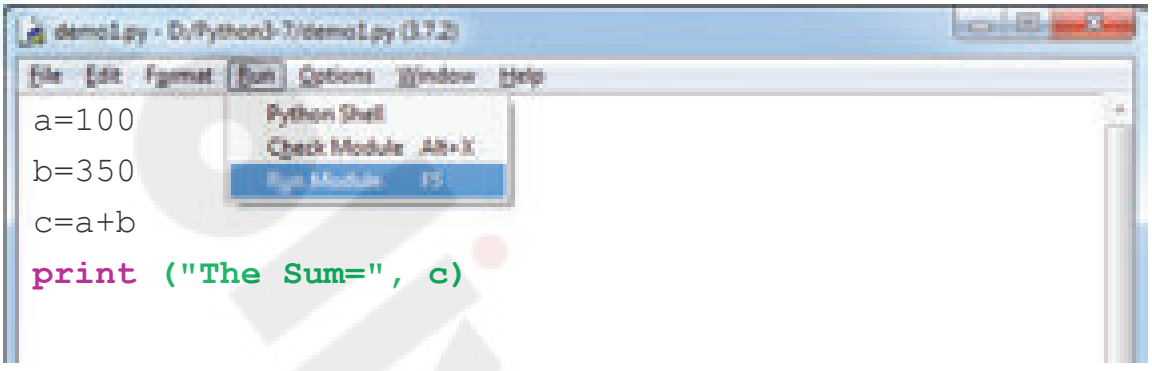


Figure 5.7 – To Execute Python Script

(2) If your code has any error, it will be shown in red color in the IDLE window, and Python describes the type of error occurred. To correct the errors, go back to Script editor, make corrections, save the file using **Ctrl + S** or **File** → **Save** and execute it again.

(3) For all error free code, the output will appear in the IDLE window of Python as shown in **Figure 5.8**

Figure 5.8 –Python Script Output Window

5.4 Input and Output Functions

A program needs to interact with the user to accomplish the desired task; this can be achieved using **Input-Output functions**. The **input()** function helps to enter data at run time by the user and the output function **print()** is used to display the result of the program on the screen after execution.

5.4.1 The print() function

In Python, the **print()** function is used to display result on the screen. The syntax for **print()** is as follows:

Example

```
print ("string to be displayed as output " )
print (variable )
print ("String to be displayed as output ", variable)
print ("String1 ", variable, "String 2", variable, "String 3" ……)
```

Example

```
>>> print ("Welcome to Python Programming")
        Welcome to Python Programming
>>> x = 5
>>> y = 6
>>> z = x + y
>>> print (z)
        11
>>> print ("The sum = ", z)
        The sum = 11
>>> print ("The sum of ", x, " and ", y, " is ", z)
        The sum of 5 and 6 is 11
```

The **print ( )** evaluates the expression before printing it on the monitor. The print () displays an entire statement which is specified within print ( ). **Comma (,)** is used as a separator in **print ( )** to print more than one item.

5.4.2 input() function

In Python, **input( )** function is used to accept data as input at run time. The syntax for **input()** function is,

```
Variable = input ("prompt string")
```

Where, **prompt string** in the syntax is a statement or message to the user, to know what input can be given.

If a prompt string is used, it is displayed on the monitor; the user can provide expected data from the input device. The **input( )** takes whatever is typed from the keyboard and stores the entered data in the given variable. If prompt string is not given in **input( )** no message is displayed on the screen, thus, the user will not know what is to be typed as input.

Example 1:input( ) with prompt string

```
>>> city=input ("Enter Your City: ")
        Enter Your City: Madurai
>>> print ("I am from ", city)
        I am from Madurai
```

Example 2:input( ) without prompt string

```
>>> city=input()
        Rajarajan
>>> print ("I am from", city)
        I am from Rajarajan
```

Note that in example-2, the **input( )** is not having any prompt string, thus the user will not know what is to be typed as input. If the user inputs irrelevant data as given in the above example, then the output will be unexpected. So, to make your program more interactive, provide prompt string with **input( ).**

The **input ( )** accepts all data as string or characters but not as numbers. If a numerical value is entered, the input values should be explicitly converted into numeric data type. The **int( )** function is used to convert string data as integer data explicitly. We will learn about more such functions in later chapters.

Example 3:

```
x = int (input("Enter Number 1: "))
y = int (input("Enter Number 2: "))
print ("The sum = ", x+y)
```

Output:

```
        Enter Number 1: 34
        Enter Number 2: 56
        The sum = 90
```

Example 4: Alternate method for the above program

```
x,y=int (input("Enter Number 1 :")),int(input("Enter Number 2:"))
print ("X = ",x," Y = ",y)
```

Output:

```
Enter Number 1 :30
Enter Number 2:50
X =  30  Y =  50
```

5.5 Comments in Python

In Python, comments begin with hash symbol (#). The lines that begins with # are considered as comments and ignored by the Python interpreter. Comments may be single line or no multi-lines. The multiline comments should be enclosed within a set of ''' '''(triple quotes) as given below.

 # It is Single line Comment

''' It is multiline comment

which contains more than one line '''

5.6 Indentation

Python uses whitespace such as **spaces** and **tabs** to define program blocks whereas other languages like C, C++, java use curly braces { } to indicate blocks of codes for class, functions or body of the loops and block of selection command. The number of whitespaces (spaces and tabs) in the indentation is not fixed, but all statements within the block must be indented with same amount spaces.

5.7 Tokens

Python breaks each logical line into a sequence of elementary lexical components known as **Tokens**. The normal token types are

1) Identifiers,
2) Keywords,
3) Operators,
4) Delimiters and
5) Literals.

Whitespace separation is necessary between tokens, identifiers or keywords.

5.7.1. Identifiers

An Identifier is a name used to identify a variable, function, class, module or object.

- An identifier must start with an alphabet (A..Z or a..z) or underscore (_).
- Identifiers may contain digits (0 .. 9)
- Python identifiers are case sensitive i.e. uppercase and lowercase letters are distinct.
- Identifiers must not be a **python** keyword.
- Python does not allow punctuation character such as %,$, @ etc., within identifiers.

Example of valid identifiers

Sum, total_marks, regno, num1

Example of invalid identifiers

12Name, name$, total-mark, continue

5.7.2. Keywords

Keywords are special words used by Python interpreter to recognize the structure of program. As these words have specific meaning for interpreter, they cannot be used for any other purpose.

Table 5.1 Python's Keywords

False	class	finally	is	return
none	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	None
break	except	in	raise	

5.7.3 Operators

In computer programming languages operators are special symbols which represent computations, conditional matching etc. The value of an operator used is called **operands**. Operators are categorized as Arithmetic, Relational, Logical, Assignment etc. Value and variables when used with operator are known as **operands**.

(i) Arithmetic operators

An arithmetic operator is a mathematical operator that takes two operands and performs a calculation on them. They are used for simple arithmetic. Most computer languages contain a set of such operators that can be used within equations to perform different types of sequential calculations.

Python supports the following Arithmetic operators.

Operator - Operation	Examples	Result
Assume a=100 and b=10. Evaluate the following expressions		
+ (Addition)	>>> a + b	110
- (Subtraction)	>>>a – b	90
* (Multiplication)	>>> a*b	1000
/ (Divisioin)	>>> a / b	10.0
% (Modulus)	>>> a % 30	10
** (Exponent)	>>> a ** 2	10000
// (Floor Division)	>>> a//30 (Integer Division)	3

Program 5.1 To test Arithmetic Operators:

```
#Demo Program to test Arithmetic Operators
        a=100
        b=10
        print ("The Sum        = ",a+b)
        print ("The Difference = ",a-b)
        print ("The Product    = ",a*b)
        print ("The Quotient   = ",a/b)
        print ("The Remainder  = ",a%30)
        print ("The Exponent   = ",a**2)
        print ("The Floor Division =",a//30)
#Program End
```

Output:

```
The Sum            = 110
The Difference     = 90
The Product        = 1000
The Quotient       = 10.0
The Remainder      = 10
The Exponent       = 10000
The Floor Division = 3
```

(ii) Relational or Comparative operators

A Relational operator is also called as **Comparative** operator which checks the relationship between two operands. If the relation is true, it returns **True**; otherwise it returns **False**.

Python supports following relational operators

Operator - Operation	Examples	Result
Assume the value of a=100 and b=35. Evaluate the following expressions.		
== (is Equal)	>>> a==b	False
> (Greater than)	>>> a > b	True
< (Less than)	>>> a < b	False
>= (Greater than or Equal to)	>>> a >= b	True
<= (Less than or Equal to)	>>> a <= b	False
!= (Not equal to)	>>> a != b	True

Coding 5.2 To test Relational Operators:

```
#Demo Program to test Relational Operators
        a=int (input("Enter a Value for A:"))
        b=int (input("Enter a Value for B:"))
        print ("A = ",a," and B = ",b)
        print ("The a==b = ",a==b)
        print ("The a > b = ",a>b)
        print ("The a < b = ",a<b)
        print ("The a >= b = ",a>=b)
        print ("The a <= b = ",a<=0)
        print ("The a != b = ",a!=b)
#Program End
```

Output:

```
Enter a Value for A:35
Enter a Value for B:56
A =  35  and B        = 56
The a==b              = False
The a > b             = False
The a < b             = True
The a >= b            = False
The a <= b            = False
The a != b            = True
```

(iii) Logical operators

In python, Logical operators are used to perform logical operations on the given relational expressions. There are three logical operators they are **and, or** and **not**.

Operator	Example	Result
Assume a = 97 and b = 35, Evaluate the following Logical expressions		
or	>>> a>b or a==b	True
and	>>> a>b and a==b	False
not	>>> not a>b	False i.e. Not True

Program 5.3 To test Logical Operators:

Example – Code

```
#Demo Program to test Logical Operators
a=int (input("Enter a Value for A:"))
b=int (input("Enter a Value for B:"))
print ("A = ",a, " and b = ",b)
print ("The a > b or a == b  = ",a>b or a==b)
print ("The a > b and a == b = ",a>b and a==b)
print ("The not a > b       = ",not a>b)
#Program End
```

Example - Result

```
Enter a Value for A:50
Enter a Value for B:40
A =  50  and b =  40
The a > b or a == b  =  True
The a > b and a == b =  False
The not a > b       =  False
```

(iv) Assignment operators

In Python, = is a simple assignment operator to assign values to variable. Let **a** = 5 and **b** = 10 assigns the value 5 to **a** and 10 to **b** these two assignment statement can also be given as **a,b=5,10** that assigns the value 5 and 10 on the right to the variables a and b respectively. There are various compound operators in Python like +=, -=, *=, /=, %=, **= and //= are also available.

Operator	Description	Example
Assume x=10		
=	Assigns right side operands to left variable	>>> x=10 >>> b="Computer"
+=	Added and assign back the result to left operand i.e. x=30	>>> x+=20 # x=x+20
-=	Subtracted and assign back the result to left operand i.e. x=25	>>> x-=5 # x=x-5
*=	Multiplied and assign back the result to left operand i.e. x=125	>>> x*=5 # x=x*5
/=	Divided and assign back the result to left operand i.e. x=62.5	>>> x/=2 # x=x/2

%=	Taken modulus(Remainder) using two operands and assign the result to left operand i.e. x=2.5	>>> x%=3 # x=x%3
**=	Performed exponential (power) calculation on operators and assign value to the left operand i.e. x=6.25	>>> x**=2 # x=x**2
//=	Performed floor division on operators and assign value to the left operand i.e. x=2.0	>>> x//=3

Program 5.4 To test Assignment Operators:

Program Coding	Output

#Demo Program to test Assignment Operators
x=int (input("Type a Value for X : "))
print ("X = ",x)
print ("The x is =",x)
x+=20
print ("The x += 20 is =",x)
x-=5
print ("The x -= 5 is = ",x)
x*=5
print ("The x *= 5 is = ",x)
x/=2
print ("The x /= 2 is = ",x)
x%=3
print ("The x %= 3 is = ",x)
x**=2
print ("The x **= 2 is = ",x)
x//=3
print ("The x //= 3 is = ",x)
#Program End
	Type a Value for X : 10
	X = 10
	The x is = 10
	The x += 20 is = 30
	The x -= 5 is = 25
	The x *= 5 is = 125
	The x /= 2 is = 62.5
	The x %= 3 is = 2.5
	The x **= 2 is = 6.25
	The x //= 3 is = 2.0

(v) Conditional operator

Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false. It simply allows testing a condition in a single line replacing the multiline if-else making the code compact.

The Syntax conditional operator is,

Variable Name = [on_true] if [Test expression] else [on_false]

Example :

```
min= 50 if 49<50 else 70 # min = 50
min= 50 if 49>50 else 70 # min = 70
```

Program 5.5 To test Conditional (Ternary) Operator:

```
# Program to demonstrate conditional operator
        a, b = 30, 20
# Copy value of a in min if a < b else copy b
        min = a if a < b else b
        print ("The Minimum of A and B is ",min)
# End of the Program
```

Output:

```
        The Minimum of A and B is  20
```

5.7.4 Delimiters

Python uses the symbols and symbol combinations as delimiters in expressions, lists, dictionaries and strings. Following are the delimiters.

(	)	[	]	{	}
,	:	.	‘	=	;
+=	-=	*=	/=	//=	%=
&=	|=	^=	>>=	<<=	**=

5.7.5 Literals

Literal is a raw data given to a variable or constant. In Python, there are various types of literals.

1) Numeric
2) String
3) Boolean

(i) Numeric Literals

Numeric Literals consists of digits and are immutable (unchangeable). Numeric literals can belong to 3 different numerical types Integer, Float and Complex.

Program 5.6 : To demonstrate Numeric literals

```
# Program to demonstrate Numeric Literals
        a = 0b1010                  #Binary Literals
        b = 100                     #Decimal Literal
        c = 0o310                   #Octal Literal
        d = 0x12c                   #Hexadecimal Literal
        print ("Integer Literals :",a,b,c,d)
        #Float Literal
        float_1 = 10.5
        float_2 = 1.5e2
        print ("Float Literals :",float_1,float_2)
        #Complex Literal
        x = 1 + 3.14 j
        print ("Complex Literals :", x)
        Print ("x = ", x , "Imaginary part of x = ", x.imag, "Real part of x = ", x.real)
#End of the Program
```

Output:

```
Integer Literals : 10 100 200 300
Float Literals : 10.5 150.0
Complex Literals :
x = (1.3.14) Imaginary part of x = 3.14 Real part of x = 1.0
```

(ii) String Literals

In Python a string literal is a sequence of characters surrounded by quotes. Python supports single, double and triple quotes for a string. A character literal is a single character surrounded by single or double quotes. The value with triple-quote ''' ''' is used to give multi-line string literal.

Program 5.7 To test String Literals

```
# Demo Program to test String Literals
        strings = "This is Python"
        char = "C"
        multiline_str = '''This is a multiline string with more than one line code.'''
        print (strings)
        print (char)
        print (multiline_str)
# End of the Program
```

Output:

```
This is Python
C
This is a multiline string with more than one line code.
```

(iii) Boolean Literals

A Boolean literal can have any of the two values: True or False.

Program 5.8 To test Boolean Literals:

```
# Demo Program to test String Literals
        boolean_1 = True
        boolean_2 = False
        print ("Demo Program for Boolean Literals")
        print ("Boolean Value1 :",boolean_1)
        print ("Boolean Value2 :",boolean_2)
# End of the Program
```

Output:

```
Demo Program for Boolean Literals
Boolean Value1 : True
Boolean Value2 : False
```

(iv) Escape Sequences

In Python strings, the backslash "\" is a special character, also called the **"escape"** character. It is used in representing certain whitespace characters: **"\t"** is a tab, **"\n"** is a newline, and **"\r"** is a carriage return. For example to print the message "It's raining", the Python command is

>>> print ("It\'s rainning")

It's rainning

Python supports the following escape sequence characters.

Escape sequence character	Description	Example	Output
\\	Backslash	>>> print("\\test")	\test
\'	Single-quote	>>> print("Doesn\'t")	Doesn't
\"	Double-quote	>>> print("\"Python\"")	"Python"
\n	New line	print("Python","\n","Lang..")	Python Lang..
\t	Tab	print("Python","\t","Lang..")	Python Lang..

5.8 Python Data types

All data values in Python are objects and each object or value has type. Python has Built-in or Fundamental data types such as Number, String, Boolean, tuples, lists and dictionaries.

5.8.1 Number Data type

The built-in number objects in Python supports integers, floating point numbers and complex numbers.

Integer Data can be decimal, octal or hexadecimal. Octal integer use digit **0** (Zero) followed by letter '**o**' to denote octal digits and hexadecimal integer use **0X** (Zero and either uppercase or lowercase X) and L (only upper case) to denote long integer.

Example :

```
102, 4567, 567          # Decimal integers
0o102, 0o876, 0o432     # Octal integers
0X102, 0X876, 0X432     # Hexadecimal integers
34L, 523L               # Long decimal integers
```

A floating point data is represented by a sequence of decimal digits that includes a decimal point. An Exponent data contains decimal digit part, decimal point, exponent part followed by one or more digits.

Example :

```
123.34,  456.23,  156.23     # Floating point data
12.E04, 24.e04               # Exponent data
```

Complex number is made up of two floating point values, one each for the real and imaginary parts.

5.8.2 Boolean Data type

A Boolean data can have any of the two values: True or False.

Example :

```
Bool_var1=True
Bool_var2=False
```

5.8.3 String Data type

String data can be enclosed in single quotes or double quotes or triple quotes.

Example :

```
Char_data = 'A'
    String_data= "Computer Science"
    Multiline_data= """String data can be enclosed in single quotes or
    double quotes or triple quotes."""
```

Points to remember:

- Python is a general purpose programming language created by Guido Van Rossum.
- Python shell can be used in two ways, viz., Interactive mode and Script mode.
- Python uses whitespace (spaces and tabs) to define program blocks
- Whitespace separation is necessary between tokens, identifiers or keywords.
- A Program needs to interact with end user to accomplish the desired task, this is done using Input-Output facility.
- Python breaks each logical line into a sequence of elementary lexical components known as Tokens.
- Keywords are special words that are used by Python interpreter to recognize the structure of program.

Evaluation

Part - I

Choose the best answer **(1 Marks)**

1. Who developed Python ?

   A) Ritche B) Guido Van Rossum

   C) Bill Gates D) Sunder Pitchai

2. The Python prompt indicates that Interpreter is ready to accept instruction.

   A) >>> B) <<<

   C) # D) <<

3. Which of the following shortcut is used to create new Python Program ?

   A) Ctrl + C B) Ctrl + F

   C) Ctrl + B D) Ctrl + N

4. Which of the following character is used to give comments in Python Program ?

   A) # B) & C) @ D) $

5. This symbol is used to print more than one item on a single line.

   A) Semicolon(;) B) Dollor($)

   C) comma(,) D) Colon(:)

6. Which of the following is not a token ?

   A) Interpreter B) Identifiers

   C) Keyword D) Operators

AMV4TZ

7. Which of the following is not a Keyword in Python ?

A) break B) while

C) continue D) operators

8. Which operator is also called as Comparative operator?

A) Arithmetic B) Relational

C) Logical D) Assignment

9. Which of the following is not Logical operator?

A) and B) or

C) not D) Assignment

10. Which operator is also called as Conditional operator?

A) Ternary B) Relational

C) Logical D) Assignment

Part - II

Answer the following questions : **(2 Marks)**

1. What are the different modes that can be used to test Python Program ?
2. Write short notes on Tokens.
3. What are the different operators that can be used in Python ?
4. What is a literal? Explain the types of literals ?
5. Write short notes on Exponent data?

Part - III

Answer the following questions : **(3 Marks)**

1. Write short notes on Arithmetic operator with examples.
2. What are the assignment operators that can be used in Python?
3. Explain Ternary operator with examples.
4. Write short notes on Escape sequences with examples.
5. What are string literals? Explain.

Part - IV

Answer the following questions : **(5 Marks)**

1. Describe in detail the procedure Script mode programming.
2. Explain input() and print() functions with examples.
3. Discuss in detail about Tokens in Python

Unit II

CHAPTER 6

CONTROL STRUCTURES

BHMXUZ

Learning Objectives

After studying this lesson, students will be able to:

- To gain knowledge on the various flow of control in Python language.
- To learn through the syntax how to use conditional construct to improve the efficiency of the program flow.
- To apply iteration structures to develop code to repeat the program segment for specific number of times or till the condition is satisfied.

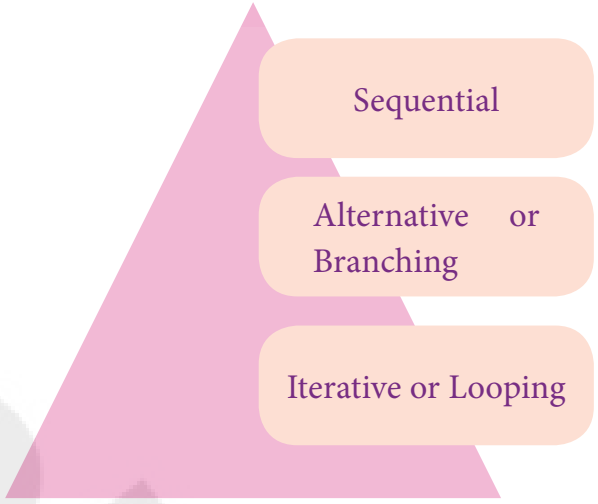
6.1 Introduction

Programs may contain set of statements. These statements are the executable segments that yield the result. In general, statements are executed sequentially, that is the statements are executed one after another. There may be situations in our real life programming where we need to skip a segment or set of statements and execute another segment based on the test of a condition. This is called ***alternative*** or ***branching***. Also, we may need to execute a set of statements multiple times, called ***iteration*** or ***looping***. In this chapter we are to focus on the various control structures in Python, their syntax and learn how to develop the programs using them.

6.2 Control Structures

A program statement that causes a jump of control from one part of the program to another is called **control structure** or **control statement.** As you have already learnt in C++, these control statements are compound statements used to alter the control flow of the process or program depending on the state of the process.

There are three important control structures



6.2.1 Sequential Statement

A **sequential statement** is composed of a sequence of statements which are executed one after another. A code to print your name, address and phone number is an example of sequential statement.

Example 6.1

```
# Program to print your name and address - example for sequential statement
print ("Hello! This is Shyam")
print ("43, Second Lane, North Car Street, TN")
```

Output

```
Hello! This is Shyam
43, Second Lane, North Car Street, TN
```

6.2.2 Alternative or Branching Statement

In our day-to-day life we need to take various decisions and choose an alternate path to achieve our goal. May be we would have taken an alternate route to reach our destination when we find the usual road by which we travel is blocked. This type of decision making is what we are to learn through alternative or branching statement. Checking whether the given number is positive or negative, even or odd can all be done using alternative or branching statement.

Python provides the following types of alternative or branching statements:

- **Simple if statement**
- **if..else statement**
- **if..elif statement**

(i) Simple if statement

Simple if is the simplest of all decision making statements. Condition should be in the form of relational or logical expression.

Syntax:

if <condition>:

statements-block1

In the above syntax if the condition is true statements - block 1 will be executed.

Example 6.2

```
# Program to check the age and print whether eligible for voting
x=int (input("Enter your age :"))
if x>=18:
        print ("You are eligible for voting")
```

Output 1:

```
Enter your age :34
You are eligible for voting
```

Output 2:

```
Enter your age :16
>>>
```

As you can see in the second execution no output will be printed, only the Python prompt will be displayed because the program does not check the alternative process when the condition is failed.

(ii) if..else statement

The **if .. else** statement provides control to check the true block as well as the false block. Following is the syntax of **'if..else'** statement.

Syntax:

if <condition>:

statements-block 1

else:

statements-block 2

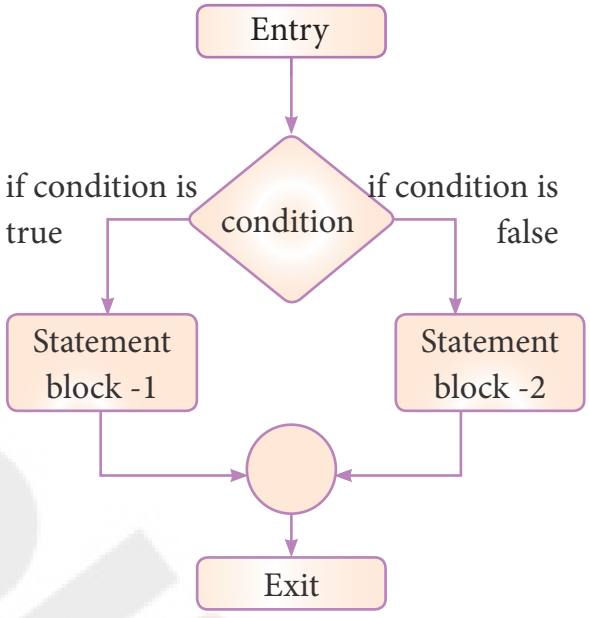


Fig. 6.1 if..else statement execution

if..else statement thus provides two possibilities and the condition determines which BLOCK is to be executed.

Example 6.3: #Program to check if the accepted number odd or even

```
a = int(input("Enter any number :"))
if a%2==0:
        print (a, " is an even number")
else:
        print (a, " is an odd number")
```

Output 1:

```
Enter any number :56
56  is an even number
```

Output 2:

```
Enter any number :67
67  is an odd number
```

An alternate method to rewrite the above program is also available in Python. The complete **if..else** can also written as:

Syntax:

variable = variable1 if condition else variable 2

Note

The condition specified in the if is checked, if it is true, the value of variable1 is stored in variable on the left side of the assignment, otherwise variable2 is taken as the value.

Example 6.4: #Program to check if the accepted number is odd or even (using alternate method of if...else)

```
a = int (input("Enter any number :"))
x="even" if a%2==0 else "odd"
print (a, " is ",x)
```

Output 1:

```
Enter any number :3
3  is  odd
```

Output 2:

```
Enter any number :22
22  is  even
```

(iii) Nested if..elif...else statement:

When we need to construct a chain of **if** statement(s) then **'elif'** clause can be used instead of **'else'.**

Syntax:

if <condition-1>:
    statements-block 1
elif <condition-2>:
    statements-block 2
else:
    statements-block n

In the syntax of **if..elif..else** mentioned above, condition-1 is tested if it is true then statements-block1 is executed, otherwise the control checks condition-2, if it is true statements-block2 is executed and even if it fails statements-block n mentioned in **else** part is executed.

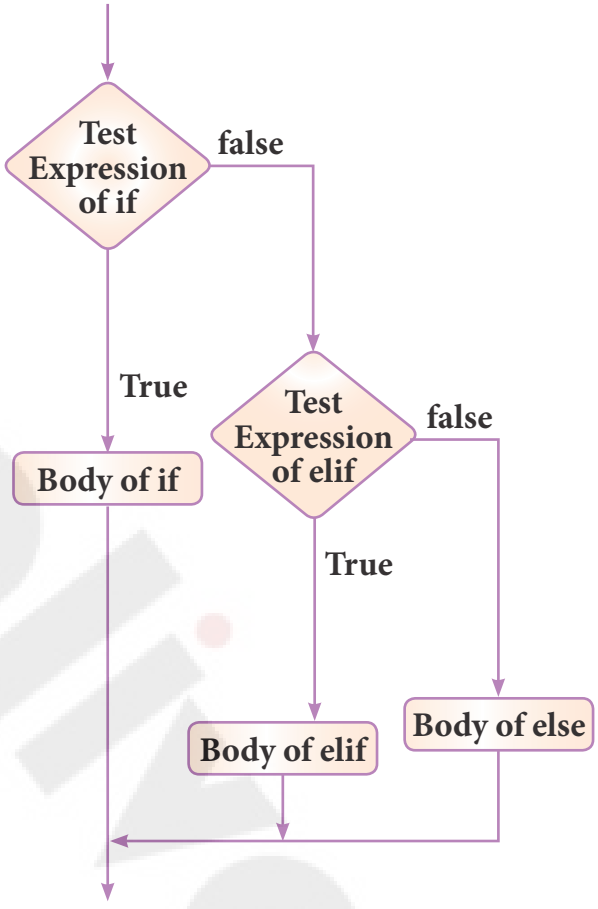


Fig 6.2 if..elif..else statement execution

'elif' clause combines **if..else-if..else** statements to one **if..elif…else. 'elif'** can be considered to be abbreviation of **'else if'.** In an **'if'** statement there is no limit of **'elif'** clause that can be used, but an **'else'** clause if used should be placed at the end.

Note

if..elif..else statement is similar to nested if statement which you have learnt in C++.

Example 6.5: #Program to illustrate the use of nested if statement

Average	***Grade***
>=80 and above	*A*
>=70 and <80	*B*
>=60 and <70	*C*
>=50 and <60	D
Otherwise	E

```
m1=int (input("Enter mark in first subject : "))
m2=int (input("Enter mark in second subject : "))
avg= (m1+m2)/2
if avg>=80:
        print ("Grade : A")
elif avg>=70 and avg<80:
        print ("Grade : B")
elif avg>=60 and avg<70:
        print ("Grade : C")
elif avg>=50 and avg<60:
        print ("Grade : D")
else:
        print ("Grade : E")
```

Output 1:

```
Enter mark in first subject : 34
Enter mark in second subject : 78
Grade : D
```

Output 2 :

```
Enter mark in first subject : 67
Enter mark in second subject : 73
Grade : B
```

Note

The two blocks of code in our example of if-statement are both **indented** four spaces, which is a typical amount of **indentation** for **Python**. In most other programming languages, **indentation** is used only to help make the code look pretty. But in **Python**, it is required to indicate to which block of code the statement belongs to.

Example 6.5a: #Program to illustrate the use of 'in' and 'not in' in if statement

```
ch=input ("Enter a character :")
# to check if the letter is vowel
if ch in ('a', 'A', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U'):
        print (ch,' is a vowel')
# to check if the letter typed is not 'a' or 'b' or 'c'
if ch not in ('a', 'b', 'c'):
        print (ch,'  the letter is not a/b/c')
```

Output 1:

```
Enter a character :e
e is a vowel
```

Output 2:

```
Enter a character :x
x the letter is not a/b/c
```

6.2.3. Iteration or Looping constructs

Iteration or loop are used in situation when the user need to execute a block of code several of times or till the condition is satisfied. A **loop** statement allows to execute a statement or group of statements multiple times.

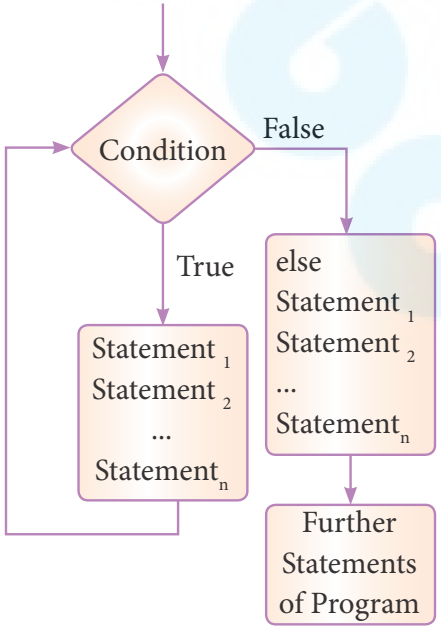


Fig 6.3 Diagram to illustrate how looping construct gets executed

Python provides two types of looping constructs:

- while loop
- for loop

(i) while loop

The syntax of while loop in Python has the following syntax:

Syntax:

while <condition>:
    statements block 1
[else:
    statements block2]

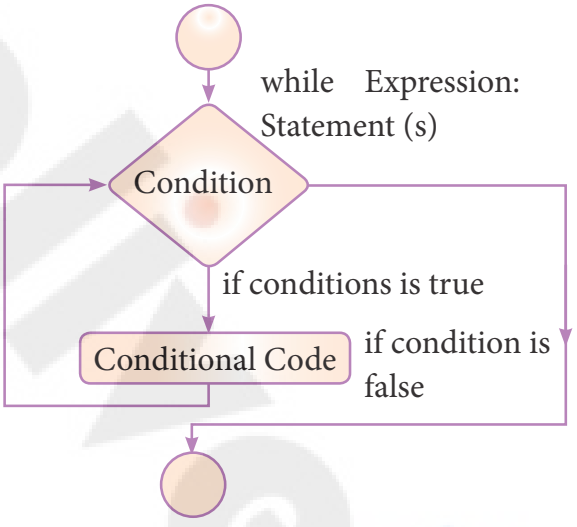


Fig 6.4 while loop execution

In the **while** loop, the condition is any valid Boolean expression returning True or False. The **else** part of while is optional part of **while**. The **statements block1** is kept executed till the condition is True. If the **else** part is written, it is executed when the condition is tested False. Recall **while** loop belongs to entry check loop type, that is it is not executed even once if the condition is tested False in the beginning.

Example 6.6: program to illustrate the use of while loop - to print all numbers from 10 to 15

```
i=10                    # intializing part of the control variable
while (i<=15):          # test condition
    print (i,end='\t')  # statements - block 1
    i=i+1               # Updation of the control variable
```

Output:

```
10    11    12    13    14    15
```

Note

That the control variable is **i**, which is initialized to 10, the condition is tested **i<=15,** if true value of **i** gets printed, then the control variable i gets updated as **i=i+1** (this can also be written as i +=1 using shorthand assignment operator). When **i** becomes **16**, the condition is tested False and this will terminate the loop.

Note

print can have end, sep as parameters. *end* parameter can be used when we need to give any escape sequences like '\t' for tab, '\n' for new line and so on. *sep* as parameter can be used to specify any special characters like, (comma) ; (semicolon) as separator between values (Recall the concept which you have learnt in previous chapter about the formatting options in print()).

Following is an example for using else part within while loop.

Example 6.7: program to illustrate the use of while loop - with else part

```
i=10                                        # intializing part of the control variable
while (i<=15):                              # test condition
      print (i,end='\t')                    # statements - block 1
      i=i+1                                 # Updation of the control variable
else:
    print ("\nValue of i when the loop exit ",i)
```

Output: 1

```
10    11    12    13    14    15
Value of i when the loop exit  16
```

(ii) for loop

for loop is the most comfortable loop. It is also an entry check loop. The condition is checked in the beginning and the body of the loop(statements-block 1) is executed if it is only True otherwise the loop is not executed.

Syntax:

for counter_variable in sequence:
statements-block 1
[else: # optional block
statements-block 2]

The *counter_variable* mentioned in the syntax is similar to the control variable that we

used in the **for** loop of C++ and the *sequence* refers to the initial, final and increment value. Usually in Python, **for** loop uses the *range()* function in the sequence to specify the initial, final and increment values. *range()* generates a list of values starting from **start** till **stop-1.**

The syntax of range() is as follows:

range (start,stop,[step])

Where,

start – refers to the initial value

stop – refers to the final value

step – refers to increment value, this is optional part.

Example 6.8: Examples for range()

range (1,30,1)	*will start the range of values from 1 and end at 29*
range (2,30,2)	*will start the range of values from 2 and end at 28*
range (30,3,-3)	*- will start the range of values from 30 and end at 6*
range (20)	*will consider this value 20 as the end value(or upper limit) and starts the range count from 0 to 19 (remember always range() will work till stop -1 value only)*

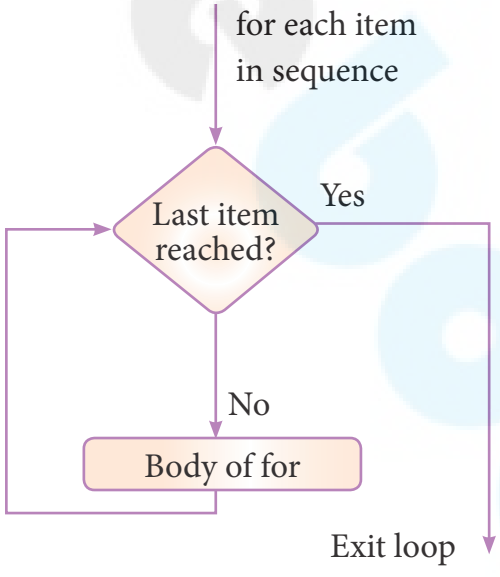


Fig 6.5 for loop execution

Example 6.9: #program to illustrate the use of for loop - to print single digit even number

```
for i in range (2,10,2):
        print (i, end=' ')
```

Output:

2 4 6 8

Following is an illustration using else part in for loop

Example 6.10 : #program to illustrate the use of for loop - to print single digit even number with else part

```
for i in range(2,10,2):
        print (i,end=' ')
else:
        print ("\nEnd of the loop")
```

Output:

```
2 4 6 8
End of the loop
```

Note

In Python, indentation is important in loop and other control statements. Indentation only creates blocks and sub-blocks like how we create blocks within a set of { } in languages like C, C++ etc.

Here is another program which illustrates the use of range() to find the sum of numbers 1 to 100

Example 6.11: # program to calculate the sum of numbers 1 to 100

```
n = 100
sum = 0
for counter in range(1,n+1):
        sum = sum + counter
print("Sum of 1 until %d: %d" % (n,sum))
```

Output:

Sum of 1 until 100: 5050

In the above code, *n* is initialized to 100, *sum* is initialized to 0, the *for* loop starts executing from 1, for every iteration the value of sum is added with the value of counter variable and stored in sum. Note that the for loop will iterate from 1 till the upper limit -1 (ie. Value of n is set as 100, so this loop will iterate for values from 1 to 99 only, that is the reason why we have set the upper limit as n+1)

Note

range () can also take values from string, list, dictionary etc. which will be dealt in the later chapters.

Following is an example to illustrate the use of string in range()

Example 6.12: program to illustrate the use of string in range() of for loop

```
for word in 'Computer':
    print (word,end=' ')
else:
    print ("\nEnd of the loop")
```

Output

```
C o m p u t e r
End of the loop
```

(iii) Nested loop structure

A loop placed within another loop is called as nested loop structure. One can place a **while** within another **while; for** within another **for; for** within **while** and **while** within **for** to construct such nested loops.

Following is an example to illustrate the use of for loop to print the following pattern

```
1
1    2
1    2    3
1    2    3    4
1    2    3    4    5
```

Example 6.13: program to illustrate the use nested loop -for within while loop

```
i=1
while (i<=6):
    for j in range (1,i):
        print (j,end='\t')
    print (end='\n')
    i +=1
```

Output:

```
1
1    2
1    2  3
1    2   3     4
1    2   3     4       5
```

6.2.4 Jump Statements in Python

The jump statement in Python, is used to unconditionally transfer the control from one part of the program to another. There are three keywords to achieve jump statements in Python : **break, continue, pass.** The following flowchart illustrates the use of break and continue.

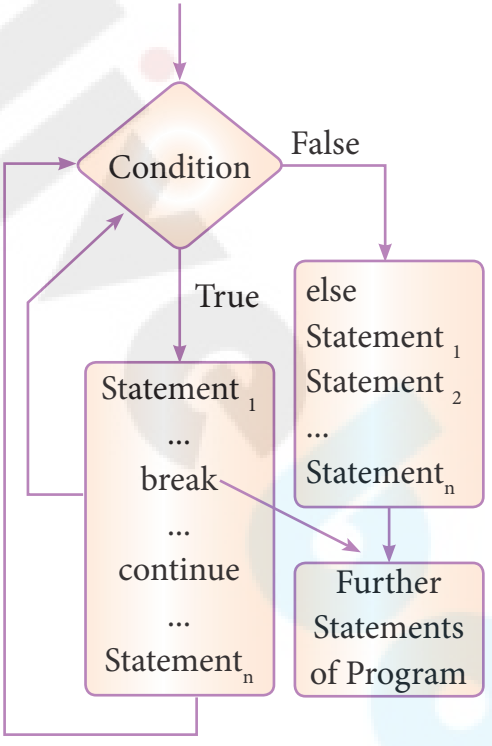


Fig 6.6 Use of break, continue statement in loop structure

(i) break statement

The **break** statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.

A **while** or **for** loop will iterate till the condition is tested false, but one can even transfer the control out of the loop (terminate) with help of **break** statement. When the break statement is executed, the control flow of the program comes out of the loop and starts executing the segment of code after the loop structure.

If break statement is inside a nested loop (loop inside another loop), break will terminate the innermost loop.

Syntax:

break

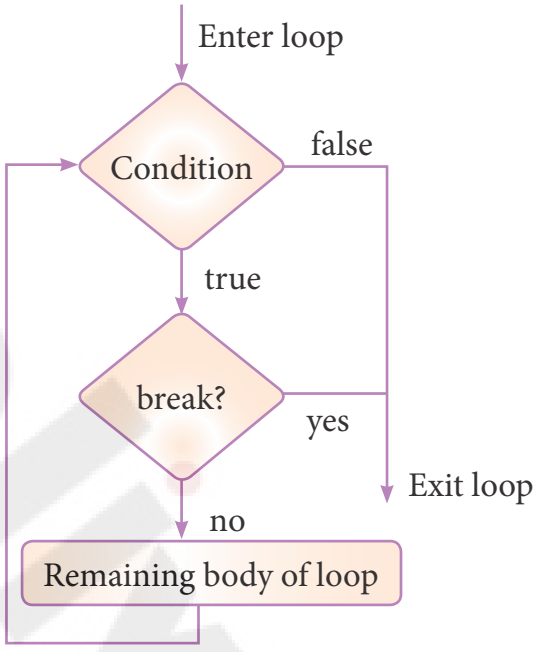


Fig 6.7 Working of break statement

The working of break statement in **for** loop and **while** loop is shown below.

```
for var in sequence:
        if condition:
                break
                #code inside for loop
#code outside for loop
while test expression:
        #code inside while loop
        if condition:
            break
        #code inside while loop
#code outside while loop
```

Example 6.14: Program to illustrate the use of break statement inside for loop

```
for word in "Jump Statement":
    if word = = "e":
        break
        print (word, end= ' ')
```

Output:

```
Jump Stat
```

The above program will repeat the iteration with the given "Jump Statement" as string. Each letter of the given string sequence is tested till the letter 'e' is encountered, when it is encountered the control is transferred outside the loop block or it terminates. As shown in the output, it is displayed till the letter 'e' is checked after which the loop gets terminated.

One has to note an important point here is that **'if a loop is left by break, the else part is not executed'.** To explain this lets us enhance the previous program with an 'else' part and see what output will be:

Example 6.15: Program to illustrate the use of break statement inside for loop

```
for word in "Jump Statement":
    if word = = "e":
        break
        print (word, end="")
else:
        print ("End of the loop")
print ("\n End of the program")
```

Output:

```
Jump Stat
End of the program
```

Note that the **break** statement has even skipped the 'else' part of the loop and has transferred the control to the next line following the loop block.

(ii) continue statement

Continue statement unlike the break statement is used to skip the remaining part of a loop and start with next iteration.

Syntax:

continue

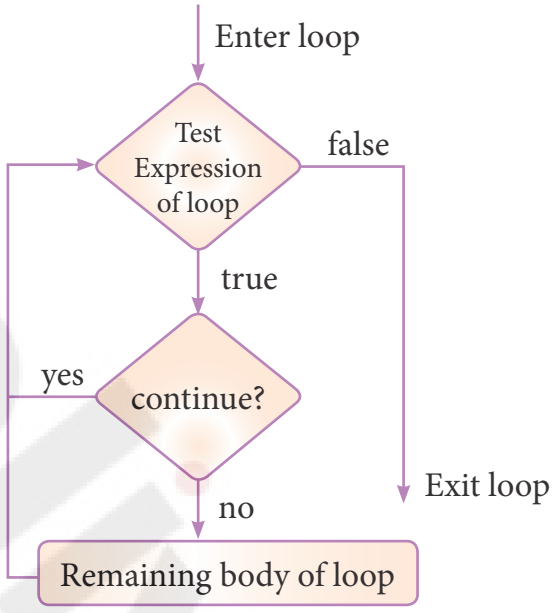


Fig 6.8 Working of continue statement

The working of continue statement in **for** and **while** loop is shown below.

```
for var in sequence:
        # code inside for loop
        if condition:
                continue
                #code inside for loop
#code outside for loop
while test expression:
        #code inside while loop
        if condition:
                continue
                #code inside while loop
#code outside while loop
```

Example 6.16: Program to illustrate the use of continue statement inside for loop

```
for word in "Jump Statement":
    if word = = "e":
        continue
    print (word, end = ' ')
print ("\n End of the program")
```

Output:

```
Jump Statmnt
End of the program
```

The above program is same as the program we had written for 'break' statement except that we have replaced it with 'continue'. As you can see in the output except the letter 'e' all the other letters get printed.

(iii) pass statement

pass statement in Python programming is a null statement. pass statement when executed by the interpreter it is completely ignored. Nothing happens when pass is executed, it results in no operation.

pass statement can be used in **'if'** clause as well as within loop construct, when you do not want any statements or commands within that block to be executed.

Syntax:

pass

Example 6.17: Program to illustrate the use of pass statement

```
a=int (input("Enter any number :"))
if (a==0):
    pass
else:
    print ("non zero value is accepted")
```

Output:

```
Enter any number :3
non zero value is accepted
```

When the above code is executed if the input value is 0 (zero) then no action will be performed, for all the other input values the output will be as follows:

Note

pass statement is generally used as a placeholder. When we have a loop or function that is to be implemented in the future and not now, we cannot develop such functions or loops with empty body segment because the interpreter would raise an error. So, to avoid this we can use pass statement to construct a body that does nothing.

Example 6.18: Program to illustrate the use of pass statement in for loop

```
for val in "Computer":
        pass
print ("End of the loop, loop structure will be built in future")
```

Output:

```
End of the loop, loop structure will be built in future.
```

Points to remember:

- Programs consists of statements which are executed in sequence, to alter the flow we use control statements.
- A program statement that causes a jump of control from one part of the program to another is called control structure or control statement.
- Three types of flow of control are
  - Sequencing
  - Branching or Alternative
  - Iteration
- In Python, branching is done using various forms of 'if' structures.
- Indentation plays a vital role in Python programming, it is the indentation that group statements no need to use {}.
- Python Interpreter will throw error for all indentation errors.
- To accept input at runtime, earlier versions of Python supported raw_input(), latest versions support input().
- print() supports the use of escape sequence to format the output to the user's choice.
- range() is used to supply a range of values in for loop.
- break, continue, pass act as jump statements in Python.
- pass statement is a null statement, it is generally used as a place holder.

Hands on Experience

1. Write a program to check whether the given character is a vowel or not.
2. Using if..else..elif statement check smallest of three numbers.
3. Write a program to check if a number is Positive, Negative or zero.
4. Write a program to display Fibonacci series 0 1 1 2 3 4 5…… (upto n terms)
5. Write a program to display sum of natural numbers, upto n.
6. Write a program to check if the given number is a palindrome or not.
7. Write a program to print the following pattern

```
* * * * *
* * * *
* * *
* *
*
```

8. Write a program to check if the year is leap year or not.

Evaluation

Part - I

Choose the best answer **1 Marks**

1. How many important control structures are there in Python?

   A) 3 B) 4

   C) 5 D) 6

2. elif can be considered to be abbreviation of

   A) nested if B) if..else

   C) else if D) if..elif

3. What plays a vital role in Python programming?

   A) Statements B) Control

   C) Structure D) Indentation

4. Which statement is generally used as a placeholder?

   A) continue B) break

   C) pass D) goto

CCPMXM

5. The condition in the if statement should be in the form of

A) Arithmetic or Relational expression

B) Arithmetic or Logical expression

C) Relational or Logical expression

D) Arithmetic

6. Which is the most comfortable loop?

A) do..while B) while

C) for D) if..elif

7. What is the output of the following snippet?

```
i=1
while True:
        if i%3 ==0:
                break
        print(i,end='')
        i +=1
```

A) 12 B) 123

C) 1234 D) 124

8. What is the output of the following snippet?

```
T=1
while T:
        print(True)
        break
```

A) False B) True

C) 0 D) 1

9. Which amongst this is not a jump statement ?

A) for B) pass

C) continue D) break

10. Which punctuation should be used in the blank?

if <condition>_

statements-block 1

else:

statements-block 2

A) ; B) :

C) :: D) !

Part -II

Answer the following questions **2 Marks**

1. List the control structures in Python.
2. Write note on break statement.
3. Write is the syntax of if..else statement
4. Define control structure.
5. Write note on range () in loop

Part -III

Answer the following questions **3 Marks**

1. Write a program to display

   A

   A B

   A B C

   A B C D

   A B C D E
2. Write note on if..else structure.
3. Using if..else..elif statement write a suitable program to display largest of 3 numbers.
4. Write the syntax of while loop.
5. List the differences between break and continue statements.

Part -IV

Answer the following questions **5 Marks**

1. Write a detail note on for loop
2. Write a detail note on if..else..elif statement with suitable example.
3. Write a program to display all 3 digit odd numbers.
4. Write a program to display multiplication table for a given number.

Unit II

CHAPTER 7

PYTHON FUNCTIONS

D7HGYM

Learning Objectives

After studying this chapter, students will be able to:

- Understand the concept of function and their types.
- Know the difference between User defined and Built in functions.
- Know how to call a function.
- Understand the function arguments.
- Know Anonymous functions.
- Know Mathematical and some String functions.

7.1 Introduction

Functions are named blocks of code that are designed to do specific job. When you want to perform a particular task that you have defined in a function, you call the name of the function responsible for it. If you need to perform that task multiple times throughout your program, you don't need to type all the code for the same task again and again; you just call the function dedicated to handling that task, and the call tells Python to run the code inside the function. You'll find that using functions makes your programs easier to write, read, test, and fix errors.

Main advantages of functions are

- *It avoids repetition and makes high degree of code reusing.*
- *It provides better modularity for your application.*

Note

Functions are nothing but a group of related statements that perform a specific task.

7.1.1 Types of Functions

Basically, we can divide functions into the following types:

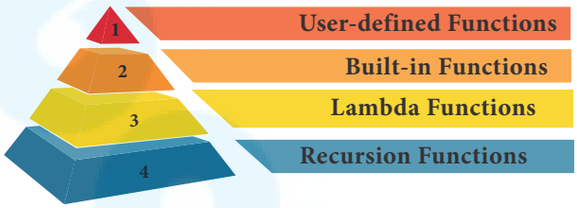


Figure – 7.1 – Types of Python Functions

Functions	Description
User-defined functions	Functions defined by the users themselves.
Built-in functions	Functions that are inbuilt with in Python.
Lambda functions	Functions that are anonymous un-named function.

XII Std - CS EM Chapter-7.indd 89 24-01-2020 11:03:48

Recursion functions	Functions that calls itself is known as recursive.

Table – 7.1 – Python Functions and it's Description

7.2 Defining Functions

Functions must be defined, to create and use certain functionality. There are many built-in functions that comes with the language python (for instance, the print() function), but you can also define your own function. When defining functions there are multiple things that need to be noted;

- Function blocks begin with the keyword **"def"** followed by function name and parenthesis ().
- Any input parameters or arguments should be placed within these parentheses when you define a function.
- The code block always comes after a colon (:) and is indented.
- The statement **"return [expression]"** exits a function, optionally passing back an expression to the caller. A **"return"** with no arguments is the same as return None.

Note

Python keywords should not be used as function name.

7.2.1 Syntax for User defined function

```
def <function_name ([parameter1, parameter2...] )> :
        <Block of Statements>
        return <expression / None>
```

Note

In the above Syntax, the Text which is given in square bracket [] is optional.

Block:

A block is *one or more lines of code*, grouped together so that they are treated as one big sequence of statements while execution. In Python, statements in a block are written with *indentation*. Usually, a block begins when a line is indented (by four spaces) and all the statements of the block should be at same indent level.

Nested Block:

A block within a block is called nested block. When the first block statement is indented by a single tab space, the second block of statement is indented by double tab spaces.

Here is an example of defining a function;

```
Do_Something( ):
value =1          #Assignment Statement
return value      #Return Statement
```


XII Std - CS EM Chapter-7.indd 90 24-01-2020 11:03:48

Now let's check out functions in action so you can visually see how they work within a program. Here is an example for a simple function to display the given string.

Example:

```
def hello():
        print ("hello - Python")
        return
```

7.2.2 Advantages of User-defined Functions

1. Functions help us to divide a program into modules. This makes the code easier to manage.
2. It implements code reuse. Every time you need to execute a sequence of statements, all you need to do is to call the function.
3. Functions, allows us to change functionality easily, and different programmers can work on different functions.

7.3 Calling a Function

To call the hello() function from *example 7.2-1*, you use this code:

When you call the "hello()" function, the program displays the following string as output:

Output

hello – Python

Alternatively we can call the "hello()" function within the print() function as in the example given below.

Example:

```
def hello():
        print ("hello - Python")
        return
print (hello())
```

If the return has no argument, **"None"** will be displayed as the last statement of the output.

The above function will output the following.

Output:

hello – Python
None

XII Std - CS EM Chapter-7.indd 91 24-01-2020 11:03:49

7.4 Passing Parameters in Functions

Parameters or arguments can be passed to functions

def function_name (parameter(s) separated by comma):

Let us see the use of parameters while defining functions. The parameters that you place in the parenthesis will be used by the function itself. You can pass all sorts of data to the functions. Here is an example program that defines a function that helps to pass parameters into the function.

Example:

```
# assume w = 3 and h = 5
def area(w,h):
    return w * h
print (area (3,5))
```

The above code assigns the width and height values to the parameters **w** and **h**. These parameters are used in the creation of the function "area". When you call the above function, it returns the product of width and height as output.

The value of 3 and 5 are passed to **w** and **h** respectively, the function will return 15 as output.

DO YOU KNOW

We often use the terms parameters and arguments interchangeably. However, there is a slight difference between them. Parameters are the variables used in the function definition whereas arguments are the values we pass to the function parameters

7.5 Function Arguments

Arguments are used to call a function and there are primarily 4 types of functions that one can use: *Required arguments, Keyword arguments, Default arguments and Variable-length arguments.*

XII Std - CS EM Chapter-7.indd 92 24-01-2020 11:03:49

Function Arguments

1. Required arguments
2. Keyword arguments
3. Default arguments
4. Variable-length arguments

7.5.1 Required Arguments

"Required Arguments" are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function definition. You need atleast one parameter to prevent syntax errors to get the required output.

Example :

```
def printstring(str):
        print ("Example - Required  arguments ")
        print (str)
        return
# Now you can call printstring() function
printstring()
```

When the above code is executed, it produces the following error.

```
Traceback (most recent call last):
File "Req-arg.py", line 10, in <module>
printstring()
    TypeError: printstring() missing 1 required positional argument: 'str'
```

Instead of **printstring()** in the above code if we use **printstring ("Welcome")** then the output is

Output:

Example - Required arguments
Welcome

XII Std - CS EM Chapter-7.indd 93 24-01-2020 11:03:49

7.5.2 Keyword Arguments

Keyword arguments will invoke the function after the parameters are recognized by their parameter names. The value of the keyword argument is matched with the parameter name and so, one can also put arguments in improper order (not in order).

Example:

```
def printdata (name):
        print ("Example-1 Keyword arguments")
        print ("Name :",name)
        return
# Now you can call printdata() function
printdata(name = "Gshan")
```

When the above code is executed, it produces the following output :

Output:

Example-1 Keyword arguments
Name :Gshan

Example:

```
def printdata (name):
        print ("Example-2 Keyword arguments")
        print ("Name :", name)
        return
# Now you can call printdata() function
printdata (name1 = "Gshan")
```

When the above code is executed, it produces the following result :

TypeError: printdata() got an unexpected keyword argument 'name1'

Example:

```
def printdata (name, age):
        print ("Example-3 Keyword arguments")
        print ("Name :",name)
        print ("Age :",age)
        return
# Now you can call printdata()  function
printdata (age=25, name="Gshan")
```


XII Std - CS EM Chapter-7.indd 94 24-01-2020 11:03:49

When the above code is executed, it produces the following result:

Output:

Example-3 Keyword arguments
Name : Gshan
Age : 25

Note

In the above program the parameters orders are changed

7.5.3 Default Arguments

In Python the default argument is an argument that takes a default value if no value is provided in the function call. The following example uses default arguments, that prints default salary when no argument is passed.

Example:

```
def printinfo( name, salary = 3500):
        print ("Name: ", name)
        print ("Salary: ", salary)
        return
printinfo("Mani")
```

When the above code is executed, it produces the following output

Output:

Name: Mani
Salary: 3500

When the above code is changed as print info("Ram",2000) it produces the following output:

Output:

Name: Ram
Salary: 2000

In the above code, the value 2000 is passed to the argument salary, the default value already assigned for salary is simply ignored.

XII Std - CS EM Chapter-7.indd 95 24-01-2020 11:03:49

7.5.4 Variable-Length Arguments

In some instances you might need to pass more arguments than have already been specified. Going back to the function to redefine it can be a tedious process. Variable-Length arguments can be used instead. These are not specified in the function's definition and an asterisk (*) is used to define such arguments.

Lets see what happens when we pass more than 3 arguments in the sum() function.

Example:

```
def sum(x,y,z):
        print("sum of three nos :",x+y+z)
sum(5,10,15,20,25)
```

When the above code is executed, it produces the following **result** :

TypeError: sum() takes 3 positional arguments but 5 were given

7.5.4.1 Syntax - Variable-Length Arguments

```
def function_name(*args):
        function_body
        return_statement
```

Example:

```
def printnos (*nos):
        for n in nos:
        print(n)
        return
# now invoking the printnos() function
print ('Printing two values')
printnos (1,2)
print ('Printing three values')
printnos (10,20,30)
```

Output:

```
Printing two values
        1
        2
Printing three values
        10
        20
        30
```

Evaluate Yourself

In the above program change the function name printnos as printnames in all places wherever it is used and give the appropriate data Ex. printnos (10, 20, 30) as printnames ('mala', 'kala', 'bala') and see output.

XII Std - CS EM Chapter-7.indd 96 24-01-2020 11:03:49

In Variable Length arguments we can pass the arguments using two methods.

1. Non keyword variable arguments
2. Keyword variable arguments

Non-keyword variable arguments are called **tuples**. You will learn more about tuples in the later chapters. The Program given is an illustration for non keyword variable argument.

Note

Keyword variable arguments are beyond the scope of this book.

DID YOU KNOW?

The Python's print() function is itself an example of such a function which supports variable length arguments.

7.6 Anonymous Functions

What is anonymous function?

In Python, anonymous function is a function that is defined without a name. While normal functions are defined using the **def** keyword, in Python anonymous functions are defined using the **lambda** keyword. Hence, anonymous functions are also called as **lambda** functions.

What is the use of lambda or anonymous function?

- Lambda function is mostly used for creating small and one-time anonymous function.
- Lambda functions are mainly used in combination with the functions like filter(), map() and reduce().

Note

filter(), map() and reduce() functions are beyond the scope of this book.

DID YOU KNOW?

Lambda function can take any number of arguments and must return one value in the form of an expression. Lambda function can only access global variables and variables in its parameter list.

XII Std - CS EM Chapter-7.indd 97 24-01-2020 11:03:49

7.6.1 Syntax of Anonymous Functions

The syntax for anonymous functions is as follows:

lambda [argument(s)] :expression

Example:

```
sum = lambda arg1, arg2: arg1 + arg2
print ('The Sum is :', sum(30,40))
print ('The Sum is :', sum(-30,40))
```

Output:

```
The Sum is : 70
The Sum is : 10
```

The above lambda function that adds argument ***arg1*** with argument ***arg2*** and stores the result in the variable sum. The result is displayed using the print().

7.7 The return Statement

- The return statement causes your function to exit and returns a value to its caller. The point of functions in general is to take inputs and return something.
- The return statement is used when a function is ready to return a value to its caller. So, only one return statement is executed at run time even though the function contains multiple return statements.
- Any number of 'return' statements are allowed in a function definition but only one of them is executed at run time.

7.7.1 Syntax of return

return [expression list]

This statement can contain expression which gets evaluated and the value is returned. If there is no expression in the statement or the return statement itself is not present inside a function, then the function will return the None object.

XII Std - CS EM Chapter-7.indd 98 24-01-2020 11:03:49

Example :

```
# return statment
def usr_abs (n):
        if n>=0:
                return n
        else:
                return –n
# Now invoking the function
x=int (input("Enter a number :")
print (usr_abs (x))
```

Output 1:

```
        Enter a number : 25
                25
```

Output 2:

```
        Enter a number : -25
                25
```

7.8 Scope of Variables

Scope of variable refers to the part of the program, where it is accessible, i.e., area where you can refer (use) it. We can say that scope holds the current set of variables and their values. We will study two types of scopes - **local scope** and **global scope**.

7.8.1 Local Scope

A variable declared inside the function's body or in the local scope is known as local variable.

Rules of local variable

- A variable with local scope can be accessed only within the function/block that it is created in.
- When a variable is created inside the function/block, the variable becomes local to it.
- A local variable only exists while the function is executing.
- The formal arguments are also local to function.

XII Std - CS EM Chapter-7.indd 99 24-01-2020 11:03:49

Example : Create a Local Variable

```
def loc():
        y=0 # local scope
        print(y)
loc()
```

Output:

```
0
```

Example : Accessing local variable outside the scope

```
def loc():
        y = "local"
loc()
print(y)
```

When we run the above code, the output shows the following error:

The above error occurs because we are trying to access a local variable **'y'** in a global scope.

NameError: name 'y' is not defined

7.8.2 Global Scope

A variable, with global scope can be used anywhere in the program. It can be created by defining a variable outside the scope of any function/block.

Rules of global Keyword

The basic rules for ***global*** keyword in Python are:

- When we define a variable outside a function, it's global by default. You don't have to use global keyword.
- We use global keyword to read and write a global variable inside a function.
- Use of global keyword outside a function has no effect

Use of global Keyword

Example : Accessing global Variable From Inside a Function

```
c = 1          # global variable
def add():
        print(c)
add()
```

Output:

```
1
```


XII Std - CS EM Chapter-7.indd 100 24-01-2020 11:03:49

Example : Accessing global Variable From Inside a Function

```
c = 1            # global variable
def add():
        print(c)
        add()
```

Output:

```
1
```

Example : Modifying Global Variable From Inside the Function

```
c = 1            # global variable
def add():
        c = c + 2 # increment c by 2
        print(c)
add()
```

Output:

```
Unbound Local Error: local variable 'c' referenced before assignment
```

Note

Without using the global keyword we cannot modify the global variable inside the function but we can only access the global variable.

Example : Changing Global Variable From Inside a Function using global keyword

```
x = 0                    # global variable
def add():
        global x
        x = x + 5        # increment by 2
        print ("Inside add() function  x value is :", x)
add()
print ("In main x value is :", x)
```

Output:

```
Inside add() function  x value is : 5
In main x value is : 5
```

In the above program, **x** is defined as a **global** variable. Inside the **add()** function, **global** keyword is used for **x** and we increment the variable **x** by **5**. Now We can see the change on the **global** variable **x** outside the function i.e the value of **x** is **5**.

XII Std - CS EM Chapter-7.indd 101 24-01-2020 11:03:49

7.8.3 Global and local variables

Here, we will show how to use global variables and local variables in the same code.

Example : Using Global and Local variables in same code

```
x=8                    # x is a global variable
def loc():
        global x
        y = "local"
        x = x * 2
        print(x)
        print(y)
loc()
```

Output:

```
16
local
```

In the above program, we declare ***x*** as global and ***y*** as local variable in the function **loc()**.

After calling the function **loc()**, the value of ***x*** becomes 16 because we used **x=x * 2.** After that, we print the value of local variable ***y*** i.e. local.

Example : Global variable and Local variable with same name

```
x = 5
def loc():
        x = 10
        print ("local x:", x)
loc()
print ("global x:", x)
```

Output:

```
local x: 10
global x: 5
```

In above code, we used same name 'x' for both global variable and local variable. We get different result when we print same variable because the variable is declared in both scopes, i.e. the local scope inside the function loc() and global scope outside the function loc().

The output :- local x: 10, is called local scope of variable.

The output:- global x: 5, is called global scope of variable.

XII Std - CS EM Chapter-7.indd 102 24-01-2020 11:03:49

7.9 Functions using libraries

7.9.1 Built-in and Mathematical functions

Function	Description	Syntax	Example
abs ( )	Returns an absolute value of a number. The argument may be an integer or a floating point number.	abs (x)	x=20 y=-23.2 print('x = ', abs(x)) print('y = ', abs(y)) **Output:** x = 20 y = 23.2
ord ( )	Returns the ASCII value for the given Unicode character. This function is inverse of chr() function.	ord (c)	c= 'a' d= 'A' print ('c = ',ord (c)) print ('A = ',ord (d)) **Output:** c = 97 A = 65
chr ( )	Returns the Unicode character for the given ASCII value. This function is inverse of ord() function.	chr (i)	c=65 d=43 print (chr (c)) prin t(chr (d)) **Output:** A +
bin ( )	Returns a binary string prefixed with "0b" for the given integer number. **Note:** format () can also be used instead of this function.	bin (i)	x=15 y=101 print ('15 in binary : ',bin (x)) print ('101 in binary : ',bin (y)) **Output:** 15 in binary : 0b1111 101 in binary : 0b1100101

XII Std - CS EM Chapter-7.indd 103 24-01-2020 11:03:49

type ( )	Returns the type of object for the given single object. **Note:** This function used with single object parameter.	type (object)	x= 15.2 y= 'a' s= True print (type (x)) print (type (y)) print (type (s)) **Output:** <class 'float'> <class 'str'> <class 'bool'>
id ( )	id( ) Return the "identity" of an object. i.e. the address of the object in memory. **Note:** the address of x and y may differ in your system.	id (object)	x=15 y='a' print ('address of x is :',id (x)) print ('address of y is :',id (y)) **Output:** address of x is : 1357486752 address of y is : 13480736
min ( )	Returns the minimum value in a list.	min (list)	MyList = [21,76,98,23] print ('Minimum of MyList :', min(MyList)) **Output:** Minimum of MyList : 21
max ( )	Returns the maximum value in a list.	max (list)	MyList = [21,76,98,23] print ('Maximum of MyList :', max(MyList)) **Output:** Maximum of MyList : 98
sum ( )	Returns the sum of values in a list.	sum (list)	MyList = [21,76,98,23] print ('Sum of MyList :', sum(MyList)) **Output:** Sum of MyList : 218

XII Std - CS EM Chapter-7.indd 104 24-01-2020 11:03:49

format ( )	Returns the output based on the given format 1. Binary format. Outputs the number in base 2. 2. Octal format. Outputs the number in base 8. 3. Fixed-point notation. Displays the number as a fixed-point number. The default precision is 6.	format (value [, format_ spec])	x= 14 y= 25 print ('x value in binary :',format(x,'b')) print ('y value in octal :',format(y,'o')) print('y value in Fixed-point no ',format(y,'f')) **Output:** x value in binary : 1110 y value in octal : 31 y value in Fixed-point no : 25.000000
round ( )	Returns the nearest integer to its input. 1. First argument (number) is used to specify the value to be rounded.	round (number [,ndigits])	x= 17.9 y= 22.2 z= -18.3 print ('x value is rounded to', round (x)) print ('y value is rounded to', round (y)) print ('z value is rounded to', round (z))

XII Std - CS EM Chapter-7.indd 105 24-01-2020 11:03:49

	2. Second argument (ndigits) is used to specify the number of decimal digits desired after rounding.		**Output:1** x value is rounded to 18 y value is rounded to 22 z value is rounded to -18 n1=17.89 print (round (n1,0)) print (round (n1,1)) print (round (n1,2)) **Output:2** 18.0 17.9 17.89
pow ( )	Returns the computation of ab i.e. (a**b) a raised to the power of b.	pow (a,b)	a= 5 b= 2 c= 3.0 print (pow (a,b)) print (pow (a,c)) print (pow (a+b,3)) **Output:** 25 125.0 343

Mathematical Functions

Note

Specify **import math** module before using all mathematical functions in a program

Function	Description	Syntax	Example
floor ( )	Returns the largest integer less than or equal to x	math.floor (x)	import math x=26.7 y=-26.7 z=-23.2 print (math.floor (x)) print (math.floor (y)) print (math.floor (z)) **Output:** 26 -27 -24

XII Std - CS EM Chapter-7.indd 106 24-01-2020 11:03:49

ceil ( )	Returns the smallest integer greater than or equal to x	math.ceil (x)	import math x= 26.7 y= -26.7 z= -23.2 print (math.ceil (x)) print (math.ceil (y)) print (math.ceil (z)) **Output:** 27 -26 -23
sqrt ( )	Returns the square root of x Note: x must be greater than 0 (zero)	sqrt (x)	import math a= 30 b= 49 c= 25.5 print (math.sqrt (a)) print (math.sqrt (b)) print (math.sqrt (c)) **Output:** 5.477225575051661 7.0 5.049752469181039

7.9.2 Composition in functions

What is Composition in functions?

The value returned by a function may be used as an argument for another function in a nested manner. This is called **composition**. For example, if we wish to take a numeric value or an expression as a input from the user, we take the input string from the user using the function **input()** and apply **eval()** function to evaluate its value, for example:

Example :

```
# This program explains composition
        >>> n1 = eval (input ("Enter a number: "))
Enter a number: 234
        >>> n1
        234
>>> n2 = eval (input ("Enter an arithmetic expression: "))
        Enter an arithmetic expression: 12.0+13.0 * 2
>>> n2
        38.0
```

7.10 Python recursive functions

When a function calls itself is known as recursion. Recursion works like loop but sometimes it makes more sense to use recursion than loop. You can convert any loop to

XII Std - CS EM Chapter-7.indd 107 24-01-2020 11:03:49

recursion.

A recursive function calls itself. Imagine a process would iterate indefinitely if not stopped by some condition! Such a process is known as infinite iteration. The condition that is applied in any recursive function is known as base condition. A base condition is must in every recursive function otherwise it will continue to execute like an infinite loop.

Overview of how recursive function works

1. Recursive function is called by some external code.
2. If the base condition is met then the program gives meaningful output and exits.
3. Otherwise, function does some required processing and then calls itself to continue recursion.

Here is an example of recursive function used to calculate factorial.

Example :

```
def fact(n):
        if n == 0:
                return 1
        else:
                return n * fact (n-1)
print (fact (0))
print (fact (5))
```

Output:

```
1
120
```

DO YOU KNOW?

print(fact (2000)) will give Runtime Error after maximum recursion depth exceeded in comparison. This happens because python stops calling recursive function after 1000 calls by default. It also allows you to change the limit using sys.setrecursionlimit (limit_value).

Example:

```
import sys
sys.setrecursionlimit(3000)
def fact(n):
        if n == 0:
                return 1
        else:
                return n * fact(n-1)
print(fact (2000))
```


XII Std - CS EM Chapter-7.indd 108 24-01-2020 11:03:49

Points to remember:

- Functions are named blocks of code that are designed to do one specific job.
- Types of Functions are User defined, Built-in, lambda and recursion.
- Function blocks begin with the keyword "def" followed by function name and parenthesis ().
- A "return" with no arguments is the same as return None. Return statement is optional in python.
- In Python, statements in a block should begin with indentation.
- A block within a block is called nested block.
- Arguments are used to call a function and there are primarily 4 types of functions that one can use: Required arguments, Keyword arguments, Default arguments and Variable-length arguments.
- Required arguments are the arguments passed to a function in correct positional order.
- Keyword arguments will invoke the function after the parameters are recognized by their parameter names.
- A Python function allows to give the default values for parameters in the function definition. We call it as Default argument.
- Variable-Length arguments are not specified in the function's definition and an asterisk (*) is used to define such arguments.
- Anonymous Function is a function that is defined without a name.
- Scope of variable refers to the part of the program, where it is accessible, i.e., area where you can refer (use) it.
- The value returned by a function may be used as an argument for another function in a nested manner. This is called composition.
- A function which calls itself is known as recursion. Recursion works like a loop but sometimes it makes more sense to use recursion than loop.

XII Std - CS EM Chapter-7.indd 109 24-01-2020 11:03:49

Hands on Experience

1. Try the following code in the above program

Slno	code	Result
1	printinfo("3500")	
2	printinfo("3500","Sri")	
3	printinfo(name="balu")	
4	printinfo("Jose",1234)	
5	printinfo(" ",salary=1234)	

2. Evaluate the following functions and write the output

Slno	Function	Output
1	eval('25*2-5*4')	
2	math.sqrt(abs(-81))	
3	math.ceil(3.5+4.6)	
4	math.floor(3.5+4.6)	

3. Evaluate the following functions and write the output

Slno	function	Output
1	1) abs(-25+12.0)) 2) abs(-3.2)	
2	1) ord('2') 2) ord('$')	
3	type('s')	
4	bin(16)	
5	1) chr(13) 2) print(chr(13))	
6	1) round(18.2,1) 2) round(18.2,0) 3) round(0.5100,3) 4) round(0.5120,3)	

XII Std - CS EM Chapter-7.indd 110 24-01-2020 11:03:49

7	1) format(66, 'c') 2) format(10, 'x') 3) format(10, 'X') 4) format(0b110, 'd') 5) format(0xa, 'd')	
8	1) pow(2,-3) 2) pow(2,3.0) 3) pow(2,0) 4) pow((1+2),2) 5) pow(-3,2) 6) pow(2*2,2)	

Evaluation

Part - I

Choose the best answer: **(1 Mark)**

1. A named blocks of code that are designed to do one specific job is called as
   (a) Loop (b) Branching
   (c) Function (d) Block
2. A Function which calls itself is called as
   (a) Built-in (b) Recursion
   (c) Lambda (d) return
3. Which function is called anonymous un-named function
   (a) Lambda (b) Recursion
   (c) Function (d) define
4. Which of the following keyword is used to begin the function block?
   (a) define (b) for
   (c) finally (d) def
5. Which of the following keyword is used to exit a function block?
   (a) define (b) return
   (c) finally (d) def
6. While defining a function which of the following symbol is used.
   (a) ; (semicolon) (b) . (dot)
   (c) : (colon) (d) $ (dollar)

DM1BZY

XII Std - CS EM Chapter-7.indd 111 24-01-2020 11:03:49

7. In which arguments the correct positional order is passed to a function?

(a) Required (b) Keyword

(c) Default (d) Variable-length

8. Read the following statement and choose the correct statement(s).

(I) In Python, you don't have to mention the specific data types while defining function.

(II) Python keywords can be used as function name.

(a) I is correct and II is wrong

(b) Both are correct

(c) I is wrong and II is correct

(d) Both are wrong

9. Pick the correct one to execute the given statement successfully.

if ____ : print(x, " is a leap year")

(a) x%2=0 (b) x%4==0

(c) x/4=0 (d) x%4=0

10. Which of the following keyword is used to define the function testpython(): ?

(a) define (b) pass

(c) def (d) while

Part - II

Answer the following questions: **(2 Marks)**

1. What is function?
2. Write the different types of function.
3. What are the main advantages of function?
4. What is meant by scope of variable? Mention its types.
5. Define global scope.
6. What is base condition in recursive function
7. How to set the limit for recursive function? Give an example.

XII Std - CS EM Chapter-7.indd 112 24-01-2020 11:03:50

Part - III

Answer the following questions: **(3 Marks)**

1. Write the rules of local variable.
2. Write the basic rules for global keyword in python.
3. What happens when we modify global variable inside the function?
4. Differentiate ceil() and floor() function?
5. Write a Python code to check whether a given year is leap year or not.
6. What is composition in functions?
7. How recursive function works?
8. What are the points to be noted while defining a function?

Part - IV

Answer the following questions: **(5 Marks)**

1. Explain the different types of function with an example.
2. Explain the scope of variables with an example.
3. Explain the following built-in functions.
   (a) id()
   (b) chr()
   (c) round()
   (d) type()
   (e) pow()
4. Write a Python code to find the L.C.M. of two numbers.
5. Explain recursive function with an example.

Reference Books

1. *Python Tutorial book from tutorialspoint.com*
2. *Python Programming: A modular approach by Pearson – Sheetal, Taneja*
3. *Fundamentals of Python –First Programs by Kenneth A. Lambert*

XII Std - CS EM Chapter-7.indd 113 24-01-2020 11:03:50

Unit II

CHAPTER 8

STRINGS AND STRING MANIPULATION

Learning Objectives

After completion of this chapter, the student will be able to

- Know how to process text.
- Understanding various string functions in Python.
- Know how to format Strings.
- Know about String Slicing.
- Know about Strings application in real world.

EGT61Y

8.1 Introduction

String is a data type in python, which is used to handle array of characters. String is a sequence of Unicode characters that may be a combination of letters, numbers, or special symbols enclosed within single, double or even triple quotes.

Example

'Welcome to learning Python'

"Welcome to learning Python"

''' "Welcome to learning Python" '''

In python, strings are immutable, it means, once you define a string, it cannot be changed during execution.

8.2 Creating Strings

As we learnt already, a string in Python can be created using single or double or even triple quotes. String in single quotes cannot hold any other single quoted character in it, because the compiler will not recognize where to start and end the string. To overcome this problem, you have to use double quotes. Strings which contains double quotes should be define within triple quotes. Defining strings within triple quotes also allows creation of ***multiline*** strings.

XII Std - CS EM Chapter-8.indd 114 24-01-2020 11:18:42

Example

```
#A string defined within single quotes
>>> print ('Greater Chennai Corporation')
        Greater Chennai Corporation

#single quoted string defined within single quotes
>>> print ('Greater Chennai Corporation's student')
        SyntaxError: invalid syntax

#A string defined within double quotes
>>>print ("Computer Science")
        Computer Science

#double quoted string defined within double quotes
>>> print (''' "Computer Science" ''')
        "Computer Science"

#single and double quoted multiline string defined within triple quotes
>>> print (''' "Strings are immutable in 'Python',
                which means you can't make any changes
                once you declared" ''')

        "Strings are immutable in 'Python',
        which means you can't make any changes once you declared"
```

8.3 Accessing characters in a String

Once you define a string, python allocate an index value for its each character. These index values are otherwise called as subscript which are used to access and manipulate the strings. The subscript can be positive or negative integer numbers.

The positive subscript **0** is assigned to the first character and **n-1** to the last character, where n is the number of characters in the string. The negative index assigned from the last character to the first character in reverse order begins with **-1**.

Example

String	S	C	H	O	O	L
Positive subscript	0	1	2	3	4	5
Negative subscript	-6	-5	-4	-3	-2	-1

XII Std - CS EM Chapter-8.indd 115 24-01-2020 11:18:42

Example 1 : Program to access each character with its positive subscript of a giving string

```
str1 = input ("Enter a string: ")
index=0
for i in str1:
        print ("Subscript[",index,"] : ", i)
        index + = 1
```

Output

```
Enter a string: welcome
Subscript [ 0 ] :  w
Subscript [ 1 ] :  e
Subscript [ 2 ] :  l
Subscript [ 3 ] :  c
Subscript [ 4 ] :  o
Subscript [ 5 ] :  m
Subscript [ 6 ] :  e
```

Example 2 : Program to access each character with its negative subscript of a giving string

```
str1 = input ("Enter a string: ")
index=-1
while index >= -(len(str1)):
        print ("Subscript[",index,"] : " + str1[index])
        index += -1
```

Output

```
Enter a string: welcome
Subscript [ -1 ] : e
Subscript [ -2 ] : m
Subscript [ -3 ] : o
Subscript [ -4 ] : c
Subscript [ -5 ] : l
Subscript [ -6 ] : e
Subscript [ -7 ] : w
```

8.4 Modifying and Deleting Strings

As you already learnt, strings in python are immutable. That means, once you define a string modifications or deletion is not allowed. If you want to modify the string, a new string value can be assign to the existing string variable.

XII Std - CS EM Chapter-8.indd 116 24-01-2020 11:18:42

Example

```
>>> str1="How are you"
>>> str1[0]="A"
Traceback (most recent call last):
        File "<pyshell#1>", line 1, in <module>
        str1[0]="A"
TypeError: 'str' object does not support item assignment
```

In the above example, string variable str1 has been assigned with the string "How are you" in statement 1. In the next statement, we try to update the first character of the string with character 'A'. But python will not allow the update and it shows a TypeError.

To overcome this problem, you can define a new string value to the existing string variable. Python completely overwrite new string on the existing string.

Example

```
>>> str1="How are you"
>>> print (str1)
        How are you
>>> str1="How about you"
>>> print (str1)
        How about you
```

Usually python does not support any modification in its strings. But, it provides a function replace() to change all occurrences of a particular character in a string.

General formate of replace function:

replace("char1", "char2")

The replace function replaces all occurrences of char1 with char2.

Example

```
>>> str1="How are you"
>>> print (str1)
        How are you
>>> print (str1.replace("o", "e"))
        Hew are yeu
```

Similar as modification, python will not allow deleting a particular character in a string. Whereas you can remove entire string variable using **del** command.

XII Std - CS EM Chapter-8.indd 117 24-01-2020 11:18:42

Example 3: Code lines to delete a particular character in a string:

```
>>> str1="How are you"
>>> del str1[2]
Traceback (most recent call last):
        File "<pyshell#7>", line 1, in <module>
                del str1[2]
        TypeError: 'str' object doesn't support item deletion
```

Example 4: Code lines to delete a string variable

```
>>> str1="How about you"
>>> print (str1)
        How about you
>>> del str1
>>> print (str1)
Traceback (most recent call last):
        File "<pyshell#14>", line 1, in <module>
                print (str1)
        NameError: name 'str1' is not defined
```

8.5 String Operators

Python provides the following operators for string operations. These operators are useful to manipulate string.

(i) Concatenation (+)

Joining of two or more strings is called as Concatenation. The plus (+) operator is used to concatenate strings in python.

Example

```
>>> "welcome" + "Python"
        'welcomePython'
```

(ii) Append (+ =)

Adding more strings at the end of an existing string is known as append. The operator += is used to append a new string with an existing string.

Example

```
>>> str1="Welcome to "
```


XII Std - CS EM Chapter-8.indd 118 24-01-2020 11:18:42

```
>>> str1+="Learn Python"
>>> print (str1)
```

Welcome to Learn Python

(iii) Repeating (*)

The multiplication operator (*) is used to display a string in multiple number of times.

Example

```
>>> str1="Welcome "
>>> print (str1*4)
```

Welcome Welcome Welcome Welcome

(iv) String slicing

Slice is a substring of a main string. A substring can be taken from the original string by using [ ] operator and index or subscript values. Thus, [ ] is also known as slicing operator. Using slice operator, you have to slice one or more substrings from a main string.

General format of slice operation:

str[start:end]

Where ***start*** is the beginning index and ***end*** is the last index value of a character in the string. Python takes the end value less than one from the actual index specified. For example, if you want to slice first 4 characters from a string, you have to specify it as 0 to 5. Because, python consider only the end value as n-1.

Example I : slice a single character from a string

```
>>> str1="THIRUKKURAL"
>>> print (str1[0])
```

T

Example II : slice a substring from index 0 to 4

```
>>> print (str1[0:5])
```

THIRU

Example III : slice a substring using index 0 to 4 but without specifying the beginning index.

```
>>> print (str1[:5])
```

THIRU

Example IV : slice a substring using index 0 to 4 but without specifying the end index.

```
>>> print (str1[6:])
```

KURAL

XII Std - CS EM Chapter-8.indd 119 24-01-2020 11:18:42

Example V : Program to slice substrings using for loop

```
str1="COMPUTER"
index=0
for i in str1:
        print (str1[:index+1])
        index+=1
```

Output

```
C
C O
C O M
C O M P
C O M P U
C O M P U T
C O M P U T E
C O M P U T E R
```

(v) Stride when slicing string

When the slicing operation, you can specify a third argument as the stride, which refers to the number of characters to move forward after the first character is retrieved from the string. The default value of stride is 1.

Example

```
>>> str1 = "Welcome to learn Python"
>>> print (str1[10:16])
        learn
>>> print (str1[10:16:4])
        r
>>> print (str1[10:16:2])
        er
>>> print (str1[::3])
Wceoenyo
```

Note: Remember that, python takes the last value as n-1

You can also use negative value as stride (third argument). If you specify a negative value, it prints in reverse order.

XII Std - CS EM Chapter-8.indd 120 24-01-2020 11:18:43

Example

```
>>> str1 = "Welcome to learn Python"
>>> print(str1[::-2])
nhy re teolW
```

8.6 String Formatting Operators

The string formatting operator is one of the most exciting feature of python. The formatting operator % is used to construct strings, replacing parts of the strings with the data stored in variables.

Syntax:

("String to be display with %val1 and %val2" %(val1, val2))

Example

```
name = "Rajarajan"
mark = 98
print ("Name: %s and Marks: %d" %(name,mark))
```

Output

```
Name: Rajarajan and Marks: 98
```

8.7 Formatting characters

Format characters	USAGE
%c	Character
%d (or) %i	Signed decimal integer
%s	String
%u	Unsigned decimal integer
%o	Octal integer
%x or %X	Hexadecimal integer (lower case x refers a-f; upper case X refers A-F)
%e or %E	Exponential notation
%f	Floating point numbers
%g or %G	Short numbers in floating point or exponential notation.

XII Std - CS EM Chapter-8.indd 121 24-01-2020 11:18:43

Escape sequence in python

Escape sequences starts with a backslash and it can be interpreted differently. When you have use single quote to represent a string, all the single quotes inside the string must be escaped. Similar is the case with double quotes.

Example

```
# String within triple quotes to display a string with single quote
>>> print ('''They said, "What's there?"''')
        They said, "What's there?"

# String within single quotes to display a string with single quote using escape sequence
>>> print ('They said, "What\'s there?"')
        They said, "What's there?"

# String within double quotes to display a string with single quote using escape sequence
>>> print ("They said, \"What's there?\"")
        He said, "What's there?"
```

Escape sequences supported by python

Escape Sequence	DESCRIPTION
\newline	Backslash and newline ignored
\\	Backslash
\'	Single quote
\"	Double quote
\a	ASCII Bell
\b	ASCII Backspace
\f	ASCII Form feed
\n	ASCII Linefeed
\r	ASCII Carriage Return
\t	ASCII Horizontal Tab
\v	ASCII Vertical Tab
\ooo	Character with octal value ooo
\xHH	Character with hexadecimal value HH

8.8 The format( ) function

The format( ) function used with strings is very versatile and powerful function used for formatting strings. The curly braces { } are used as placeholders or replacement fields which get replaced along with format( ) function.

XII Std - CS EM Chapter-8.indd 122 24-01-2020 11:18:43

Example

```
num1=int (input("Number 1: "))
num2=int (input("Number 2: "))
print ("The sum of { } and { } is { }".format(num1, num2,(num1+num2)))
```

Out Put

```
Number 1: 34
Number 2: 54
The sum of 34 and 54 is 88
```

8.9 Built-in String functions

Python supports the following built-in functions to manipulate string.

Syntax	Description	Example

len(str)	Returns the length (no of characters) of the string.	>>> A="Corporation"
		>>> print(len(A))
		11

capitalize( )	Used to capitalize the first character of the string	>>> city="chennai"
		>>> print(city.capitalize())
		Chennai

center(width, fillchar)	Returns a string with the original string centered to a total of width columns and filled with fillchar in columns that do not have characters	>>> str1="Welcome"
		>>> print(str1.center(15,'*'))
		***********Welcome********

find(sub[, start[, end]])	The function is used to search the first occurrence of the sub string in the given string. It returns the index at which the substring starts. It returns -1 if the substring does not occur in the string.	>>>str1='mammals' >>>str1.find('ma') ***0*** ***On omitting the start parameters, the function starts the search from the beginning.*** >>>str1.find('ma',2) 3 >>>str1.find('ma',2,4) -1 ***Displays -1 because the substring could not be found between the index 2 and 4-1.*** >>>str1.find('ma',2,5) 3

XII Std - CS EM Chapter-8.indd 123 24-01-2020 11:18:43

Syntax	Description	Example
isalnum( )	Returns True if the string contains only letters and digit. It returns False. If the string contains any special character like _, @, #, *, etc.	>>>str1='Save Earth' >>>str1.isalnum() ***False*** The function returns False as space is an alphanumeric character. >>>'Save1Earth'.isalnum() ***True***
isalpha( )	Returns True if the string contains only letters. Otherwise return False.	>>>'Click123'.isalpha() ***False*** >>>'python'.isalpha( ) ***True***
isdigit( )	Returns True if the string contains only numbers. Otherwise it returns False.	>>> str1='Save Earth' >>>print(str1.isdigit( )) ***False***
lower( )	Returns the exact copy of the string with all the letters in lowercase.	>>>str1='SAVE EARTH' >>>print(str1.lower()) ***save earth***
islower( )	Returns True if the string is in lowercase.	>>> str1='welcome' >>>print (str1.islower( )) ***True***
isupper( )	Returns True if the string is in uppercase.	>>> str1='welcome' >>>print (str1.isupper( )) ***False***
upper( )	Returns the exact copy of the string with all letters in uppercase.	>>> str1='welcome' >>>print (str.upper( )) ***WELCOME***

title( )	Returns a string in title case	>>> str1='education department'
		>>> print(str1.title())
		Education Department

swapcase( )	It will change case of every character to its opposite case vice-versa.	>>> str1="tAmiL NaDu"
		>>> print(str1.swapcase())
		TaMIl nAdU

XII Std - CS EM Chapter-8.indd 124 24-01-2020 11:18:43

Syntax	Description	Example

count(str, beg, end)	Returns the number of substrings occurs within the given range. Remember that substring may be a single character. Range (beg and end) arguments are optional. If it is not given, python searched in whole string. Search is case sensitive.	>>> str1="Raja Raja Chozhan"
		>>> print(str1.count('Raja'))
		2
		>>> print(str1.count('r'))
		0
		>>> prin***t(str1.count('R'))***
		2
		>>> print(str1.count('a'))
		5
		>>> print(str1.count('a',0,5))
		2
		>>> print(str1.count('a',11))
		1

ord(char)	Returns the ASCII code of the character.	>>> ch = 'A'
		>>> print(ord(ch))
		65
		>>> print(***ord('B'))***
		66

chr(ASII)	Returns the character represented by a ASCII.	>>> ch=97
		>>> print(chr(ch))
		a
		>>> print(chr(87))
		W

8.10 Membership Operators

The '**in**' and '**not in**' operators can be used with strings to determine whether a string is present in another string. Therefore, these operators are called as Membership Operators.

Example

```
str1=input ("Enter a string: ")
str2="chennai"
if str2 in str1:
        print ("Found")
else:
        print ("Not Found")
```

Output : 1

```
Enter a string: Chennai G HSS, Saidapet
Found
```

Output : 2

```
Enter a string: Govt G HSS, Ashok Nagar
Not Found
```


XII Std - CS EM Chapter-8.indd 125 24-01-2020 11:18:43

Example 8.11 Programs using Strings :

Example 8.11.1 : Program to check whether the given string is palindrome or not

```
str1 = input ("Enter a string: ")
str2 = ' '
index=-1
for i in str1:
        str2 += str1[index]
        index -= 1
print ("The given string = { } \n The Reversed string = { }".format(str1, str2))
if (str1==str2):
        print ("Hence, the given string is Palindrome")
else:
        print ("Hence, the given is not a palindrome")
```

Output : 1

```
Enter a string: malayalam
The given string = malayalam
The Reversed string = malayalam
Hence, the given string is Palindrome
```

Output : 2

```
Enter a string: welcome
The given string = welcome
The Reversed string = emoclew
Hence, the given string is not a palindrome
```

Example 8.11.2 : Program to display the following pattern

```
*
* *
* * *
* * * *
* * * * *
str1=' * '
i=1
while i<=5:
        print (str1*i)
        i+=1
```

Output

```
*
* *
* * *
* * * *
* * * * *
```


XII Std - CS EM Chapter-8.indd 126 24-01-2020 11:18:43

Example 8.11.3 : Program to display the number of vowels and consonants in the given string

```
str1=input ("Enter a string: ")
str2="aAeEiIoOuU"
v,c=0,0
for i in str1:
        if i in str2:
                v+=1
        else:
                c+=1
print ("The given string contains { } vowels and { } consonants".format(v,c))
```

Output

```
Enter a string: Tamilnadu School Education
The given string contains 11 vowels and 15 consonants
```

Example 8.11.4 : Program to create an Abecedarian series. (Abecedarian refers list of elements appear in alphabetical order)

```
str1="ABCDEFGH"
str2="ate"
for i in str1:
        print ((i+str2),end='\t')
```

Output

```
Aate    Bate    Cate    Date    Eate    Fate    Gate    Hate
```

Example 8.11.5 : Program that accept a string from the user and display the same after removing vowels from it

```
def rem_vowels(s):
        temp_str=''
        for i in s:
                if i in "aAeEiIoOuU":
                        pass
                else:
                        temp_str+=i
        print ("The string without vowels: ", temp_str)
str1= input ("Enter a String: ")
rem_vowels (str1)
```

Output

```
Enter a String: Mathematical fundations of Computer Science
The string without vowels:  Mthmtcl fndtns f Cmptr Scnc
```


XII Std - CS EM Chapter-8.indd 127 24-01-2020 11:18:43

Example 8.11.6 : Program that count the occurrences of a character in a string

```
def count(s, c):
        c1=0
        for i in s:
                if i == c:
                        c1+=1
        return c1
str1=input ("Enter a String: ")
ch=input ("Enter a character to be searched: ")
cnt=count (str1, ch)
print ("The given character {} is occurs {} times in the given string".format(ch,cnt))
```

Out Put

```
Enter a String: Software Engineering
Enter a character to be searched: e
The given character e is occurs 3 times in the given string
```

Points to remember:

- String is a data type in python.
- Strings are immutable, that means once you define string, it cannot be changed during execution.
- Defining strings within triple quotes also allows creation of multiline strings.
- In a String, python allocate an index value for its each character which is known as subscript.
- The subscript can be positive or negative integer numbers.
- Slice is a substring of a main string.
- Stride is a third argument in slicing operation.
- Escape sequences starts with a backslash and it can be interpreted differently.
- The format( ) function used with strings is very versatile and powerful function used for formatting strings.
- The 'in' and 'not in' operators can be used with strings to determine whether a string is present in another string.

Hands on Experience

1. Write a python program to find the length of a string.
2. Write a program to count the occurrences of each word in a given string.

XII Std - CS EM Chapter-8.indd 128 24-01-2020 11:18:43

3. Write a program to add a prefix text to all the lines in a string.
4. Write a program to print integers with '*' on the right of specified width.
5. Write a program to create a mirror image of the given string. For example, "wel" = "lew".
6. Write a program to removes all the occurrences of a give character in a string.
7. Write a program to append a string to another string without using += operator.
8. Write a program to swap two strings.
9. Write a program to replace a string with another string without using replace().
10. Write a program to count the number of characters, words and lines in a given string.

EWC13A

Evaluation

Part - I

Choose the best answer **(1 Mark)**

1. Which of the following is the output of the following python code?

```
str1="TamilNadu"
print(str1[::-1])
```

(a) Tamilnadu (b) Tmlau
(c) udanlimaT d) udaNlimaT

2. What will be the output of the following code?

```
str1 = "Chennai Schools"
str1[7] = "-"
```

(a) Chennai-Schools (b) Chenna-School
(c) Type error (D) Chennai

3. Which of the following operator is used for concatenation?

(a) + (b) & (c) * d) =

4. Defining strings within triple quotes allows creating:

(a) Single line Strings (b) Multiline Strings
(c) Double line Strings (d) Multiple Strings

5. Strings in python:

(a) Changeable (b) Mutable
(c) Immutable (d) flexible

XII Std - CS EM Chapter-8.indd 129 24-01-2020 11:18:43

6. Which of the following is the slicing operator?

   (a) { } (b) [ ] (c) < > (d) ( )

7. What is stride?

   (a) index value of slide operation (b) first argument of slice operation

   (c) second argument of slice operation (d) third argument of slice operation

8. Which of the following formatting character is used to print exponential notation in upper case?

   (a) %e (b) %E (c) %g (d) %n

9. Which of the following is used as placeholders or replacement fields which get replaced along with format( ) function?

   (a) { } (b) < > (c) ++ (d) ^^

10. The subscript of a string may be:

   (a) Positive (b) Negative

   (c) Both (a) and (b) (d) Either (a) or (b)

Part -II

Answer the following questions (2 Marks)

1. What is String?
2. Do you modify a string in Python?
3. How will you delete a string in Python?
4. What will be the output of the following python code?

   str1 = "School"

   print(str1*3)

5. What is slicing?

Part -III

Answer the following questions (3 Marks)

1. Write a Python program to display the given pattern

   C O M P U T E R
   C O M P U T E
   C O M P U T
   C O M P U
   C O M P
   C O M
   C O
   C

XII Std - CS EM Chapter-8.indd 130 24-01-2020 11:18:43

2. Write a short about the followings with suitable example:

   (a) capitalize( ) (b) swapcase( )

3. What will be the output of the given python program?

```
str1 = "welcome"
str2 = "to school"
str3=str1[:2]+str2[len(str2)-2:]
print(str3)
```

4. What is the use of format( )? Give an example.
5. Write a note about count( ) function in python.

Part -IV

Answer the following questions (5 Marks)

1. Explain about string operators in python with suitable example.

Reference Books

1. *https://docs.python.org/3/tutorial/index.html*
2. *https://www.techbeamers.com/python-tutorial-step-by-step/#tutorial-list*
3. *Python programming using problem solving approach – Reema Thareja – Oxford University press.*
4. *Python Crash Course – Eric Matthes – No starch press, San Francisco.*

XII Std - CS EM Chapter-8.indd 131 24-01-2020 11:18:43

Unit III

CHAPTER 9

LISTS, TUPLES, SETS AND DICTIONARY

FBUV4L

Learning Objectives

After studying this chapter, students will be able to:

- Understand the basic concepts of various collection data types in python such as List, Tuples, sets and Dictionary.
- Work with List, Tuples, sets and Dictionaries using variety of functions.
- Writting Python programs using List, Tuples, sets and Dictionaries.
- Understand the relationship between List, Tuples and Dictionaries.

9.1 Introduction to List

Python programming language has four collections of data types such as List, Tuples, Set and Dictionary. A list in Python is known as a **"sequence data type"** like strings. It is an ordered collection of values enclosed within square brackets [ ]. Each value of a list is called as element. It can be of any type such as numbers, characters, strings and even the nested lists as well. The elements can be modified or mutable which means the elements can be replaced, added or removed. Every element rests at some position in the list. The position of an element is indexed with numbers beginning with zero which is used to locate and access a particular element. Thus, lists are similar to arrays, what you learnt in XI std.

9.1.1 Create a List in Python

In python, a list is simply created by using square bracket. The elements of list should be specified within square brackets. The following syntax explains the creation of list.

Syntax:

Variable = [element-1, element-2, element-3 …… element-n]

Example

Marks = [10, 23, 41, 75]
Fruits = ["Apple", "Orange", "Mango", "Banana"]
MyList = [ ]

In the above example, the list Marks has four integer elements; second list Fruits has four string elements; third is an empty list. The elements of a list need not be homogenous type of data. The following list contains multiple type elements.

Mylist = ["Welcome", 3.14, 10, [2, 4, 6]]

In the above example, Mylist contains another list as an element. This type of list is known as **"Nested List".**

Nested list is a list containing another list as an element.

9.1.2 Accessing List elements

Python assigns an automatic index value for each element of a list begins with zero. Index value can be used to access an element in a list. In python, index value is an integer number which can be positive or negative.

Example

Marks = [10, 23, 41, 75]

Marks	10	23	41	75
Index (Positive)	0	1	2	3
IndexNegative)	-4	-3	-2	-1

Positive value of index counts from the beginning of the list and negative value means counting backward from end of the list (i.e. in reverse order).

To access an element from a list, write the name of the list, followed by the index of the element enclosed within square brackets.

Syntax:

List_Variable = [E1, E2, E3 …… En]

print (List_Variable[index of a element])

Example (Accessing single element):

```
>>> Marks = [10, 23, 41, 75]
>>> print (Marks[0])
10
```

In the above example, print command prints 10 as output, as the index of 10 is zero.

Example: Accessing elements in revevrse order

```
>>> Marks = [10, 23, 41, 75]
>>> print (Marks[-1])
75
```

Note

A negative index can be used to access an element in reverse order.

(i) Accessing all elements of a list

Loops are used to access all elements from a list. The initial value of the loop must be zero. Zero is the beginning index value of a list.

Example

```
Marks = [10, 23, 41, 75]
i = 0
while i < 4:
        print (Marks[i])
        i = i + 1
```

Output

```
10
23
41
75
```

In the above example, Marks list contains four integer elements i.e., 10, 23, 41, 75. Each element has an index value from 0. The index value of the elements are 0, 1, 2, 3 respectively. Here, the while loop is used to read all the elements. The initial value of the loop is zero, and the test condition is i < 4, as long as the test condition is true, the loop executes and prints the corresponding output.

During the first iteration, the value of **i** is **zero**, where the condition is true. Now, the following statement **print (Marks [i])** gets executed and prints the value of Marks [0] element ie. 10.

The next statement **i = i + 1** increments the value of **i** from **0** to **1**. Now, the flow of control shifts to the while statement for checking the test condition. The process repeats to print the remaining elements of **Marks** list until the test condition of while loop becomes false.

The following table shows that the execution of loop and the value to be print.

Iteration	i	while i < 4	print (Marks[i])	i = i + 1
1	0	0 < 4 True	Marks [0] = 10	0 + 1 = 1
2	1	1 < 4 True	Marks [1] = 23	1 + 1 = 2
3	2	2 < 4 True	Marks [2] = 41	2 + 1 = 3
4	3	3 < 4 True	Marks [3] = 75	3 + 1 = 4
5	4	4 < 4 False	--	--

(ii) Reverse Indexing

Python enables reverse or negative indexing for the list elements. Thus, python lists index in opposite order. The python sets -1 as the index value for the last element in list and -2 for the preceding element and so on. This is called as **Reverse Indexing.**

Example

```
Marks = [10, 23, 41, 75]
i = -1
while i >= -4:
        print (Marks[i])
        i = i + -1
```

Output

```
75
41
23
10
```

The following table shows the working process of the above python coding

Iteration	i	while i >= -4	print (Marks[i])	i = i + -1
1	-1	-1 >= -4 True	Marks[-1] = 75	-1 + (-1) = -2
2	-2	-2 >= -4 True	Marks[-2] = 41	-2 + (-1) = -3
3	-3	-3 >= -4 True	Marks[-3] = 23	-3 + (-1) = -4
4	-4	-4 >= -4 True	Marks[-4] = 10	-4 + (-1) = -5
5	-5	-5 >= -4 False	--	--

9.1.3 List Length

The len( ) function in Python is used to find the length of a list. (i.e., the number of elements in a list). Usually, the len( ) function is used to set the upper limit in a loop to read all the elements of a list. If a list contains another list as an element, len( ) returns that inner list as a single element.

Example :Accessing single element

```
>>> MySubject = ["Tamil", "English", "Comp. Science", "Maths"]
>>> len(MySubject)
    4
```

Example : Program to display elements in a list using loop

```
MySubject = ["Tamil", "English", "Comp. Science", "Maths"]
i = 0
while i < len(MySubject):
        print (MySubject[i])
        i = i + 1
```

Output

```
Tamil
English
Comp. Science
Maths
```

9.1.4 Accessing elements using for loop

In Python, the ***for*** loop is used to access all the elements in a list one by one. This is just like the ***for*** keyword in other programming language such as C++.

Syntax:

for *index_var* ***in*** *list:*
    print *(index_var)*

Here, ***index_var*** represents the index value of each element in the list. Python reads this "for" statement like English: *"For (every) element in (the list of) list and print (the name of the) list items"*

Example

```
Marks=[23, 45, 67, 78, 98]
for x in Marks:
        print( x )
```

Output

```
23
45
67
78
98
```

In the above example, Marks list has 5 elements; each element is indexed from 0 to 4. The Python reads the ***for*** loop and ***print*** statements like English: *"For (every) element (represented as x) in (the list of) Marks and print (the values of the) elements"*.

9.1.5 Changing list elements

In Python, the lists are mutable, which means they can be changed. A list element or range of elements can be changed or altered by using simple assignment operator (=).

Syntax:

List_Variable [index of an element] = Value to be changed
List_Variable [index from : index to] = Values to changed

Where, ***index from*** is the beginning index of the range; ***index to*** is the upper limit of the range which is excluded in the range. For example, if you set the range [0:5] means, Python takes only 0 to 4 as element index. Thus, if you want to update the range of elements from 1 to 4, it should be specified as [1:5].

Example 9.1: Python program to update/change single value

```
MyList = [2, 4, 5, 8, 10]
print ("MyList elements before update... ")
for x in MyList:
    print (x)
MyList[2] = 6
print ("MyList elements after updation... ")
for y in MyList:
    print (y)
```

Output:

```
MyList elements before update...
        2
        4
        5
        8
        10
MyList elements after updation...
        2
        4
        6
        8
        10
```

Example 9.2: Python program to update/change range of values

```
MyList = [1, 3, 5, 7, 9]
print ("List Odd numbers... ")
for x in MyList:
    print (x)
MyList[0:5] = 2,4,6,8,10
print ("List Even numbers... ")
for y in MyList:
    print (y)
```

Output

```
List Odd numbers...
        1
        3
        5
        7
        9
List Even numbers...
        2
        4
        6
        8
        10
```

9.1.6 Adding more elements in a list

In Python, append( ) function is used to add a single element and extend( ) function is used to add more than one element to an existing list.

Syntax:

List.append (element to be added)
List.extend ([elements to be added])

In extend( ) function, multiple elements should be specified within square bracket as arguments of the function.

Example

```
>>> Mylist=[34, 45, 48]
>>> Mylist.append(90)
>>> print(Mylist)
        [34, 45, 48, 90]
```

In the above example, Mylist is created with three elements. Through >>> **Mylist.append(90)** statement, an additional value 90 is included with the existing list as last element, following print statement shows all the elements within the list MyList.

Example

```
>>> Mylist.extend([71, 32, 29])
>>> print(Mylist)
        [34, 45, 48, 90, 71, 32, 29]
```

In the above code, extend( ) function is used to include multiple elements, the print statement shows all the elements of the list after the inclusion of additional elements.

9.1.7 Inserting elements in a list

As you learnt already, append( ) function in Python is used to add more elements in a list. But, it includes elements at the end of a list. If you want to include an element at your desired position, you can use insert ( ) function. The insert( ) function is used to insert an element at any position of a list.

Syntax:

List.insert (position index, element)

Example

```
>>> MyList=[34,98,47,'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan' ]
>>> print(MyList)
        [34, 98, 47, 'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan']
>>> MyList.insert(3, 'Ramakrishnan')
>>> print(MyList)
[34, 98, 47, 'Ramakrishnan', 'Kannan', 'Gowrisankar', 'Lenin', 'Sreenivasan']
```

In the above example, insert( ) function inserts a new element 'Ramakrishnan' at the index value 3, ie. at the 4th position. While inserting a new element in between the existing elements, at a particular location, the existing elements shifts one position to the right.

9.1.8 Deleting elements from a list

There are two ways to delete an element from a list viz. **del** statement and **remove( )** function. del statement is used to delete known elements whereas remove( ) function is used to delete elements of a list if its index is unknown. The del statement can also be used to delete entire list.

Syntax:

del List [index of an element]

to delete a particular element

del List [index from : index to]

to delete multiple elements

del List

to delete entire list

Example

```
>>> MySubjects = ['Tamil', 'Hindi', 'Telugu', 'Maths']
>>> print (MySubjects)
        ['Tamil', 'Hindi', 'Telugu', 'Maths']
>>> del MySubjects[1]
>>> print (MySubjects)
        ['Tamil', 'Telugu', 'Maths']
```

In the above example, the list **MySubjects** has been created with four elements. print statement shows all the elements of the list. In >>> **del MySubjects[1]** statement, deletes an element whose index value is 1 and the following print shows the remaining elements of the list.

Example

```
>>> del MySubjects[1:3]
>>> print(MySubjects)
        ['Tamil']
```

In the above codes, >>> **del MySubjects[1:3]** deletes the second and third elements from the list. The upper limit of index is specified within square brackets, will be taken as -1 by the python.

Example

```
>>> del MySubjects
>>> print(MySubjects)
Traceback (most recent call last):
File "<pyshell#9>", line 1, in <module>
print(MySubjects)
NameError: name 'MySubjects' is not defined
```

Here, >>> del MySubjects, deletes the list MySubjects entirely. When you try to print the elements, Python shows an error as the list is not defined. Which means, the list MySubjects has been completely deleted.

As already stated, the remove( ) function can also be used to delete one or more elements if the index value is not known. Apart from remove( ) function, pop( ) function can also be used to delete an element using the given index value. pop( ) function deletes and returns the last element of a list if the index is not given.

The function clear( ) is used to delete all the elements in list, it deletes only the elements and retains the list. Remember that, the del statement deletes entire list.

Syntax:

List.remove(element) *# to delete a particular element*
List.pop(index of an element)
List.clear( )

Example

```
>>> MyList=[12,89,34,'Kannan', 'Gowrisankar', 'Lenin']
>>> print(MyList)
        [12, 89, 34, 'Kannan', 'Gowrisankar', 'Lenin']
>>> MyList.remove(89)
>>> print(MyList)
        [12, 34, 'Kannan', 'Gowrisankar', 'Lenin']
```

In the above example, MyList has been created with three integer and three string elements, the following print statement shows all the elements available in the list. In the statement >>> **MyList.remove(89)**, deletes the element 89 from the list and the print statement shows the remaining elements.

Example

```
>>> MyList.pop(1)
      34
>>> print(MyList)
      [12, 'Kannan', 'Gowrisankar', 'Lenin']
```

In the above code, pop( ) function is used to delete a particular element using its index value, as soon as the element is deleted, the pop( ) function shows the element which is deleted. pop( ) function is used to delete only one element from a list. Remember that, del statement deletes multiple elements.

Example

```
>>> MyList.clear( )
>>> print(MyList)
      [ ]
```

In the above code, clear( ) function removes only the elements and retains the list. When you try to print the list which is already cleared, an empty square bracket is displayed without any elements, which means the list is empty.

9.1.9 List and range ( ) function

The range( ) is a function used to generate a series of values in Python. Using range( ) function, you can create list with series of values. The range( ) function has three arguments.

Syntax of range ( ) function:

range (start value, end value, step value)

where,

- **start value** – beginning value of series. Zero is the default beginning value.
- **end value** – upper limit of series. Python takes the ending value as upper limit – 1.
- **step value** – It is an optional argument, which is used to generate different interval of values.

Example : Generating whole numbers upto 10

```
for x in range (1, 11):
    print(x)
```

Output

```
1
2
3
4
5
6
7
8
9
10
```

Example : Generating first 10 even numbers

```
for x in range (2, 11, 2):
    print(x)
```

Output

```
2
4
6
8
10
```

(i) Creating a list with series of values

Using the range( ) function, you can create a list with series of values. To convert the result of range( ) function into list, we need one more function called list( ). The list( ) function makes the result of range( ) as a list.

Syntax:

List_Varibale = list (range ( ))

Note

The list ( ) function is also used to create list in python.

Example

```
>>> Even_List = list(range(2,11,2))
>>> print(Even_List)
[2, 4, 6, 8, 10]
```

In the above code, list( ) function takes the result of range( ) as Even_List elements. Thus, Even_List list has the elements of first five even numbers.

Similarly, we can create any series of values using range( ) function. The following example explains how to create a list with squares of first 10 natural numbers.

Example : Generating squares of first 10 natural numbers

```
squares = [ ]
for x in range(1,11):
    s = x ** 2
    squares.append(s)
print (squares)
```

In the above program, an empty list is created named "squares". Then, the for loop generates natural numbers from 1 to 10 using range( ) function. Inside the loop, the current value of x is raised to the power 2 and stored in the variables. Each new value of square is appended to the list "squares". Finally, the program shows the following values as output.

Output

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

9.1.10 List comprehensions

List comprehension is a simplest way of creating sequence of elements that satisfy a certain condition.

Syntax:

List = [expression for variable in range]

Example : Generating squares of first 10 natural numbers using the concept of List comprehension

```
>>> squares = [ x ** 2 for x in range(1,11) ]
>>> print (squares)
```

Output:

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

In the above example, x ** 2 in the expression is evaluated each time it is iterated. This is the shortcut method of generating series of values.

9.1.11 Other important list funcion

Function	Description	Syntax	Example
copy ( )	Returns a copy of the list	List.copy( )	MyList=[12, 12, 36] x = MyList.copy() print(x) **Output:** [12, 12, 36]
count ( )	Returns the number of similar elements present in the last.	List.count(value)	MyList=[36 ,12 ,12] x = MyList.count(12) print(x) **Output:** 2
index ( )	Returns the index value of the first recurring element	List.index(element)	MyList=[36 ,12 ,12] x = MyList.index(12) print(x) **Output:** 1
reverse ( )	Reverses the order of the element in the list.	List.reverse( )	MyList=[36 ,23 ,12] MyList.reverse() print(MyList) **Output:** [12 ,23 ,36]
sort ( )	Sorts the element in list	List.sort(reverse=True|False, key=myFunc)	
Both arguments are optional • If reverse is set as True, list sorting is in descending order. • Ascending is default. • Key=myFunc; "myFunc" - the name of the user defined function that specifies the sorting criteria. **Note:** sort( ) will affect the original list.			MyList=['Thilothamma', 'Tharani', 'Anitha', 'SaiSree', 'Lavanya'] MyList.sort( ) print(MyList) MyList.sort(reverse=True) print(MyList) **Output:** ['Anitha', 'Lavanya', 'SaiSree', 'Tharani', 'Thilothamma'] ['Thilothamma', 'Tharani', 'SaiSree', 'Lavanya', 'Anitha']

max( )	Returns the maximum value in a list.	max(list)	MyList=[21,76,98,23] print(max(MyList)) **Output:** 98
min( )	Returns the minimum value in a list.	min(list)	MyList=[21,76,98,23] print(min(MyList)) **Output:** 21
sum( )	Returns the sum of values in a list.	sum(list)	MyList=[21,76,98,23] print(sum(MyList)) **Output:** 218

9.1.12 Programs using List

Program 1: write a program that creates a list of numbers from 1 to 20 that are divisible by 4

```
divBy4=[ ]
for i in range(21):
        if (i%4==0):
                divBy4.append(i)
print(divBy4)
```

Output

```
[0, 4, 8, 12, 16, 20]
```

Program 2: Write a program to define a list of countries that are a member of BRICS. Check whether a county is member of BRICS or not

```
country=["India", "Russia", "Srilanka", "China", "Brazil"]
is_member = input("Enter the name of the country: ")
if is_member in country:
        print(is_member, " is the member of BRICS")
else:
        print(is_member, " is not a member of BRICS")
```

Output

```
Enter the name of the country: India
India  is the member of BRICS
```

Output

```
Enter the name of the country: Japan
Japan  is not a member of BRICS
```

Program 3: Python program to read marks of six subjects and to print the marks scored in each subject and show the total marks

```
marks=[]
subjects=['Tamil', 'English', 'Physics', 'Chemistry', 'Comp. Science', 'Maths']
for i in range(6):
        m=int(input("Enter Mark = "))
        marks.append(m)
for j in range(len(marks)):
        print("{ }. { } Mark = { } ".format(j1+,subjects[j],marks[j]))
print("Total Marks = ", sum(marks))
```

Output

```
Enter Mark = 45
Enter Mark = 98
Enter Mark = 76
Enter Mark = 28
Enter Mark = 46
Enter Mark = 15
  1. Tamil Mark = 45
  2. English Mark = 98
  3. Physics Mark = 76
  4. Chemistry Mark = 28
  5. Comp. Science Mark = 46
  6. Maths Mark = 15
  Total Marks =  308
```

Program 4: Python program to read prices of 5 items in a list and then display sum of all the prices, product of all the prices and find the average

```
items=[]
prod=1
for i in range(5):
        print ("Enter price for item { } : ".format(i+1))
        p=int(input())
        items.append(p)
for j in range(len(items)):
        print("Price for item { } = Rs. { }".format(j+1,items[j]))
        prod = prod * items[j]
print("Sum of all prices = Rs.", sum(items))
print("Product of all prices = Rs.", prod)
print("Average of all prices = Rs.",sum(items)/len(items))
```

Output:

```
Enter price for item  1 :
5
Enter price for item  2 :
10
Enter price for item  3 :
15
Enter price for item  4 :
20
Enter price for item  5 :
25
Price for item  1 = Rs. 5
Price for item  2 = Rs. 10
Price for item  3 = Rs. 15
Price for item  4 = Rs. 20
Price for item  5 = Rs. 25
Sum of all prices = Rs. 75
Product of all prices = Rs. 375000
Average of all prices = Rs. 15.0
```

Program 5: Python program to count the number of employees earning more than 1 lakh per annum. The monthly salaries of n number of employees are given

```
count=0
n=int(input("Enter no. of employees: "))
print("No. of Employees",n)
salary=[]
for i in range(n):
    print("Enter Monthly Salary of Employee { } Rs.: ".format(i+1))
    s=int(input())
    salary.append(s)
for j in range(len(salary)):
    annual_salary = salary[j] * 12
    print ("Annual Salary of Employee { } is:Rs. { }".format(j+1,annual_salary))
    if annual_salary >= 100000:
        count = count + 1
print("{ } Employees out of { } employees are earning more than Rs. 1 Lakh per annum".
format(count, n))
```

Output:

```
Enter no. of employees: 5
No. of Employees 5
Enter Monthly Salary of Employee 1 Rs.:
3000
Enter Monthly Salary of Employee 2 Rs.:
9500
Enter Monthly Salary of Employee 3 Rs.:
12500
Enter Monthly Salary of Employee 4 Rs.:
5750
Enter Monthly Salary of Employee 5 Rs.:
8000
Annual Salary of Employee 1 is:Rs. 36000
Annual Salary of Employee 2 is:Rs. 114000
Annual Salary of Employee 3 is:Rs. 150000
Annual Salary of Employee 4 is:Rs. 69000
Annual Salary of Employee 5 is:Rs. 96000
2 Employees out of 5 employees are earning more than Rs. 1 Lakh per annum
```

Program 6: Write a program to create a list of numbers in the range 1 to 10. Then delete all the even numbers from the list and print the final list.

```
Num = []
for x in range(1,11):
    Num.append(x)
print("The list of numbers from 1 to 10 = ", Num)

for index, i in enumerate(Num):
    if(i%2==0):
        del Num[index]
print("The list after deleting even numbers = ", Num)
```

Output

```
The list of numbers from 1 to 10 =  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
The list after deleting even numbers =  [1, 3, 5, 7, 9]
```

Program 7: Write a program to generate in the Fibonacci series and store it in a list. Then find the sum of all values.

```
a=-1
b=1
n=int(input("Enter no. of terms: "))
i=0
sum=0
Fibo=[]
while i<n:
        s = a + b
        Fibo.append(s)
        sum+=s
        a = b
        b = s
        i+=1
print("Fibonacci series upto "+ str(n) +" terms is : " + str(Fibo))
print("The sum of Fibonacci series: ",sum)
```

Output

```
Enter no. of terms: 10
Fibonacci series upto 10 terms is : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
The sum of Fibonacci series:  88
```

9.2 Tuples

Introduction to Tuples

Tuples consists of a number of values separated by comma and enclosed within parentheses. Tuple is similar to list, values in a list can be changed but not in a tuple.

DO YOU KNOW?

The term Tuple is originated from the Latin word represents an abstraction of the sequence of numbers:

single(1), double(2), triple(3), quadruple(4), quintuple(5), sextuple(6), septuple(7), octuple(8), ..., n-tuple, ...,

9.2.1 Advantages of Tuples over list

1. The elements of a list are changeable (mutable) whereas the elements of a tuple are unchangeable (immutable), this is the key difference between tuples and list.
2. The elements of a list are enclosed within square brackets. But, the elements of a tuple are enclosed by paranthesis.
3. Iterating tuples is faster than list.

9.2.2 Creating Tuples

Creating tuples is similar to list. In a list, elements are defined within square brackets, whereas in tuples, they may be enclosed by parenthesis. The elements of a tuple can be even defined without parenthesis. Whether the elements defined within parenthesis or without parenthesis, there is no differente in it's function.

Syntax:

 # Empty tuple

Tuple_Name = ( )

 # Tuple with n number elements

Tuple_Name = (E1, E2, E2 ……. En)

 # Elements of a tuple without parenthesis

Tuple_Name = E1, E2, E3 ….. En

Example

```
>>> MyTup1 = (23, 56, 89, 'A', 'E', 'I', "Tamil")
>>> print(MyTup1)
        (23, 56, 89, 'A', 'E', 'I', 'Tamil')

>>> MyTup2 = 23, 56, 89, 'A', 'E', 'I', "Tamil"
>>> print (MyTup2)
        (23, 56, 89, 'A', 'E', 'I', 'Tamil')
```

(i) Creating tuples using tuple( ) function

The tuple( ) function is used to create Tuples from a list. When you create a tuple, from a list, the elements should be enclosed within square brackets.

Syntax:

Tuple_Name = tuple([list elements])

Example

```
>>> MyTup3 = tuple( [23, 45, 90] )
>>> print(MyTup3)
        (23, 45, 90)
>>> type (MyTup3)
        <class 'tuple'>
```

Note

Type ( ) function is used to know the data type of a python object.

(ii) Creating Single element tuple

While creating a tuple with a single element, add a comma at the end of the element. In the absence of a comma, Python will consider the element as an ordinary data type; not a tuple. Creating a Tuple with one element is called "Singleton" tuple.

Example

```
>>> MyTup4 = (10)
>>> type(MyTup4)
        <class 'int'>
>>> MyTup5 = (10,)
>>> type(MyTup5)
        <class 'tuple'>
```

9.2.3 Accessing values in a Tuple

Like list, each element of tuple has an index number starting from zero. The elements of a tuple can be easily accessed by using index number.

Example

```
>>> Tup1 = (12, 78, 91, "Tamil", "Telugu", 3.14, 69.48)
# to access all the elements of a tuple
>>> print(Tup1)
        (12, 78, 91, 'Tamil', 'Telugu', 3.14, 69.48)
#accessing selected elements using indices
>>> print(Tup1[2:5])
        (91, 'Tamil', 'Telugu')
#accessing from the first element up to the specified index value
>>> print(Tup1[:5])
        (12, 78, 91, 'Tamil', 'Telugu')
# accessing from the specified element up to the last element.
>>> print(Tup1[4:])
        ('Telugu', 3.14, 69.48)
# accessing from the first element to the last element
>>> print(Tup1[:])
        (12, 78, 91, 'Tamil', 'Telugu', 3.14, 69.48)
```

9.2.4 Update and Delete Tuple

As you know a tuple is immutable, the elements in a tuple cannot be changed. Instead of altering values in a tuple, joining two tuples or deleting the entire tuple is possible.

Example

```
# Program to join two tuples
Tup1 = (2,4,6,8,10)
Tup2 = (1,3,5,7,9)
Tup3 = Tup1 + Tup2
print(Tup3)
```

Output

```
(2, 4, 6, 8, 10, 1, 3, 5, 7, 9)
```

To delete an entire tuple, the del command can be used.

Syntax:

del tuple_name

Example

```
Tup1 = (2,4,6,8,10)
print("The elements of Tup1 is ", Tup1)
del Tup1
print (Tup1)
```

Output:

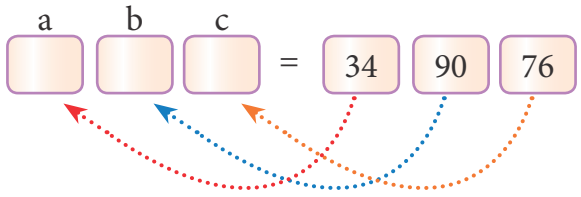
```
The elements of Tup1 is  (2, 4, 6, 8, 10)
Traceback (most recent call last):
File "D:/Python/Tuple Examp 1.py", line 4, in <module>
print (Tup1)
NameError: name 'Tup1' is not defined
```

Note that, the print statement in the above code prints the elements. Then, the del statement deletes the entire tuple. When you try to print the deleted tuple, Python shows the error.

9.2.5 Tuple Assignment

Tuple assignment is a powerful feature in Python. It allows a tuple variable on the left of the assignment operator to be assigned to the values on the right side of the assignment

operator. Each value is assigned to its respective variable.



Example

```
>>> (a, b, c) = (34, 90, 76)
>>> print(a,b,c)
        34 90 76
# expression are evaluated before assignment
>>> (x, y, z, p) = (2**2, 5/3+4, 15%2, 34>65)
>>> print(x,y,z,p)
        4 5.666666666666667 1 False
```

Note that, when you assign values to a tuple, ensure that the number of values on both sides of the assignment operator are same; otherwise, an error is generated by Python.

9.2.6 Returning multiple values in Tuples

A function can return only one value at a time, but Python returns more than one value from a function. Python groups multiple values and returns them together.

Example : Program to return the maximum as well as minimum values in a list

```
def Min_Max(n):
    a = max(n)
    b = min(n)
    return(a, b)
Num = (12, 65, 84, 1, 18, 85, 99)
(Max_Num, Min_Num) = Min_Max(Num)
print("Maximum value = ", Max_Num)
print("Minimum value = ", Min_Num)
```

Output:

```
Maximum value =  99
Minimum value =  1
```

9.2.7 Nested Tuples

In Python, a tuple can be defined inside another tuple; called Nested tuple. In a nested tuple, each tuple is considered as an element. The for loop will be useful to access all the elements in a nested tuple.

Example

```
Toppers = (("Vinodini", "XII-F", 98.7), ("Soundarya", "XII-H", 97.5),
                ("Tharani", "XII-F", 95.3), ("Saisri", "XII-G", 93.8))
for i in Toppers:
        print(i)
```

Output:

```
('Vinodini', 'XII-F', 98.7)
('Soundarya', 'XII-H', 97.5)
('Tharani', 'XII-F', 95.3)
('Saisri', 'XII-G', 93.8)
```

Note

All the functions used in List can be applicable even for tuples.

9.2.8 Programs using Tuples

Program 1: Write a program to swap two values using tuple assignment

```
a = int(input("Enter value of A: "))
b = int(input("Enter value of B: "))
print("Value of A = ", a, "\n Value of B = ", b)
(a, b) = (b, a)
print("Value of A = ", a, "\n Value of B = ", b)
```

Output:

```
Enter value of A: 54
Enter value of B: 38
Value of A =  54
 Value of B =  38
Value of A =  38
 Value of B =  54
```

Program 2: Write a program using a function that returns the area and circumference of a circle whose radius is passed as an argument.two values using tuple assignment

```
pi = 3.14
def Circle(r):
    return (pi*r*r, 2*pi*r)
radius = float(input("Enter the Radius: "))
(area, circum) = Circle(radius)
print ("Area of the circle = ", area)
print ("Circumference of the circle = ", circum)
```

Output:

```
Enter the Radius: 5
Area of the circle =  78.5
Circumference of the circle =  31.400000000000002
```

Program 3: Write a program that has a list of positive and negative numbers. Create a new tuple that has only positive numbers from the list

```
Numbers = (5, -8, 6, 8, -4, 3, 1)
Positive = ( )
for i in Numbers:
    if i > 0:
        Positive += (i, )
print("Positive Numbers: ", Positive)
```

Output:

```
Positive Numbers:  (5, 6, 8, 3, 1)
```

9.3 Sets

Introduction

In python, a set is another type of collection data type. A Set is a mutable and an unordered collection of elements without duplicates. That means the elements within a set cannot be repeated. This feature used to include membership testing and eliminating duplicate elements.

9.3.1 Creating a Set

A set is created by placing all the elements separated by comma within a pair of curly brackets. The set( ) function can also used to create sets in Python.

Syntax:

Set_Variable = {E1, E2, E3 …….. En}

G6MP5L

Example

```
>>> S1={1,2,3,'A',3.14}
>>> print(S1)
        {1, 2, 3, 3.14, 'A'}

>>> S2={1,2,2,'A',3.14}
>>> print(S2)
        {1, 2, 'A', 3.14}
```

In the above examples, the set S1 is created with different types of elements without duplicate values. Whereas in the set S2 is created with duplicate values, but python accepts only one element among the duplications. Which means python removed the duplicate value, because a set in python cannot have duplicate elements.

Note

When you print the elements from a set, python shows the values in different order.

9.3.2 Creating Set using List or Tuple

A list or Tuple can be converted as set by using set( ) function. This is very simple procedure. First you have to create a list or Tuple then, substitute its variable within set( ) function as argument.

Example

```
MyList=[2,4,6,8,10]
MySet=set(MyList)
print(MySet)
```

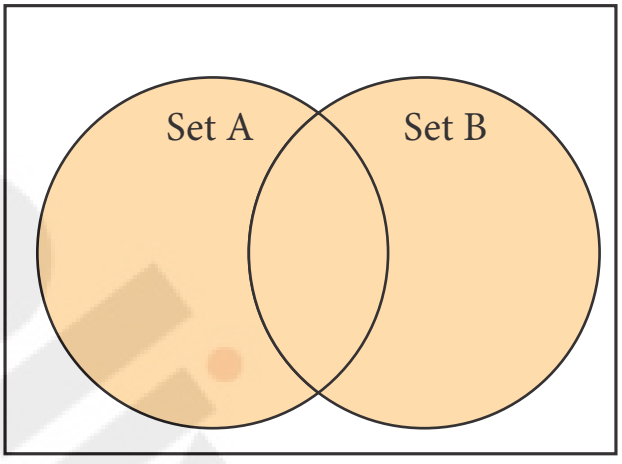
Output:

```
{2, 4, 6, 8, 10}
```

9.3.3 Set Operations

As you learnt in mathematics, the python is also supports the set operations such as Union, Intersection, difference and Symmetric difference.

(i) Union: It includes all elements from two or more sets



In python, the operator | is used to union of two sets. The function union( ) is also used to join two sets in python.

Example: Program to Join (Union) two sets using union operator

```
set_A={2,4,6,8}
set_B={'A', 'B', 'C', 'D'}
U_set=set_A|set_B
print(U_set)
```

Output:

{2, 4, 6, 8, 'A', 'D', 'C', 'B'}

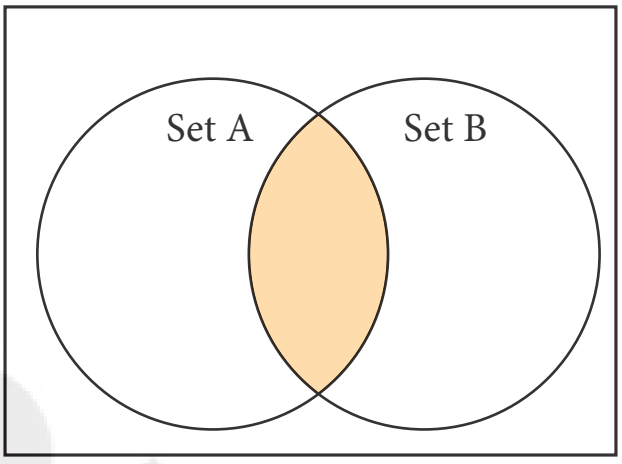
Example: Program to Join (Union) two sets using union function

```
set_A={2,4,6,8}
set_B={'A', 'B', 'C', 'D'}
set_U=set_A.union(set_B)
print(set_U)
```

Output:

{'D', 2, 4, 6, 8, 'B', 'C', 'A'}

(ii) Intersection: It includes the common elements in two sets



The operator & is used to intersect two sets in python. The function **intersection( )** is also used to intersect two sets in python.

Example: Program to insect two sets using intersection operator

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A & set_B)
```

Output:

```
{'A', 'D'}
```

Example: Program to insect two sets using intersection function

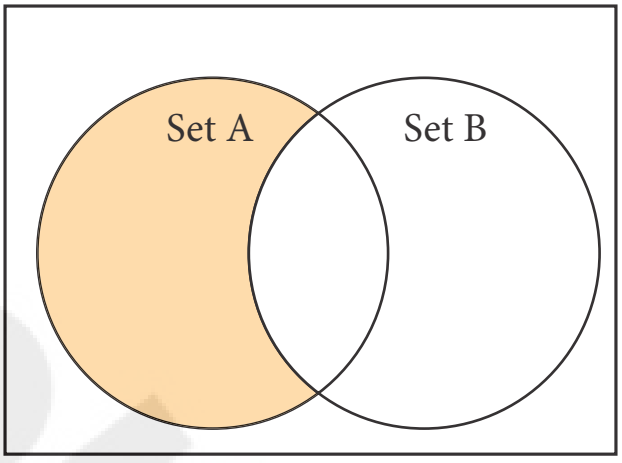
```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.intersection(set_B))
```

Output:

```
{'A', 'D'}
```

(iii) Difference

It includes all elements that are in first set (say set A) but not in the second set (say set B)



The minus (-) operator is used to difference set operation in python. The function **difference( )** is also used to difference operation.

Example: Program to difference of two sets using minus operator

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A - set_B)
```

Output:

```
{2, 4}
```

Example: Program to difference of two sets using difference function

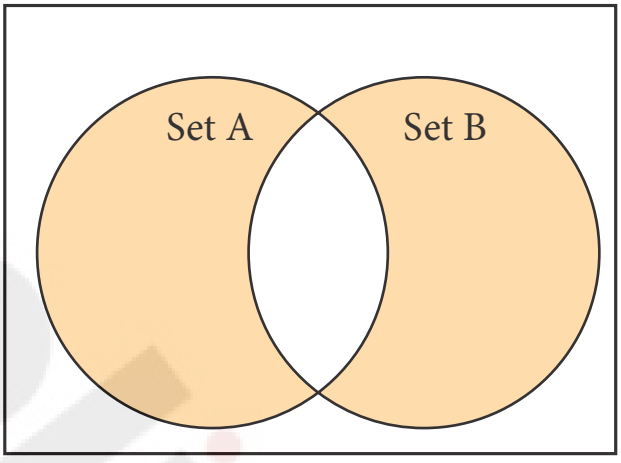
```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.difference(set_B))
```

Output:

```
{2, 4}
```

(iv) Symmetric difference

It includes all the elements that are in two sets (say sets A and B) but not the one that are common to two sets.



The caret (^) operator is used to symmetric difference set operation in python. The function **symmetric_difference( )** is also used to do the same operation.

Example: Program to symmetric difference of two sets using caret operator

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A ^ set_B)
```

Output:

```
{2, 4, 'B', 'C'}
```

Example: Program to difference of two sets using symmetric difference function

```
set_A={'A', 2, 4, 'D'}
set_B={'A', 'B', 'C', 'D'}
print(set_A.symmetric_difference(set_B))
```

Output:

```
{2, 4, 'B', 'C'}
```

9.3.4 Programs using Sets

Program 1: Program that generate a set of prime numbers and another set of even numbers. Demonstrate the result of union, intersection, difference and symmetirc difference operations.

Example

```
even=set([x*2 for x in range(1,11)])
primes=set()
for i in range(2,20):
        j=2
        f=0
        while j<=i/2:
                if i%j==0:
                        f=1
                j+=1
        if f==0:
                primes.add(i)
print("Even Numbers: ", even)
print("Prime Numbers: ", primes)
print("Union: ", even.union(primes))
print("Intersection: ", even.intersection(primes))
print("Difference: ", even.difference(primes))
print("Symmetric Difference: ", even.symmetric_difference(primes))
```

Output:

```
Even Numbers:  {2, 4, 6, 8, 10, 12, 14, 16, 18, 20}
Prime Numbers:  {2, 3, 5, 7, 11, 13, 17, 19}
Union:  {2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20}
Intersection:  {2, 4}
Difference:  {6, 8, 10, 12, 14, 16, 18, 20}
Symmetric Difference:  {3, 5, 6, 7, 8, 10, 11, 12, 13, 14, 16, 17, 18, 19, 20}
```

9.4 Dictionaries

Introduction

In python, a dictionary is a mixed collection of elements. Unlike other collection data types such as a list or tuple, the dictionary type stores a key along with its element. The keys in a Python dictionary is separated by a colon (:) while the commas work as a separator for the elements. The key value pairs are enclosed with curly braces { }.

Syntax of defining a dictionary:

```
Dictionary_Name = {        Key_1: Value_1,
                           Key_2:Value_2,
                           ........
                           Key_n:Value_n
                           }
```

Key in the dictionary must be unique case sensitive and can be of any valid Python type.

9.4.1 Creating a Dictionary

 # Empty dictionary

Dict1 = { }

 # Dictionary with Key

Dict_Stud = { 'RollNo': '1234', 'Name':'Murali', 'Class':'XII', 'Marks':'451'}

9.4.2 Dictionary Comprehensions

In Python, comprehension is another way of creating dictionary. The following is the syntax of creating such dictionary.

Syntax

Dict = { expression for variable in sequence [if condition] }

The if condition is optional and if specified, only those values in the sequence are evaluated using the expression which satisfy the condition.

Example

*Dict = { x : 2 * x for x in range(1,10)}*

Output of the above code is

{1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18}

9.4.3 Accessing, Adding, Modifying and Deleting elements from a Dictionary

Accessing all elements from a dictionary is very similar as Lists and Tuples. Simple print function is used to access all the elements. If you want to access a particular element, square brackets can be used along with key.

Example : Program to access all the values stored in a dictionary

```
MyDict = {      'Reg_No': '1221',
                'Name' : 'Tamilselvi',
                'School' : 'CGHSS',
                'Address' : 'Rotler St., Chennai 112' }
print(MyDict)
print("Register Number: ", MyDict['Reg_No'])
print("Name of the Student: ", MyDict['Name'])
print("School: ", MyDict['School'])
print("Address: ", MyDict['Address'])
```

Output:

```
{'Reg_No': '1221', 'Name': 'Tamilselvi', 'School': 'CGHSS', 'Address': 'Rotler St., Chennai 112'}
Register Number:  1221
Name of the Student:  Tamilselvi
School:  CGHSS
Address:  Rotler St., Chennai 112
```

Note that, the first print statement prints all the values of the dictionary. Other statements are printing only the specified values which is given within square brackets.

In an existing dictionary, you can add more values by simply assigning the value along with key. The following syntax is used to understand adding more elements in a dictionary.

dictionary_name [key] = value/element

Example : Program to add a new value in the dictionary

```
MyDict = {      'Reg_No': '1221',
                'Name' : 'Tamilselvi',
                'School' : 'CGHSS', 'Address' : '
                Rotler St., Chennai 112'}
print(MyDict)
print("Register Number: ", MyDict['Reg_No'])
print("Name of the Student: ", MyDict['Name'])
MyDict['Class'] = 'XII - A'          # Adding new value
print("Class: ", MyDict['Class'])    # Printing newly added value
print("School: ", MyDict['School'])
print("Address: ", MyDict['Address'])
```

Modification of a value in dictionary is very similar as adding elements. When you assign a value to a key, it will simply overwrite the old value.

In Python dictionary, del keyword is used to delete a particular element. The clear( ) function is used to delete all the elements in a dictionary. To remove the dictionary, you can use del keyword with dictionary name.

Syntax:

```
# To delete a particular element.
del dictionary_name[key]
# To delete all the elements
dictionary_name.clear( )
# To delete an entire dictionary
del dictionary_name
```

Example : Program to delete elements from a dictionary and finally deletes the dictionary.

```
Dict = {'Roll No' : 12001, 'SName' : 'Meena', 'Mark1' : 98, 'Marl2' : 86}
print("Dictionary elements before deletion: \n", Dict)
del Dict['Mark1']          # Deleting a particular element
print("Dictionary elements after deletion of a element: \n", Dict)
Dict.clear()               # Deleting all elements
print("Dictionary after deletion of all elements: \n", Dict)
del Dict
print(Dict)                # Deleting entire dictionary
```

Output:

```
Dictionary elements before deletion:
    {'Roll No': 12001, 'SName': 'Meena', 'Mark1': 98, 'Marl2': 86}
Dictionary elements after deletion of a element:
    {'Roll No': 12001, 'SName': 'Meena', 'Marl2': 86}
Dictionary after deletion of all elements:
    { }
Traceback (most recent call last):
    File "E:/Python/Dict_Test_02.py", line 8, in <module>
    print(Dict)
NameError: name 'Dict' is not defined
```

9.4.4 Difference between List and Dictionary

(1) List is an ordered set of elements. But, a dictionary is a data structure that is used for matching one element (Key) with another (Value).

(2) The index values can be used to access a particular element. But, in dictionary key represents index. Remember that, key may be a number of a string.

(3) Lists are used to look up a value whereas a dictionary is used to take one value and look up another value.

Points to remember:

- Python programming language has four collections of data types such as List, Tuple, Set and Dictionary.
- A list is known as a "sequence data type". Each value of a list is called as element.
- The elements of list should be specified within square brackets.
- Each element has a unique value called index number begins with zero.
- Python allows positive and negative values as index.
- Loops are used access all elements from a list.
- The "for" loop is a suitable loop to access all the elements one by one.
- The append ( ), extend ( ) and insert ( ) functions are used to include more elements in a List.
- The del, remove ( ) and pop ( ) are used to delete elements from a list.
- The range ( ) function is used to generate a series of values.
- Tuples consists of a number of values separated by comma and enclosed within parentheses.
- Iterating tuples is faster than list.
- The tuple ( ) function is also used to create Tuples from a list.
- Creating a Tuple with one element is called "Singleton" tuple.
- A Set is a mutable and an unordered collection of elements without duplicates.
- A set is created by placing all the elements separated by comma within a pair of curly brackets.
- A dictionary is a mixed collection of elements.

Hands on Experience

1. Write a program to remove duplicates from a list.
2. Write a program that prints the maximum value in a Tuple.
3. Write a program that finds the sum of all the numbers in a Tuples using while loop.
4. Write a program that finds sum of all even numbers in a list.

5. Write a program that reverse a list using a loop.

6. Write a program to insert a value in a list at the specified location.

7. Write a program that creates a list of numbers from 1 to 50 that are either divisible by 3 or divisible by 6.

8. Write a program to create a list of numbers in the range 1 to 20. Then delete all the numbers from the list that are divisible by 3.

9. Write a program that counts the number of times a value appears in the list. Use a loop to do the same.

10. Write a program that prints the maximum and minimum value in a dictionary.

Evaluation

GL6J6X

Part - I

Choose the best answer **(1 Marks)**

1. Pick odd one in connection with collection data type

   (a) List (b) Tuple (c) Dictionary (d) Loop

2. Let list1=[2,4,6,8,10], then print(List1[-2]) will result in

   (a) 10 (b) 8 (c) 4 (d) 6

3. Which of the following function is used to count the number of elements in a list?

   (a) count() (b) find() (c)len() (d) index()

4. If List=[10,20,30,40,50] then List[2]=35 will result

   (a) [35,10,20,30,40,50] (b) [10,20,30,40,50,35]

   (c) [10,20,35,40,50] (d) [10,35,30,40,50]

5. If List=[17,23,41,10] then List.append(32) will result

   (a) [32,17,23,41,10] (b) [17,23,41,10,32]

   (c) [10,17,23,32,41] (d) [41,32,23,17,10]

6. Which of the following Python function can be used to add more than one element within an existing list?

   (a) append() (b) append_more() (c)extend() (d) more()

7. What will be the result of the following Python code?

   S=[x**2 for x in range(5)]

```
print(S)
```

(a) [0,1,2,4,5] (b) [0,1,4,9,16] (c) [0,1,4,9,16,25] (d) [1,4,9,16,25]

8. What is the use of type() function in python?

   (a) To create a Tuple

   (b) To know the type of an element in tuple.

   (c) To know the data type of python object.

   (d) To create a list.

9. Which of the following statement is not correct?

   (a) A list is mutable

   (b) A tuple is immutable.

   (c) The append() function is used to add an element.

   (d) The extend() function is used in tuple to add elements in a list.

10. Let setA={3,6,9}, setB={1,3,9}. What will be the result of the following snippet?

```
print(setA|setB)
```

(a) {3,6,9,1,3,9} (b) {3,9} (c) {1} (d) {1,3,6,9}

11. Which of the following set operation includes all the elements that are in two sets but not the one that are common to two sets?

   (a) Symmetric difference (b) Difference

   (c) Intersection (d) Union

12. The keys in Python, dictionary is specified by

   (a) = (b) ; (c)+ (d) :

Part - II

Answer the following questions (2 Marks)

1. What is List in Python?
2. How will you access the list elements in reverse order?
3. What will be the value of x in following python code?

```
List1=[2,4,6[1,3,5]]
x=len(List1)
```

4. Differentiate del with remove( ) function of List.

5. Write the syntax of creating a Tuple with n number of elements.
6. What is set in Python?

Part - III

Answer the following questions (3 Marks)

1. What are the advantages of Tuples over a list?
2. Write a shot note about sort( ).
3. What will be the output of the following code?

```
list = [2**x for x in range(5)]
print(list)
```

4. Explain the difference between del and clear( ) in dictionary with an example.
5. List out the set operations supported by python.
6. What are the difference between List and Dictionary?

Part - IV

Answer the following questions (5 Marks)

1. What the different ways to insert an element in a list. Explain with suitable example.
2. What is the purpose of range( )? Explain with an example.
3. What is nested tuple? Explain with an example.
4. Explain the different set operations supported by python with suitable example.

References

1. https://docs.python.org/3/tutorial/index.html
2. https://www.techbeamers.com/python-tutorial-step-by-step/#tutorial-list
3. Python programming using problem solving approach – Reema Thareja – Oxford University press.
4. Python Crash Course – Eric Matthes – No starch press, San Francisco.

Unit III

CHAPTER 10

PYTHON CLASSES AND OBJECTS

HFYD7X

Learning Objectives

After the completion of this chapter, the student is able to

- Understand the fundamental concepts of Object Oriented Programming like: Classes, Objects, Constructor and Destructor.
- Gain the knowledge of creating classes and objects in Python.
- Create classes with Constructors.
- Write complex programs in Python using classes.

10.1 Introduction

Python is an Object Oriented Programming language. Classes and Objects are the key features of Object Oriented Programming. Theoretical concepts of classes and objects are very similar to that of C++. But, creation and implementation of classes and objects is very simple in Python compared to C++.

Class is the main building block in Python. Object is a collection of data and function that act on those data. Class is a template for the object. According to the concept of Object Oriented Programming, objects are also called as instances of a class or class variable. In Python, everything is an object. For example, all integer variables that we use in our program is an object of class int. Similarly all string variables are also object of class string.

10.2 Defining classes

In Python, a class is defined by using the keyword class. Every class has a unique name followed by a colon (:).

Syntax:

```
class class_name:
    statement_1
    statement_2
    ……..
    ……..
    statement_n
```


XII Std - CS EM Chapter-10.indd 170 24-01-2020 12:42:35

Where, statement in a class definition may be a variable declaration, decision control, loop or even a function definition. Variables defined inside a class are called as "Class Variable" and functions are called as "Methods". Class variable and methods are together known as members of the class. The class members should be accessed through objects or instance of class. A class can be defined anywhere in a Python program.

Example: Program to define a class

```
class Sample:
        x, y = 10, 20          # class variables
```

In the above code, name of the class is Sample and it has two variables x and y having the initial value 10 and 20 respectively. To access the values defined inside the class, you need an object or instance of the class.

10.3 Creating Objects

Once a class is created, next you should create an object or instance of that class. The process of creating object is called as "Class Instantiation".

HVH899

Syntax:

Object_name = class_name( )

Note that the class instantiation uses function notation ie. class_name with

10.4 Accessing Class Members

Any class member ie. class variable or method (function) can be accessed by using object with a dot (.) operator.

Syntax:

Object_name . class_member

Example : Program to define a class and access its member variables

```
class Sample:
        x, y = 10, 20          #class variables
S=Sample( )          # class instantiation
print("Value of x = ", S.x)
print("Value of y = ", S.y)
print("Value of x and y = ", S.x+S.y)
```

Output :

```
Value of x =  10
Value of y =  20
Value of x and y =  30
```

In the above code, the name of the class is Sample. Inside the class, we have assigned the variables **x** and **y** with initial value **10** and **20** respectively. These two variables are called as class variables or member variables of the class. In class instantiation process, we have created an object **S** to access the members of the class. The first two print statements simply print the value of class variable **x** and **y** and the last print statement add the two values and print the result.

10.5 Class Methods

Python class function or Method is very similar to ordinary function with a small difference that, the class method must have the first argument named as **self**. No need to pass a value for this argument when we call the method. Python provides its value automatically. Even if a method takes no arguments, it should be defined with the first argument called **self**. If a method is defined to accept only one argument it will take it as two arguments ie. self and the defined argument.

When you declare class variable within class, methods must be prefixed by the class name and dot operator.

Note

The statements defined inside the class must be properly indented.

Example 10.2: Program to find total and average marks using class

```
class Student:
        mark1, mark2, mark3 = 45, 91, 71          #class variable

        def process(self):                        #class method
            sum = Student.mark1 + Student.mark2 + Student.mark3
            avg = sum/3
            print("Total Marks = ", sum)
            print("Average Marks = ", avg)
            return

S=Student()
S.process()
```

In the above program, after defining the class, an object S is created. The statement S.process( ), calls the function to get the required output.

XII Std - CS EM Chapter-10.indd 172 24-01-2020 12:42:35

Note that, we have declared three variables mark1, mark2 and mark3 with the values 45, 91, 71 respectively. We have defined a method named **process** with **self** argument, which means, we are not going to pass any value to that method. First, the process method adds the values of the class variables, stores the result in the variable sum, finds the average and displays the result.

Thus the above code will show the following output.

Output

Total Marks = 207

Average Marks = 69.0

Example : program to check and print if the given number is odd or even using class

```
class Odd_Even:
        def check(self, num):
                if num%2==0:
                print(num," is Even number")
        else:
                print(num," is Odd number")
n=Odd_Even()
x = int(input("Enter a value: "))
n.check(x)
```

When you execute this program, Python accepts the value entered by the user and passes it to the method check through object.

Output 1

```
Enter a value: 4
4  is Even number
```

Output 2

```
Enter a value: 5
5  is Odd number
```

10.6 Constructor and Destructor in Python

Constructor is the special function that is automatically executed when an object of a class is created. In Python, there is a special function called **"init"** which act as a Constructor. It must begin and end with double underscore. This function will act as an ordinary function; but only difference is, it is executed automatically when the object is created. This constructor function can be defined with or without arguments. This method is used to initialize the class variables.

XII Std - CS EM Chapter-10.indd 173 24-01-2020 12:42:35

General format of __init__ method (Constructor function)

def __init__(self, [args .....]):

<statements>

Example : Program to illustrate Constructor

```
class Sample:
        def __init__(self, num):
                print("Constructor of class Sample...")
                self.num=num
                print("The value is :", num)
S=Sample(10)
```

The above class "Sample", has only a constructor with one argument named as num. When the constructor gets executed, first the print statement, prints the "Constructor of class Sample….", then, the passing value to the constructor is assigned to self.num and finally it prints the value passed along with the given string.

The above constructor gets executed automatically, when an object **S** is created with actual parameter 10. Thus, the Python display the following output.

Constructor of class Sample...

The value is : 10

Class variable defined within constructor keep count of number of objects created with the class.

Example : Program to illustrate class variable to keep count of number of objects created.

```
class Sample:
        num=0
        def __init__(self, var):
                Sample.num+=1
                self.var=var
                print("The object value is = ", var)
                print("The count of object created = ", Sample.num)

S1=Sample(15)
S2=Sample(35)
S3=Sample(45)
```


XII Std - CS EM Chapter-10.indd 174 24-01-2020 12:42:36

In the above program, class variable **num** is shared by all three objects of the class Sample. It is initialized to zero and each time an object is created, the num is incremented by 1. Since, the variable shared by all objects, change made to num by one object is reflected in other objects as well. Thus the above program produces the output given below.

Output

```
The object value is          =    15
The count of object created  =    1
The object value is          =    35
The count of object created  =    2
The object value is          =    45
The count of object created  =    3
```

***Note:* class variable is similar to static type in C++**

Destructor is also a special method gets executed automatically when an object exit from the scope. It is just opposite to constructor. In Python, __del__( ) method is used as destructor.

Example : Program to illustrate about the __del__( ) method

```
class Sample:
        num=0
        def __init__(self, var):
                Sample.num+=1
                self.var=var
                print("The object value is = ", var)
                print("The value of class variable is= ", Sample.num)
        def __del__(self):
                Sample.num-=1
                print("Object with value %d is exit from the scope"%self.var)
S1=Sample(15)
S2=Sample(35)
S3=Sample(45)
```

10.7 Public and Private Data Members

The variables which are defined inside the class is public by default. These variables can be accessed anywhere in the program using dot operator.

A variable prefixed with double underscore becomes private in nature. These variables can be accessed only within the class.

XII Std - CS EM Chapter-10.indd 175 24-01-2020 12:42:36

Example : Program to illustrate private and public variables

```
class Sample:
        def __init__(self, n1, n2):
                self.n1=n1
                self.__n2=n2
        def display(self):
                print("Class variable 1 = ", self.n1)
                print("Class variable 2 = ", self.__n2)
S=Sample(12, 14)
S.display()
print("Value 1 = ", S.n1)
print("Value 2 = ", S.__n2)
```

In the above program, there are two class variables n1 and n2 are declared. The variable n1 is a public variable and n2 is a private variable. The display( ) member method is defined to show the values passed to these two variables.

The print statements defined within class will successfully display the values of n1 and n2, even though the class variable n2 is private. Because, in this case, n2 is called by a method defined inside the class. But, when we try to access the value of n2 from outside the class Python throws an error. Because, private variable cannot be accessed from outside the class.

Output

```
Class variable 1 =  12
Class variable 2 =  14
Value 1 =  12

Traceback (most recent call last):
        File "D:/Python/Class-Test-04.py", line 12, in <module>
                print("Value 2 = ", S.__n2)
AttributeError: 'Sample' object has no attribute '__n2'
```


XII Std - CS EM Chapter-10.indd 176 24-01-2020 12:42:36

10.8 Sample Programs to illustrate classes and objects

Program 1: Write a program to calculate area and circumference of a circle

```
class Circle:
        pi=3.14
def __init__(self,radius):
        self.radius=radius
def area(self):
        return Circle.pi*(self.radius**2)
def circumference(self):
        return 2*Circle.pi*self.radius
r=int(input("Enter Radius: "))
C=Circle(r)
print("The Area =",C.area())
print("The Circumference =", C.circumference())
```

Output:

```
Enter Radius: 5
The Area = 78.5
The Circumference = 31.400000000000002
```

Program 2: write a menu driven program that keeps record of books available in you school library

```
class Library:
        def __init__(self):
                self.bookname=""
                self.author=""

        def getdata(self):
                self.bookname = input("Enter Name of the Book: ")
                self.author = input("Enter Author of the Book: ")

        def display(self):
                print("Name of the Book: ",self.bookname)
                print("Author of the Book: ",self.author)
                print("\n")
```


XII Std - CS EM Chapter-10.indd 177 24-01-2020 12:42:36

```
book=[] #empty list
ch = 'y'
while(ch=='y'):
        print("1. Add New Book \n 2.Display Books")
        resp = int(input("Enter your choice : "))

if(resp==1):
        L=Library()
        L.getdata()
        book.append(L)
elif(resp==2):
        for x in book:
                x.display()

else:
        print("Invalid input....")

ch = input("Do you want continue....")
```

Output:

```
1. Add New Book
2.Display Books
Enter your choice : 1
Enter Name of the Book: Programming in C++
Enter Author of the Book: K. Kannan
Do you want continue....y
1. Add New Book
2.Display Books
Enter your choice : 1
Enter Name of the Book: Learn Python
Enter Author of the Book: V.G.Ramakrishnan
Do you want continue....y
```


XII Std - CS EM Chapter-10.indd 178 24-01-2020 12:42:36

```
1. Add New Book
 2.Display Books
Enter your choice : 1
Enter Name of the Book: Advanced Python
Enter Author of the Book: Dr. Vidhya
Do you want continue....y
1. Add New Book
 2.Display Books
Enter your choice : 1
Enter Name of the Book: Working with OpenOffice
Enter Author of the Book: N.V.Gowrisankar
Do you want continue....y
1. Add New Book
 2.Display Books
Enter your choice : 1
Enter Name of the Book: Data Structure
Enter Author of the Book: K.Lenin
Do you want continue....y
1. Add New Book
 2.Display Books
Enter your choice : 1
Enter Name of the Book: An Introduction to Database System
Enter Author of the Book: R.Sreenivasan
Do you want continue....y
1. Add New Book
 2.Display Books
```


XII Std - CS EM Chapter-10.indd 179 24-01-2020 12:42:36

```
Enter your choice : 2
Name of the Book:  Programming in C++
Author of the Book:  K. Kannan

Name of the Book:  Learn Python
Author of the Book:  V.G.Ramakrishnan

Name of the Book:  Advanced Python
Author of the Book:  Dr. Vidhya

Name of the Book:  Working with OpenOffice
Author of the Book:  N.V.Gowrisankar

Name of the Book:  Data Structure
Author of the Book:  K.Lenin

Name of the Book:  An Introduction to Database System
Author of the Book:  R.Sreenivasan

Do you want continue....n
```

Program 3: Write a program to accept a string and print the number of uppercase, lowercase, vowels, consonants and spaces in the given string

```
class String:
        def __init__(self):
                self.uppercase=0
                self.lowercase=0
                self.vowels=0
                self.consonants=0
                self.spaces=0
                self.string=""
        def getstr(self):
                self.string=str(input("Enter a String: "))
```


XII Std - CS EM Chapter-10.indd 180 24-01-2020 12:42:36

```
def count_upper(self):
        for ch in self.string:
                if (ch.isupper()):
                        self.uppercase+=1

def count_lower(self):
        for ch in self.string:
                if (ch.islower()):
                        self.lowercase+=1

def count_vowels(self):
        for ch in self.string:
                if (ch in ('A', 'a', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U')):
                        self.vowels+=1

def count_consonants(self):
        for ch in self.string:
                if (ch not in ('A', 'a', 'e', 'E', 'i', 'I', 'o', 'O', 'u', 'U')):
                        self.consonants+=1

def count_space(self):
        for ch in self.string:
                if (ch==" "):
                        self.spaces+=1

def execute(self):
        self.count_upper()
        self.count_lower()
        self.count_vowels()
        self.count_consonants()
        self.count_space()

def display(self):
        print("The given string contains...")
        print("%d Uppercase letters"%self.uppercase)
        print("%d Lowercase letters"%self.lowercase)
        print("%d Vowels"%self.vowels)
        print("%d Consonants"%self.consonants)
        print("%d Spaces"%self.spaces)
```


XII Std - CS EM Chapter-10.indd 181 24-01-2020 12:42:36

```
S = String()
S.getstr()
S.execute()
S.display()
```

Output

```
Enter a String:Welcome To Learn Computer Science
The given string contains...
5 Uppercase letters
24 Lowercase letters
12 Vowels
17 Consonants
3 Spaces
```

Program 4: Write a program to store product and its cost price. Display all the available products and prompt to enter quantity of all the products. Finally generate a bill which displays the total amount to be paid

```
class MyStore:
    __prod_code=[]
    __prod_name=[]
    __cost_price=[]
    __prod_quant=[]

def getdata(self):
    self.p = int(input("Enter no. of products you need to store: "))
    for x in range(self.p):
        self.__prod_code.append(int(input("Enter Product Code: ")))
        self.__prod_name.append(str(input("Enter Product Name: ")))
        self.__cost_price.append(int(input("Enter Cost price: ")))
```


XII Std - CS EM Chapter-10.indd 182 24-01-2020 12:42:36

```
def display(self):
        print("Stock in Stores")
        print("-------------------------------------------------------")
        print("Product Code \t Product Name \t Cost Price")
        print("-------------------------------------------------------")
        for x in range(self.p):
                print(self.__prod_code[x], "\t\t", self.__prod_name[x], "\t\t", self.__cost_
                                                                                price[x])
        print("-------------------------------------------------------")

def print_bill(self):
        total_price = 0
        for x in range(self.p):
                q=int(input("Enter the quantify for the product code %d : "%self.__
                                                                        prod_code[x]))

self.__prod_quant.append(q)
total_price = total_price +self.__cost_price[x]*self.__prod_quant[x]

        print("          Invoice Receipt                 ")
        print("-------------------------------------------------------------------------")
        print("Product Code\t Product Name\t Cost Price\t Quantity \t Total Amount")
        print("-------------------------------------------------------------------------")
        for x in range(self.p):
                print(self.__prod_code[x], "\t\t", self.__prod_name[x], "\t\t",
                        self.__cost_price[x], "\t\t", self.__prod_quant[x], "\t\t",
                        self.__prod_quant[x]*self.__cost_price[x])
        print("-------------------------------------------------------------------------")
        print("                                        Total Amount = ", total_price)
```


XII Std - CS EM Chapter-10.indd 183 24-01-2020 12:42:36

```
S=MyStore()
S.getdata()
S.display()
S.print_bill()
```

Output:

```
Enter no. of products you need to store: 5
Enter Product Code: 101
Enter Product Name: Product-A
Enter Cost price: 25
Enter Product Code: 201
Enter Product Name: Product-B
Enter Cost price: 35
Enter Product Code: 301
Enter Product Name: Product-C
Enter Cost price: 35
Enter Product Code: 401
Enter Product Name: Product-D
Enter Cost price: 50
Enter Product Code: 501
Enter Product Name: Product-E
Enter Cost price: 120
```

Stock in Stores

Product Code	Product Name	Cost Price
101	Product-A	25
201	Product-B	35
301	Product-C	35
401	Product-D	50
501	Product-E	120

XII Std - CS EM Chapter-10.indd 184 24-01-2020 12:42:36

Enter the quantify for the product code 101 : 10
Enter the quantify for the product code 201 : 15
Enter the quantify for the product code 301 : 10
Enter the quantify for the product code 401 : 20
Enter the quantify for the product code 501 : 10

Invoice Receipt

Product Code	Product Name	Cost Price	Quantity	Total Amount
101	Product-A	25	10	250
201	Product-B	35	15	525
301	Product-C	35	10	350
401	Product-D	50	20	1000
501	Product-E	120	10	1200

Total Amount = 3325

Points to remember

- Python is an Object Oriented Programming language.
- Classes and Objects are the key features of Object Oriented Programming.
- In Python, a class is defined by using the keyword class.
- Variables defined inside a class is called as "Class Variable" and function are called as "Methods".
- The process of creating object is called as "Class Instantiation".
- Constructor is the special function that is automatically executed when an object of a class is created.
- In Python, there is a special function called "init" is used as Constructor.
- Destructor is also a special method gets execution automatically when an object exits from the scope.
- In Python, __del__( ) method is used as destructor.
- A variable prefixed with double underscore is becomes private in nature.

XII Std - CS EM Chapter-10.indd 185 24-01-2020 12:42:36

Hands on Experience

1. Write a program using class to store name and marks of students in list and print total marks.

2. Write a program using class to accept three sides of a triangle and print its area.

3. Write a menu driven program to read, display, add and subtract two distances.

IB13AK

Evaluation

Part - I

Choose the best answer **(1 Marks)**

1. Which of the following are the key features of an Object Oriented Programming language?
   (a) Constructor and Classes (b) Constructor and Object
   (c) Classes and Objects (d) Constructor and Destructor
2. Functions defined inside a class:
   (a) Functions (b) Module
   (c) Methods (d) section
3. Class members are accessed through which operator?
   (a) & (b) .
   (c) # (d) %
4. Which of the following method is automatically executed when an object is created?
   (a) __object__( ) (b) __del__( )
   (c) __func__( ) (d) __init__( )
5. A private class variable is prefixed with
   (a) __ (b) &&
   (c) ## (d) **
6. Which of the following method is used as destructor?
   (a) __init__( ) (b) __dest__( )
   (c) __rem__( ) (d) __del__( )

XII Std - CS EM Chapter-10.indd 186 24-01-2020 12:42:36

7. Which of the following class declaration is correct?

   (a) class class_name (b) class class_name<>

   (c) class class_name: (d) class class_name[ ]

8. Which of the following is the output of the following program?

```
class Student:
        def __init__(self, name):
                self.name=name
                print (self.name)
S=Student("Tamil")
```

   (a) Error (b) Tamil

   (c) name (d) self

9. Which of the following is the private class variable?

   (a) __num (b) ##num

   (c) $$num (d) &&num

10. The process of creating an object is called as:

   (a) Constructor (b) Destructor

   (c) Initialize (d) Instantiation

Part -II

Answer the following questions **(2 Marks)**

1. What is class?
2. What is instantiation?
3. What is the output of the following program?

```
class Sample:
        __num=10
        def disp(self):
                print(self.__num)
S=Sample()
S.disp()
print(S.__num)
```

4. How will you create constructor in Python?
5. What is the purpose of Destructor?

XII Std - CS EM Chapter-10.indd 187 24-01-2020 12:42:36

Part -III

Answer the following questions (3 Marks)

1. What are class members? How do you define it?
2. Write a class with two private class variables and print the sum using a method.
3. Find the error in the following program to get the given output?

```
class Fruits:
        def __init__(self, f1, f2):
                self.f1=f1
                self.f2=f2
        def display(self):
                print("Fruit 1 = %s, Fruit 2 = %s" %(self.f1, self.f2))
F = Fruits ('Apple', 'Mango')
del F.display
F.display()
```

Output

```
Fruit 1 = Apple,   Fruit 2 = Mango
```

4. What is the output of the following program?

```
class Greeting:
        def __init__(self, name):
                self.__name = name
        def display(self):
                print("Good Morning ", self.__name)
obj=Greeting('Bindu Madhavan')
obj.display()
```

5. How to define constructor and destructor in Python?

Part -IV

Answer the following questions (5 Marks)

1. Write a menu driven program to add or delete stationary items. You should use dictionary to store items and the brand.

References

1. *https://docs.python.org/3/tutorial/index.html*
2. *https://www.techbeamers.com/python-tutorial-step-by-step/#tutorial-list*
3. *Python programming using problem solving approach – Reema Thareja – Oxford University press.*
4. *Python Crash Course – Eric Matthes – No starch press, San Francisco.*

XII Std - CS EM Chapter-10.indd 188 24-01-2020 12:42:36

Unit IV

CHAPTER 11

DATABASE CONCEPTS

Learning Objectives

At the completion of this chapter, the student will be able to know

- the concept of a database and relational database.
- different components of the database.
- types of database models.
- types of relationship.
- the concepts of relational algebra.

Introduction

A database is an organized collection of data, generally stored and accessed electronically from a computer system. The term "database" is also used to refer to any of the DBMS, the database system or an application associated with the database. Because of the close relationship between them, the term "database" is often used casually to refer to both a database and the DBMS used to manipulate it. A school class register is a database where names are arranged alphabetically. Databases have been around since people started recording things. Here we tend to focus on electronic ones.

11.1 Data

Data are raw facts stored in a computer. A data may contain any character, text, word or a number.

Example : 600006, DPI Campus, SCERT, Chennai, College Road

11.2 Information

Information is formatted data, which allows to be utilized in a significant way.

Example

SCERT
College Road
DPI Campus
Chennai 600006

J5SWBK

As you can see from the example above, data appears as a set of words and numbers. However, when the data is processed, organized and formatted, it gives a meaningful information about the SCERT institution contact address.

11.3 Database

Database is a repository collection of related data organized in a way that data can be easily accessed, managed and updated. Database can be a software or hardware based, with one sole purpose of storing data.

11.4 DataBase Management System (DBMS)

A DBMS is a software that allows us to create, define and manipulate database, allowing users to store, process and analyze data easily. DBMS provides us with an interface or a tool, to perform various operations to create a database, storing of data and for updating data, etc. DBMS also provides protection and security to the

XII Std - CS EM Chapter-11.indd 189 1/24/2020 10:09:34 AM

databases. It also maintains data consistency in case of multiple users.

Examples of DBMS softwares are : Foxpro, dbase.

11.4.1 Characteristics of Database Management System

1. Data stored into Tables	Data is never directly stored into the database. Data is stored into tables, created inside the database. DBMS also allows to have relationship between tables which makes the data more meaningful and connected.
2. Reduced Redundancy	In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of data in database was a big problem But DBMS follows Normalisation which divides the data in such a way that repetition is minimum.
3.Data Consistency	On live data, it is being continuously updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it by itself.
4. Support Multiple user and Concurrent Access	DBMS allows multiple users to work on it(update, insert, delete data) at the same time and still manages to maintain the data consistency.
5.Query Language	DBMS provides users with a simple query language, using which data can be easily fetched, inserted, deleted and updated in a database.
6. Security	The DBMS also takes care of the security of data, protecting the data from unauthorized access. In a typical DBMS, we can create user accounts with different access permissions, using which we can easily secure our data by restricting user access.
7. DBMS Supports Transactions	It allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

11.4.2 Advantages of DBMS

- Segregation of application program
- Minimal data duplication or Data Redundancy
- Easy retrieval of data using the Query Language
- Reduced development time and maintenance

XII Std - CS EM Chapter-11.indd 190 1/24/2020 10:09:34 AM

11.4.3 Components of DBMS

The Database Management System can be divided into five major components as follows:

1.Hardware 2.Software

3.Data 4.Procedures/Methods

5.Database Access Languages

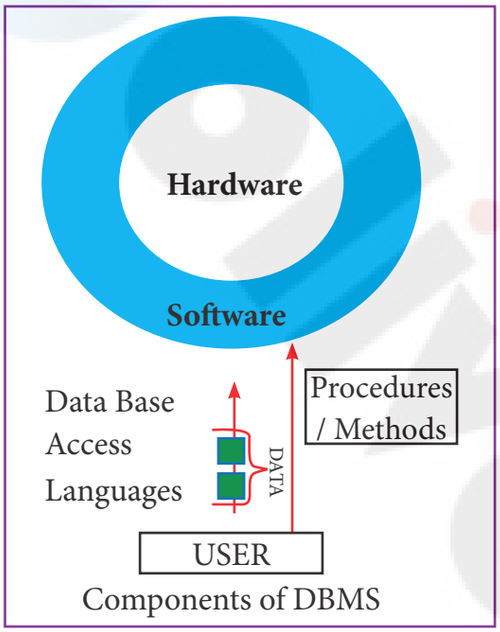


Figure 11.1

1. Hardware: The computer, hard disk, I/O channels for data, and any other physical component involved in storage of data

2. Software: This main component is a program that controls everything. The DBMS software is capable of understanding the Database Access Languages and interprets into database commands for execution.

3. Data: It is the resource for which DBMS is designed. DBMS creation is to store and utilize data.

4. Procedures/Methods: They are general instructions to use a database management system such as installation of DBMS, manage databases to take backups, report generation, etc.

5. DataBase Access Languages: They are the languages used to write commands to access, insert, update and delete data stored in any database.

Examples of popular DBMS: Dbase, FoxPro

11.5 Database Structure

Table is the entire collection of related data in one table, referred to as a File or Table where the data is organized as row and column.

Each row in a table represents a record, which is a set of data for each database entry.

Each table column represents a Field, which groups each piece or item of data among the records into specific categories or types of data. Eg. StuNo., StuName, StuAge, StuClass, StuSec.

A Table is known as a RELATION

A Row is known as a TUPLE

A column is known as an ATTRIBUTE

XII Std - CS EM Chapter-11.indd 191 1/24/2020 10:09:34 AM

Attribute / Column

Tuple / Row

StuNo	StuName	StuAge	StuClass	StuSec
1101	Kannan	14	9	A
1102	Ramakrishnan	14	9	A
1103	Vidhya	14	10	B
1104	Britto	15	9	A
1105	Padmaja	14	10	B

Table / Relation

DataBase Structure

Fig 11.2

11.6 Data Model

- A data model describes how the data can be represented and accessed from a software after complete implementation
- It is a simple abstraction of complex real world data gathering environment.
- The main purpose of data model is to give an idea as how the final system or software will look like after development is completed.

11.6.1 Types of Data Model

Following are the different types of a Data Model

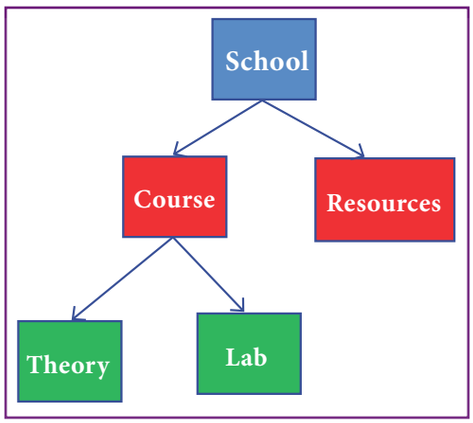
- Hierarchical Model
- Relational Model
- Network Database Model
- Entity Relationship Model
- Object Model

1. Hierarchical Model

Hierarchical model was developed by IBM as Information Management System.

In Hierarchical model, data is represented as a simple tree like structure form. This model represents a one-to-many relationship i.e parent-child relationship. One child can have only one parent but one parent can have many children. This model is mainly used in IBM Main Frame computers.

XII Std - CS EM Chapter-11.indd 192 1/24/2020 10:09:35 AM



Hierarchical Model Fig. 11.3

2. Relational Model

The Relational Database model was first proposed by E.F. Codd in 1970 . Nowadays, it is the most widespread data model used for database applications around the world.

The basic structure of data in relational model is tables (relations). All the information's related to a particular type is stored in rows of that table. Hence tables are also known as relations in a relational model. A relation key is an attribute which uniquely identifies a particular tuple (row in a relation (table)).

Stu_id	Name	Age
1	Malar	17
2	Suncar	16
3	Velu	16

Subj_id	Name	Teacher
1	C++	Kannan
2	Php	Ramakrishnan
3	Python	Vidhya

Stu_id	Subj_id	Marks
1	1	92
1	2	89
3	2	96

Relational Model
Fig. 11.4

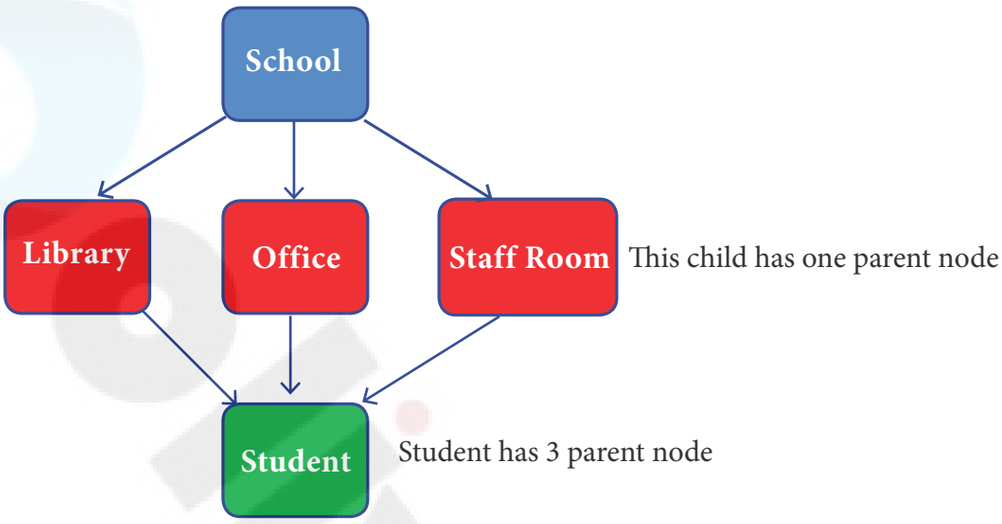
3. Network Model

Network database model is an extended form of hierarchical data model. The difference between hierarchical and Network data model is :

- In hierarchical model, a child record has only one parent node,

XII Std - CS EM Chapter-11.indd 193 1/24/2020 10:09:35 AM

- In a Network model, a child may have many parent nodes. It represents the data in many-to-many relationships.
- This model is easier and faster to access the data.



Network Model
Fig. 11.5

School represents the parent node

Library, Office and Staff room is a child to school (parent node)

Student is a child to library, office and staff room (one to many relationship)

4. Entity Relationship Model. (ER model)

In this database model, relationship are created by dividing the object into entity and its characteristics into attributes.

It was developed by Chen in 1976. This model is useful in developing a conceptual design for the database. It is very simple and easy to design logical view of data. The developer can easily understand the system by looking at ER model constructed.

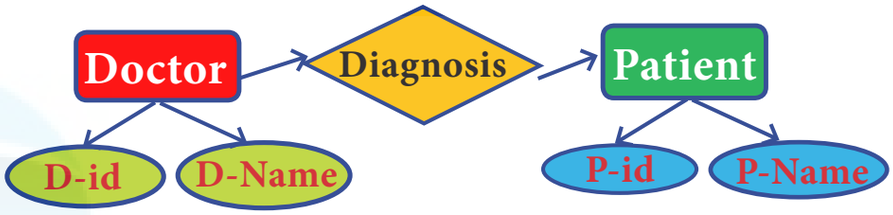
Rectangle represents the entities. E.g. Doctor and Patient

Ellipse represents the attributes E.g. D-id, D-name, P-id, P-name. Attributes describes the characteristics and each entity becomes a major part of the data stored in the database.

Diamond represents the relationship in ER diagrams

E.g. Doctor diagnosis the Patient

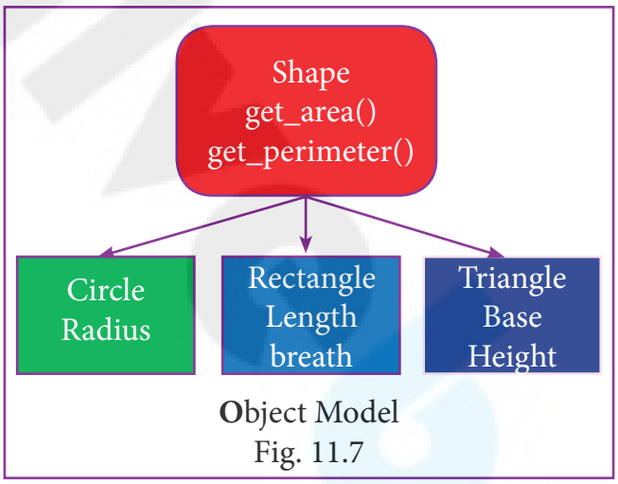
XII Std - CS EM Chapter-11.indd 194 1/24/2020 10:09:35 AM



ER model
Fig. 11.6

5. Object Model

Object model stores the data in the form of objects, attributes and methods, classes and Inheritance. This model handles more complex applications, such as Geographic information System (GIS), scientific experiments, engineering design and manufacturing. It is used in file Management System. It represents real world objects, attributes and behaviors. It provides a clear modular structure. It is easy to maintain and modify the existing code.



Object Model
Fig. 11.7

An example of the Object model is **Shape, Circle, Rectangle** and **Triangle** are all objects in this model.

- **Circle** has the attribute **radius.**
- **Rectangle** has the attributes **length and breadth.**
- **Triangle** has the attributes **base and height** .
- The objects Circle, Rectangle and Triangle **inherit** from the object Shape.

11.6.2 Types of DBMS Users

Database Administrators

Database Administrator or DBA is the one who manages the complete database management system. DBA takes care of the security of the DBMS, managing the license keys, managing user accounts and access etc.

XII Std - CS EM Chapter-11.indd 195 1/24/2020 10:09:35 AM

Application Programmers or Software Developers

This user group is involved in developing and designing the parts of DBMS.

End User

All modern applications, web or mobile, store user data. Applications are programmed in such a way that they collect user data and store the data on DBMS systems running on their server. End users are the one who store, retrieve, update and delete data.

Database designers: are responsible for identifying the data to be stored in the database for choosing appropriate structures to represent and store the data.

11.7 Difference between DBMS and RDBMS

Basis of Comparison	DBMS	RDBMS
Expansion	Database Management System	Relational DataBase Management System
Data storage	Navigational model ie data by linked records	Relational model (in tables). ie data in tables as row and column
Data redundancy	Exhibit	Not Present
Normalization	Not performed	RDBMS uses normalization to reduce redundancy
Data access	Consumes more time	Faster, compared to DBMS.
Keys and indexes	Does not use.	used to establish relationship. Keys are used in RDBMS.
Transaction management	Inefficient, Error prone and insecure.	Efficient and secure.
Distributed Databases	Not supported	Supported by RDBMS.
Example	Dbase, FoxPro.	SQL server, Oracle, mysql, MariaDB, SQLite.

Database normalization was first proposed by ***Dr. Edgar F Codd*** as an integral part of RDBMS in order to reduce data redundancy and improve data integrity. These rules are known as E F Codd Rules.

XII Std - CS EM Chapter-11.indd 196 1/24/2020 10:09:35 AM

11.8 Types of Relationships

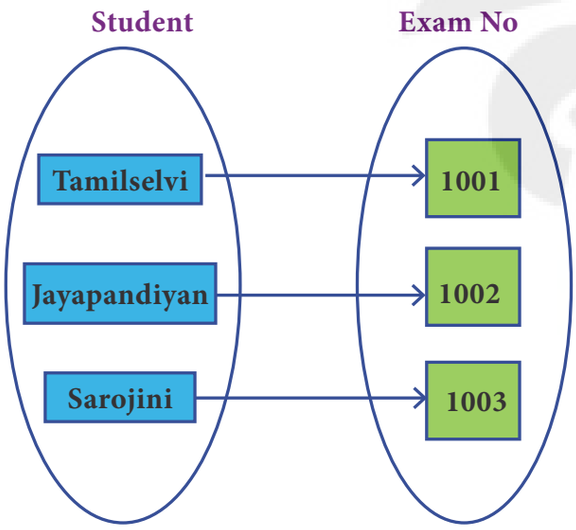
Following are the types of relationships used in a database.

1. One-to-One Relationship
2. One-to-Many Relationship
3. Many-to-One Relationship
4. Many-to-Many Relationship

1. One-to-One Relationship

In One-to-One Relationship, one entity is related with only one other entity. One row in a table is linked with only one row in another table and vice versa.

For example: A student can have only one exam number

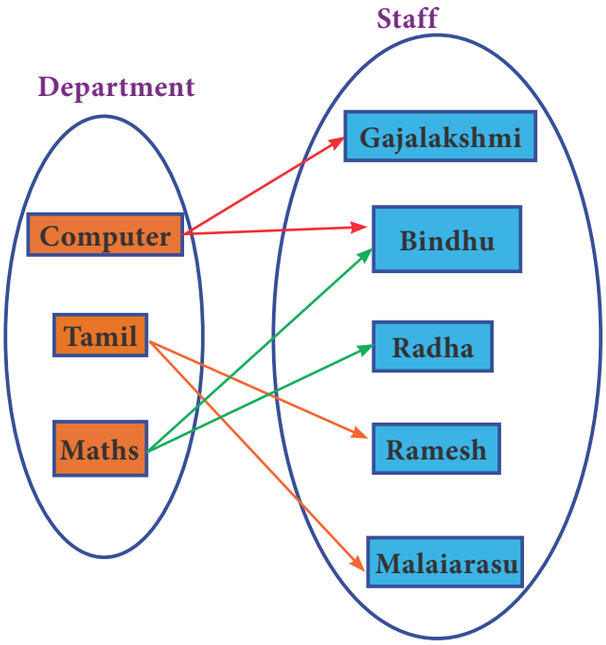


One to one Relationships
Fig 11.8

2. One-to-Many Relationship

In One-to-Many relationship, one entity is related to many other entities.

One row in a table A is linked to many rows in a table B, but one row in a table B is linked to only one row in table A. For example: One Department has many staff members.



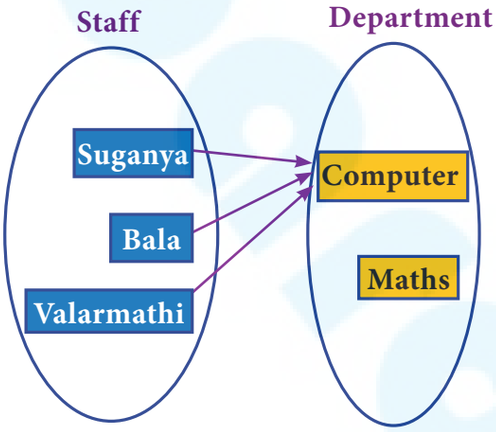
One to Many Mapping
Fig 11.9

3. Many-to-One Relationship

In Many-to-One Relationship, many entities can be related with only one in the other entity.

For example: A number of staff members working in one Department.

Multiple rows in staff members table is related with only one row in Department table.



Many to one Relationship
Fig 11.10

XII Std - CS EM Chapter-11.indd 197
1/24/2020 10:09:35 AM

4. Many-to-Many Relationship

A many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table.

Example 1: Customers and Product

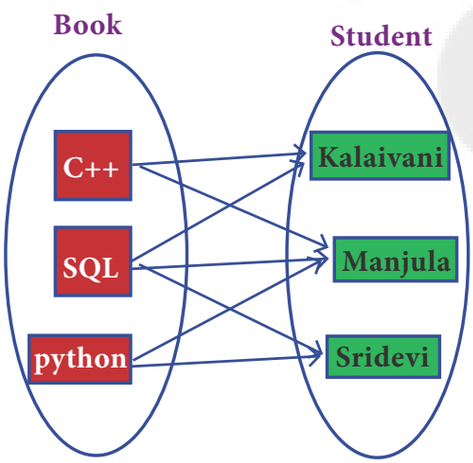
Customers can purchase various products and Products can be purchased by many customers

Example 2: Students and Courses

A student can register for many Courses and a Course may include many students

Example 3: Books and Student.

Many Books in a Library are issued to many students.



Many to Many Relationship
Fig 11.11

DO YOU KNOW?

The relational model was invented by Edgar Frank Codd (Father of Relational DataBase) as a general model of data, and subsequently promoted by Chris Date and Hugh Darwen among others.

11.9 Relational Algebra in DBMS

What is Relational Algebra?

Relational Algebra, was first created by ***Edgar F Codd*** while at IBM. It was used for modeling the data stored in relational databases and defining queries on it.

Relational Algebra is a procedural query language used to query the database tables using SQL.

Relational algebra operations are performed recursively on a relation (table) to yield an output. The output of these operations is a new relation, which might be formed by one or more input relations.

Relational Algebra is divided into various groups

Unary Relational Operations

SELECT (symbol : σ)

PROJECT (symbol : Π)

Relational Algebra Operations from Set Theory

- UNION ($\cup$)
- INTERSECTION ($\cap$)
- DIFFERENCE ($-$)
- CARTESIAN PRODUCT (X)

SELECT (symbol : σ)

General form σ_c (R) with a relation R and a condition C on the attributes of R.

The SELECT operation is used for selecting a subset with tuples according to a given condition.

Select filters out all tuples that do not satisfy C.

XII Std - CS EM Chapter-11.indd 198 1/24/2020 10:09:35 AM

STUDENT

Studno	Name	Course	Year
cs1	Kannan	Big Data	II
cs2	Gowri Shankar	R language	I
cs3	Lenin	Big Data	I
cs4	Padmaja	Python Programming	I

Table 11.1

σ_{course} = "Big Data" **(STUDENT)**

Studno	Name	Course	Year
cs1	Kannan	Big Data	II
cs3	Lenin	Big Data	I

PROJECT (symbol : Π)

The projection eliminates all attributes of the input relation but those mentioned in the projection list. The projection method defines a relation that contains a vertical subset of Relation.

Example 1 using Table 11.1

Π_{course} (STUDENT)

Result

Course
Big Data
R language
Python Programming

Note

duplicate row is removed in the result

XII Std - CS EM Chapter-11.indd 199 1/24/2020 10:09:35 AM

Example 2 (using Table 11.1)

$\Pi_{\text{studno, course}}$ (STUDENT)

Result

Studno	Course
cs1	Big Data
cs2	R language
cs3	Big Data
cs4	Python Programming

UNION (Symbol :∪)

It includes all tuples that are in tables A or in B. It also eliminates duplicates. Set A Union Set B would be expressed as A ∪ B

Example 3

Consider the following tables

Table A	
Studno	**Name**
cs1	Kannan
cs3	Lenin
cs4	Padmaja

Table B	
Studno	**Name**
cs1	Kannan
cs2	GowriShankarn
cs3	Lenin

Table 11.2

Result

Table A ∪ B	
Studno	**Name**
cs1	Kannan
cs2	GowriShankar
cs3	Lenin
cs4	Padmaja

SET DIFFERENCE (Symbol : -)

The result of A – B, is a relation which includes all tuples that are in A but not in B.

The attribute name of A has to match with the attribute name in B.

Example 4 (using Table 11.2)

XII Std - CS EM Chapter-11.indd 200 1/24/2020 10:09:35 AM

Result

Table A – B	
cs4	Padmaja

INTERSECTION (symbol : ∩) A ∩ B

Defines a relation consisting of a set of all tuple that are in both in A and B. However, A and B must be union-compatible.

Example 5 (using Table 11.2)

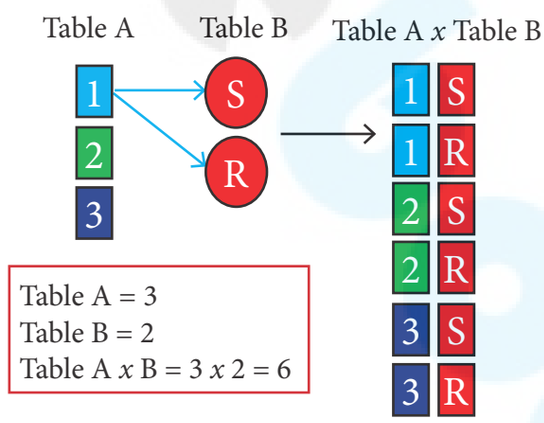
A ∩ B	
cs1	Kannan
cs3	Lenin

PRODUCT OR CARTESIAN PRODUCT (Symbol : X)

Cross product is a way of combining two relations. The resulting relation contains, both relations being combined.

A x B means A times B, where the relation A and B have different attributes.

This type of operation is helpful to merge columns from two relations.



Cartesian Product
Fig. 11.12

Table A		Table B	
studno	**name**	**studno**	**subject**
cs1	Kannan	cs28	Big Data
cs2	Gowri Shankar	cs62	R language
cs4	Padmaja	cs25	python programming

Table 11.3

XII Std - CS EM Chapter-11.indd 201 1/24/2020 10:09:35 AM

Cartesian product : Table A x Table B

studno	name	course	subject
cs1	Kannan	cs28	Big Data
cs1	Kannan	cs62	R language
cs1	Kannan	cs25	python rogramming
cs2	Gowri Shankar	cs28	Big Data
cs2	Gowri Shankar	cs62	R language
cs2	Gowri Shankar	cs25	python programming
cs4	Padmaja	cs28	Big Data
cs4	Padmaja	cs62	R language
cs4	Padmaja	cs25	python programming

Points to remember:

- DBMS is a computer based record keeping system
- Data is unprocessed data which contains any character, text, word or number has no meaning
- Information is processed data, organized and formatted.
- Examples of RDBMS are mysql, oracle, sql server, ibm db2
- Redundancy means duplication of data in a database.
- Data Consistency means that data values are the same at all instances of a database
- Data Integrity is security from unauthorized users
- Table is known a relation
- A row is called a tuple
- A column is known as an attribute
- Types of data model are Hierarchical, Relational, Network, ER and Object model.
- Hierarchical model is a simple tree like structure form with one-to-one relationship called parent-child relationship
- Relational Model represents data as relations or tables
- Network model is similar to Hierarchical model but it allows a record to have more than one parent
- ER model consists of entities, attributes and relationships

XII Std - CS EM Chapter-11.indd 202 1/24/2020 10:09:35 AM

Points to remember:

- Object model stores data as objects, attributes, methods, classes and inheritance
- Normalization reduces data redundancy and improves data integrity
- Different types of Relationship are one-to-one, one-to-many, many-to-one and many-to-many relationships
- Database Normalization was proposed by Dr.Edgar F Codd
- Relational Algebra is used for modeling data stored in relational databases and for defining queries on it.

Evaluation

JKBRCW

Part - A

Choose the best answer **(1 Marks)**

1. What is the acronym of DBMS?

   a) DataBase Management Symbol b) Database Managing System

   c) DataBase Management System d) DataBasic Management System

2 A table is known as

   a) tuple b) attribute

   c) relation d)entity

3 Which database model represents parent-child relationship?

   a) Relational b) Network

   c) Hierarchical d) Object

4 Relational database model was first proposed by

   a) E F Codd b) E E Codd

   c) E F Cadd d) E F Codder

5 What type of relationship does hierarchical model represents?

   a) one-to-one b) one-to-many

   c) many-to-one d) many-to-many

6. Who is called Father of Relational Database from the following?

   a) Chris Date b)Hugh Darween

   c) Edgar Frank Codd d) Edgar Frank Cadd

XII Std - CS EM Chapter-11.indd 203 1/24/2020 10:09:35 AM

7. Which of the following is an RDBMS?

a) Dbase b) Foxpro

c) Microsoft Access d) SQLite

8 What symbol is used for SELECT statement?

a) σ b) Π c) X d) Ω

9 A tuple is also known as

a) table b) row

c) attribute d) field

10. Who developed ER model?

a) Chen b) EF Codd

c) Chend d) Chand

Part - B

Answer the following questions (2 Marks)

1. Mention few examples of a database.
2. List some examples of RDBMS.
3. What is data consistency?
4. What is the difference between Hierarchical and Network data model?
5. What is normalization?

Part -C

Answer the following questions (3 Marks)

1. What is the difference between Select and Project command?
2. What is the role of DBA?
3. Explain Cartesian Product with a suitable example.
4. Explain Object Model with example.
5. Write a note on different types of DBMS users.

Part -D

Answer the following questions (5 Marks)

1. Explain the different types of data model.
2. Explain the different types of relationship mapping.
3. Differentiate DBMS and RDBMS.
4. Explain the different operators in Relational algebra with suitable examples.
5. Explain the characteristics of DBMS.

XII Std - CS EM Chapter-11.indd 204 1/24/2020 10:09:35 AM

Unit IV

CHAPTER 12

STRUCTURED QUERY LANGUAGE (SQL)

KF4KDW

Learning Objectives

After studying this lesson, students will be able to:

- The processing skills of SQL.
- The components of SQL.
- To create a table by specifying the fields and records.
- To apply various manipulations like inserting, updating and deleting records in a table.
- To learn about various constraints and to apply it on tables.
- To generate queries in the table by applying various clauses.
- To modify the structure of an existing table.
- The commands to delete records, table and revoke the commands.

12.1 Introduction to SQL

The Structured Query Language (SQL) is a standard programming language to access and manipulate databases. SQL allows the user to create, retrieve, alter, and transfer information among databases. It is a language designed for managing and accessing data in a Relational Data Base Management System (RDBMS).

There are many versions of SQL. The original version was developed at IBM's Research centre and originally called as Sequel in early 1970's. Later the language was changed to SQL. In 1986, ANSI (American National Standard Institute) published an SQL standard that was updated again in 1992, the latest SQL was released in 2008 and named as SQL 2008.

Note

Latest SQL standard as of now is SQL 2008, released in 2008.

12.2 Role of SQL in RDBMS

RDBMS stands for Relational DataBase Management System. Oracle, MySQL, MS SQL Server, IBM DB2 and Microsoft Access are RDBMS packages. SQL is a language used to access data in such databases.

XII Std - CS EM Chapter-12.indd 205 24-01-2020 11:20:32

In general, Database is a collection of tables that store sets of data that can be queried for use in other applications. A database management system supports the development, administration and use of database platforms. RDBMS is a type of DBMS with a row-based table structure that connects related data elements and includes functions related to Create, Read, Update and Delete operations, collectively known as CRUD.

The data in RDBMS, is stored in database objects, called Tables. A table is a collection of related data entries and it consist of rows and columns.

A field is a column in a table that is designed to maintain specific related information about every record in the table. It is a vertical entity that contains all information associated with a specific field in a table. The fields in a student table may be of the type AdmnNo, StudName, StudAge, StudClass, Place etc.

A Record is a row, which is a collection of related fields or columns that exist in a table. A record is a horizontal entity in a table which represents the details of a particular student in a student table.

12.3 Processing Skills of SQL

The various processing skills of SQL are :

1. **Data Definition Language (DDL) :** The SQL DDL provides commands for defining relation schemas (structure), deleting relations, creating indexes and modifying relation schemas.
2. **Data Manipulation Language (DML)** : The SQL DML includes commands to insert, delete, and modify tuples in the database.
3. **Embedded Data Manipulation Language :** The embedded form of SQL is used in high level programming languages.
4. **View Definition :** The SQL also includes commands for defining views of tables.
5. **Authorization :** The SQL includes commands for access rights to relations and views of tables.
6. **Integrity :** The SQL provides forms for integrity checking using condition.
7. **Transaction control :** The SQL includes commands for file transactions and control over transaction processing.

DO YOU KNOW?

SQL-Structured Query Language is a language used for accessing databases while **MySQL** is a database management system, like SQL Server, Oracle, Informix, Postgres, etc. **MySQL** is a **RDBMS**.

Refer installing MYSQL in Annexure -1

XII Std - CS EM Chapter-12.indd 206 24-01-2020 11:20:32

12.4 Creating Database

1. To create a database, type the following command in the prompt:

   CREATE DATABASE database_name;

   For example to create a database to store the tables:

   CREATE DATABASE stud;

2. To work with the database, type the following command.

   USE DATABASE;

   For example to use the stud database created, give the command

   USE stud;

KUMFF8

DO YOU KNOW

WAMP stands for **"Windows, Apache, MySQL** and **PHP". WAMP** is a variation of LAMP for windows systems and is installed as a software bundle (Apache, MySQL and PHP). It is often used for web development and internal testing, but may also be used to serve live websites.

12.5 Components of SQL

SQL commands are divided into five categories:

- DML - Data Manipulation Language
- DDL - Data Definition Language
- DCL - Data Control Language
- TCL - Transaction Control Language
- DQL - Data Query Language

12.5.1 DATA DEFINITION LANGUAGE

The Data Definition Language (DDL) consist of SQL statements used to define the database structure or schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in databases.

The DDL provides a set of definitions to specify the storage structure and access methods used by the database system.

A DDL performs the following functions :

1. It should identify the type of data division such as data item, segment, record and database file.
2. It gives a unique name to each data item type, record type, file type and data base.
3. It should specify the proper data type.
4. It should define the size of the data item.
5. It may define the range of values that a data item may use.
6. It may specify privacy locks for preventing unauthorized data entry.

SQL commands which comes under Data Definition Language are:

Create	To create tables in the database.
Alter	Alters the structure of the database.
Drop	Delete tables from database.
Truncate	Remove all records from a table, also release the space occupied by those records.

12.5.2 DATA MANIPULATION LANGUAGE

A Data Manipulation Language (DML) is a computer programming language used for adding (inserting), removing (deleting), and modifying (updating) data in a database. In SQL, the data manipulation language comprises the SQL-data change statements, which modify stored data but not the schema of the database table.

After the database schema has been specified and the database has been created, the data can be manipulated using a set of procedures which are expressed by DML.

By Data Manipulation we mean,

- Insertion of new information into the database
- Retrieval of information stored in a database.
- Deletion of information from the database.
- Modification of data stored in the database.

The DML is basically of two types:

Procedural DML – Requires a user to specify what data is needed and how to get it.

Non-Procedural DML - Requires a user to specify what data is needed without specifying how to get it.

XII Std - CS EM Chapter-12.indd 208 24-01-2020 11:20:32

SQL commands which comes under Data Manipulation Language are :

Insert	Inserts data into a table
Update	Updates the existing data within a table.
Delete	Deletes all records from a table, but not the space occupied by them.

12.5.3 DATA CONTROL LANGUAGE

A **Data Control Language (DCL)** is a programming language used to control the access of data stored in a database. It is used for controlling privileges in the database (Authorization). The privileges are required for performing all the database operations such as creating sequences, views of tables etc.

SQL commands which come under Data Control Language are:

Grant	Grants permission to one or more users to perform specific tasks.
Revoke	Withdraws the access permission given by the GRANT statement.

12.5.4 TRANSACTIONAL CONTROL LANGUAGE

Transactional control language (TCL) commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table by DML statements.

SQL command which come under Transfer Control Language are:

Commit	Saves any transaction into the database permanently.
Roll back	Restores the database to last commit state.
Save point	Temporarily save a transaction so that you can rollback.

12.5.5 DATA QUERY LANGUAGE

The Data Query Language consist of commands used to query or retrieve data from a database. One such SQL command in Data Query Language is

Select	It displays the records from the table.

12.6 Data Types

The data in a database is stored based on the kind of value stored in it. This is identified as the data type of the data or by assigning each field a data type. All the values in a given field must be of same type.

XII Std - CS EM Chapter-12.indd 209 24-01-2020 11:20:32

The ANSI SQL standard recognizes only Text and Number data type, while some commercial programs use other datatypes like Date and Time etc. The ANSI data types are listed below in Table 12.1

Data Type	Description
char (Character)	Fixed width string value. Values of this type is enclosed in single quotes. For ex. Anu's will be written as 'Anu' 's'.
varchar	Variable width character string. This is similar to char except the size of the data entry vary considerably.
dec (Decimal)	It represents a fractional number such as 15.12, 0.123 etc. Here the size argument consist of two parts : precision and scale. The precision indicates how many digits the number may have and the scale indicates the maximum number of digits to the right of the decimal point. The size (5, 2) indicates precision as 5 and scale as 2. The scale cannot exceed the precision.
numeric	It is same as decimal except that the maximum number of digits may not exceed the precision argument.
int (Integer)	It represents a number without a decimal point. Here the size argument is not used.
smallint	It is same as integer but the default size may be smaller than Integer.
float	It represents a floating point number in base 10 exponential notation and may define a precision up to a maximum of 64.
real	It is same as float, except the size argument is not used and may define a precision up to a maximum of 64.
double	Same as real except the precision may exceed 64.

Table 12.1

12.7 SQL Commands and their Functions

Tables are the only way to store data, therefore all the information has to be arranged in the form of tables. The SQL provides a predetermined set of commands to work on databases.

Keywords	They have a special meaning in SQL. They are understood as instructions.
Commands	They are instructions given by the user to the database also known as statements.
Clauses	They begin with a keyword and consist of keyword and argument.

XII Std - CS EM Chapter-12.indd 210 24-01-2020 11:20:32

Arguments	They are the values given to make the clause complete.

12.7.1 DDL Commands

CREATE TABLE Command

You can create a table by using the **CREATE TABLE** command. When a table is created, its columns are named, data types and sizes are to be specified. Each table must have at least one column. The syntax of **CREATE TABLE** command is :

```
CREATE TABLE <table-name>
(<column name><data type>[<size>]
(<column name><data type>[<size>]……
);
```

Now let us use the above syntax to store some information about the students of a class in a database, for this you first need to create table. To create a student table, let us take some information related to students like admission number which we can use it in short form as (admno), name of student (name), gender, age etc. of the student. Let us create a table having the field names Admno, Name, Gender, Age and Place.

The SQL command will be as follows:

```
CREATE TABLE Student
    (Admno   integer,
     Name  char(20),
     Gender  char(1),
     Age     integer,
     Place char(10),
     );
```

The above one is a simple table structure without any restrictions. You can also set constraints to limit the type of data that can go into the fields of the table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints could be either on a column level or a table level.

Note

Constraint is a condition applicable on a field or set of fields.

Column constraint:	Column constraint apply only to individual column.
Table constraint :	Table constraint apply to a group of one or more columns.

XII Std - CS EM Chapter-12.indd 211 24-01-2020 11:20:32

The syntax for a table created with constraint is given as below:

```
CREATE TABLE <table-name>
(<column name><data type>[<size>]<column constraint>,
(<column name><data type>[<size>]<column constraint>……
<table constraint>(<column name>,[<column name>….])…..
);
```

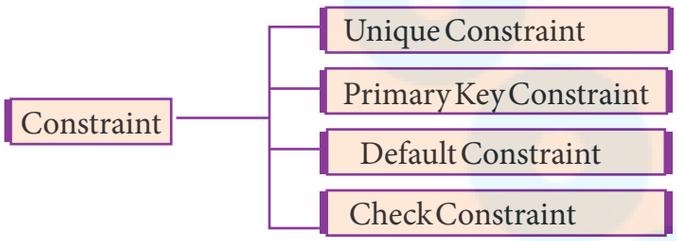
Following is an example for student table with **"NOT NULL"** column constraint. This constraint enforces a field to always contain a value.

```
CREATE TABLE Student
(
Admno integer    NOT NULL PRIMARY KEY, → Primary Key constraint
Name char(20)NOT NULL,
Gender char(1),
Age integer,
Place char(10),
);
```

The above command creates a table "student" in which the field Admno of integer type is defined NOT NULL, Name of char type is defined as NOT NULL which means these two fields must have values. The fields Gender, Age and Place do not have any constraints.

12.7.2 Type of Constraints

Constraints ensure database integrity, therefore known as database integrity constraints. The different types of constraints are :



12.7.2.1 Unique Constraint

This constraint ensures that no two rows have the same value in the specified columns. For example **UNIQUE** constraint applied on Admno of student table ensures that no two students have the same admission number and the constraint can be used as:

XII Std - CS EM Chapter-12.indd 212 24-01-2020 11:20:32

```
CREATE TABLE Student
(
Admno integer NOT NULL UNIQUE, → Unique constraint
Name char (20) NOT NULL,
Gender char (1),
Age integer,
Place char (10),
);
```

The **UNIQUE** constraint can be applied only to fields that have also been declared as **NOT NULL**.

When two constraints are applied on a single field, it is known as multiple constraints. In the above Multiple constraints **NOT NULL** and **UNIQUE** are applied on a single field Admno, the constraints are separated by a space and at the end of the field definition a comma(,) is added. By adding these two constraints the field Admno must take some value ie. will not be **NULL** and should not be duplicated.

12.7.2.2 Primary Key Constraint

This constraint declares a field as a Primary key which helps to uniquely identify a record. It is similar to unique constraint except that only one field of a table can be set as primary key. The primary key does not allow **NULL** values and therefore a field declared as primary key must have the **NOT NULL** constraint.

Example showing Primary Key Constraint in the student table:

```
CREATE TABLE Student
(
Admno integer    NOT NULL PRIMARY KEY, → Primary Key constraint
Name char(20)NOT NULL,
Gender char(1),
Age integer,
Place char(10),
);
```

In the above example the Admno field has been set as primary key and therefore will help us to uniquely identify a record, it is also set **NOT NULL**, therefore this field value cannot be empty.

12.7.2.3 DEFAULT Constraint

The **DEFAULT** constraint is used to assign a default value for the field. When no value is given for the specified field having **DEFAULT** constraint, automatically the default value will be assigned to the field.

XII Std - CS EM Chapter-12.indd 213 24-01-2020 11:20:32

Example showing **DEFAULT** Constraint in the student table:

```
CREATE TABLE Student
(
Admno integer    NOT NULL  PRIMARY KEY,
Name char(20)NOT NULL,
Gender char(1),
Age integer  DEFAULT = "17", → Default Constraint
Place char(10),
);
```

In the above example the "Age" field is assigned a default value of 17, therefore when no value is entered in age by the user, it automatically assigns 17 to Age.

12.7.2.4 Check Constraint

This constraint helps to set a limit value placed for a field. When we define a check constraint on a single column, it allows only the restricted values on that field. Example showing check constraint in the student table:

```
CREATE TABLE Student
(
Admno integer    NOT NULL PRIMARY KEY
Name char(20)NOT NULL,
Gender char(1),
Age integer CHECK (Age <=19), → Check Constraint
Place char(10),
);
```

In the above example the check constraint is set to Age field where the value of Age must be less than or equal to 19.

Note

The check constraint may use relational and logical operators for condition.

12.7.2.5 TABLE CONSTRAINT

When the constraint is applied to a group of fields of the table, it is known as Table constraint. The table constraint is normally given at the end of the table definition. Let us take a new table namely Student1 with the following fields Admno, Firstname, Lastname, Gender, Age, Place:

XII Std - CS EM Chapter-12.indd 214 24-01-2020 11:20:32

```
CREATE TABLE Student 1
(
Admno integer        NOT NULL,
Firstname char(20),
Lastname char(20),
Gender char(1),
Age integer,
Place char(10),
PRIMARY KEY (Firstname, Lastname) → Table constraint
);
```

In the above example, the two fields, Firstname and Lastname are defined as Primary key which is a Table constraint.

12.7.3 DML COMMANDS

Once the schema or structure of the table is created, values can be added to the table. The DML commands consist of inserting, deleting and updating rows into the table.

12.7.3.1 INSERT command

The **INSERT** command helps to add new data to the database or add new records to the table. The command is used as follows:

INSERT INTO <table-name> [column-list] VALUES (values);

INSERT INTO Student (Admno, Name, Gender, Age, Place)

VALUES (100,' Ashish'," M', 17,' Chennai');

INSERT INTO Student (Admno, Name, Gender, Age, Place)

VALUES (101, 'Adarsh', 'M', 18, 'Delhi');

Two new records are added to the table as shown below:

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi

The order of values must match the order of columns in the **CREATE TABLE** command. Specifying the column names is optional if data is to be added for all columns. The command to add values into the student table can also be used in the following way:

INSERT INTO Student VALUES (102, 'Akshith', 'M', '17,' 'Bangalore');

102	Akshith	M	17	Bangalore

The above command inserts the record into the student table.

To add data to only some columns in a record by specifying the column name and their data, it can be done by:

XII Std - CS EM Chapter-12.indd 215 24-01-2020 11:20:32

INSERT INTO Student(Admno, Name, Place) VALUES (103, 'Ayush', 'Delhi');

103	Ayush	M	18	Delhi

The above command adds the following record with default values of 'M' for Gender and Age as 18.

INSERT INTO Student (Admno, Name, Place) VALUES (104, 'Abinandh', 'Chennai');

104	Abinandh	M	18	Chennai

The student table will have the following data:

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	17	Bangalore
103	Ayush	M	18	Delhi
104	Abinandh	M	18	Chennai

The fields that are not given in the INSERT command will take default values, if it is defined for them, otherwise NULL value will be stored.

DO YOU KNOW

In the INSERT command the fields that are omitted will have either default value defined or NULL value.

12.7.3.2 DELETE COMMAND

The **DELETE** command permanently removes one or more records from the table. It removes the entire row, not individual fields of the row, so no field argument is needed. The **DELETE** command is used as follows :

DELETE FROM table-name WHERE condition;

For example to delete the record whose admission number is 104 the command is given as follows:

DELETE FROM Student WHERE Admno=104;

104	Abinandh	M	18	Chennai

The following record is deleted from the Student table.

To delete all the rows of the table, the command is used as :

DELETE * FROM Student;

The table will be empty now and could be destroyed using the DROP command (Discussed in section 12.7.4.3).

XII Std - CS EM Chapter-12.indd 216 24-01-2020 11:20:32

12.7.3.3 UPDATE COMMAND

The **UPDATE** command updates some or all data values in a database. It can update one or more records in a table. The **UPDATE** command specifies the rows to be changed using the **WHERE** clause and the new data using the **SET** keyword. The command is used as follows:

> ***UPDATE <table-name> SET column-name = value, column-name = value,… WHERE condition;***

For example to update the following fields:

UPDATE Student SET Age = 20 WHERE Place = "Bangalore";

The above command will change the age to 20 for those students whose place is "Bangalore".

The table will be as updated as below:

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	20	Bangalore
103	Ayush	M	18	Delhi

To update multiple fields, multiple field assignment can be specified with the **SET** clause separated by comma. For example to update multiple fields in the Student table, the command is given as:

UPDATE Student SET Age=18, Place = 'Chennai' WHERE Admno = 102;

102	Akshith	M	18	Chennai

The above command modifies the record in the following way.

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	18	Chennai
103	Ayush	M	18	Delhi

12.7.4 Some Additional DDL Commands:

12.7.4.1 ALTER COMMAND

The **ALTER** command is used to alter the table structure like adding a column, renaming the existing column, change the data type of any column or size of the column or delete the column from the table. It is used in the following way :

> ***ALTER TABLE <table-name> ADD <column-name><data type><size>;***

XII Std - CS EM Chapter-12.indd 217 24-01-2020 11:20:32

To add a new column "Address" of type 'char' to the Student table, the command is used as

ALTER TABLE Student ADD Address char;

To modify existing column of table, the **ALTER TABLE** command can be used with **MODIFY** clause like wise:

ALTER <table-name> MODIFY<column-name><data type><size>;

ALTER TABLE Student MODIFY Address char (25);

The above command will modify the address column of the Student table to now hold 25 characters.

The **ALTER** command can be used to rename an existing column in the following way :

ALTER <table-name> RENAME old-column-name TO new-column-name;

For example to rename the column Address to City, the command is used as :

ALTER TABLE Student RENAME Address TO City;

The **ALTER** command can also be used to remove a column or all columns, for example to remove a particular column, the **DROP COLUMN** is used with the **ALTER TABLE** to remove a particular field, the command can be used as:

ALTER <table-name> DROP COLUMN <column-name>;

To remove the column City from the Student table, the command is used as :

ALTER TABLE Student DROP COLUMN City;

12.7.4.2 TRUNCATE command

The **TRUNCATE** command is used to delete all the rows from the table, the structure remains and the space is freed from the table. The syntax for **TRUNCATE** command is:

TRUNCATE TABLE table-name;

For example to delete all the records of the student table and delete the table the SQL statement is given as follows:

TRUNCATE TABLE Student;

The table Student is removed and the space is freed.

12.7.4.3 DROP TABLE command

The **DROP TABLE** command is used to remove a table from the database. If you drop a table, all the rows in the table is deleted and the table structure is removed from the database. Once a table is dropped we cannot get it back, so be careful while using **DROP TABLE** command. But there is a condition for dropping a table; it must be an empty table.

XII Std - CS EM Chapter-12.indd 218 24-01-2020 11:20:32

Remove all the rows of the table using **DELETE** command. The **DELETE** command is already explained.

To delete all rows, the command is given as :

DELETE * FROM Student;

Once all the rows are deleted, the table can be deleted by **DROP TABLE** command in the following way:

DROP TABLE table-name;

For example to delete the Student table:

DROP TABLE Student;

DELETE, TRUNCATE AND DROP statement:

DELETE	The **DELETE** command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.
TRUNCATE	The **TRUNCATE** command is used to delete all the rows, the structure remains in the table and free the space containing the table.
DROP	The **DROP** command is used to remove an object from the database. If you drop a table, all the rows in the table is deleted and the table structure is removed from the database. Once a table is dropped we cannot get it back.

12.7.5 DQL COMMAND– SELECT command

One of the most important tasks when working with SQL is to generate Queries and retrieve data. A Query is a command given to get a desired result from the database table. The **SELECT** command is used to query or retrieve data from a table in the database. It is used to retrieve a subset of records from one or more tables. The **SELECT** command can be used in various forms:

Syntax of **SELECT** command :

SELECT <column-list>FROM<table-name>;

- Table-name is the name of the table from which the information is retrieved.
- Column-list includes one or more columns from which data is retrieved.

For example to view only admission number and name of students from the Student table the command is given as follows:

XII Std - CS EM Chapter-12.indd 219 24-01-2020 11:20:32

If the Student table has the following data:

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	17	Bangalore
103	Ayush	M	18	Delhi
104	Abinandh	M	18	Chennai
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
107	Hema	F	17	Chennai

SELECT Admno, Name FROM Student;

The above **SELECT** command will display the following data:

Admno	Name
100	Ashish
101	Adarsh
102	Akshith
103	Ayush
104	Abinandh
105	Revathi
106	Devika
107	Hema

To view all the fields and rows of the table the **SELECT** command can be given as

SELECT * FROM STUDENT;

12.7.5.1 DISTINCT Keyword

The **DISTINCT** keyword is used along with the **SELECT** command to eliminate duplicate rows in the table. This helps to eliminate redundant data. For Example:

SELECT DISTINCT Place FROM Student;

Will display the following data as follows :

SELECT * FROM Student;

Place
Chennai
Bangalore
Delhi

In the above output you can see, there would be no duplicate rows in the place field. When the keyword **DISTINCT** is used, only one **NULL** value is returned, even if more **NULL** values occur.

XII Std - CS EM Chapter-12.indd 220 24-01-2020 11:20:32

12.7.5.2 ALL Keyword

The **ALL** keyword retains duplicate rows. It will display every row of the table without considering duplicate entries.

SELECT ALL Place FROM Student;

The above command will display all values of place field from every row of the table without considering the duplicate entries.

Place
Chennai
Delhi
Bangalore
Delhi
Chennai
Chennai
Bangalore
Chennai

The **WHERE** clause in the **SELECT** command specifies the criteria for getting the desired result. The general form of **SELECT** command with **WHERE** Clause is:

SELECT <column-name>[,<column-name>,....] FROM <table-name>WHERE condition>;

For example to display the students admission number and name of only those students who belong to Chennai, the **SELECT** command is used in the following way :

SELECT Admno, Name, Place FROM Student WHERE Place ="Chennai";

Admno	Name	Place
100	Ashish	Chennai
104	Abinandh	Chennai
105	Revathi	Chennai
107	Hema	Chennai

SELECT Admno, Name, Age FROM Student WHERE Age >= 18;

Admno	Name	Age
101	Adarsh	18
103	Ayush	18
104	Abinandh	18
105	Revathi	19
106	Devika	19

The relational operators like =, <, <=, >, >=, <> can be used to compare two values in the **SELECT** command used with **WHERE** clause. The logical operaors **OR, AND** and **NOT**

XII Std - CS EM Chapter-12.indd 221 24-01-2020 11:20:32

can also be used to connect search conditions in the **WHERE** clause. For example :

SELECT Admno, Name, Age, Place FROM Student WHERE (Age>=18 AND Place = "Delhi");

Admno	Name	Age	Place
101	Adarsh	18	Delhi
103	Ayush	18	Delhi

The **SELECT** command can also be used in the following ways:

SELECT Admno, Name, Age, Place FROM Student WHERE (Age>=18 OR Place ="Delhi");

SELECT Admno, Name, Place FROM Student WHERE (NOT Place ="Delhi");

12.7.5.3 BETWEEN and NOT BETWEEN Keywords

The **BETWEEN** keyword defines a range of values the record must fall into to make the condition true. The range may include an upper value and a lower value between which the criteria must fall into.

SELECT Admno, Name, Age, Gender FROM Student WHERE Age BETWEEN 18 AND 19;

Admno	Name	Gender	Age	Place
101	Adarsh	M	18	Delhi
103	Ayush	M	18	Delhi
104	Abinandh	M	18	Chennai
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore

The **NOT BETWEEN** is reverse of the **BETWEEN** operator where the records not satisfying the condition are displayed.

SELECT Admno, Name, Age FROM Student WHERE Age NOT BETWEEN 18 AND 19;

Admno	Name	Age
100	Ashish	17
102	Akshith	17
107	Hema	17

12.7.5.4 IN Keyword

The **IN** keyword is used to specify a list of values which must be matched with the record values. In other words it is used to compare a column with more than one value. It is similar to an **OR** condition.

For example :

SELECT Admno, Name, Place FROM Student WHERE Place IN ("Chennai", "Delhi");

XII Std - CS EM Chapter-12.indd 222 24-01-2020 11:20:32

Admno	Name	Place
100	Ashish	Chennai
101	Adarsh	Delhi
103	Ayush	Delhi
104	Abinandh	Chennai
105	Revathi	Chennai
107	Hema	Chennai

The **NOT IN** keyword displays only those records that do not match in the list.

For example:

SELECT Admno, Name, Place FROM Student WHERE Place NOT IN ("Chennai", "Delhi");

will display students only from places other than "Chennai" and "Delhi".

Admno	Name	Place
102	Akshith	Bangalore
106	Devika	Bangalore

NULL Value :

The **NULL** value in a field can be searched in a table using the **IS NULL** in the **WHERE** clause. For example to list all the students whose Age contains no value, the command is used as:

SELECT * FROM Student WHERE Age IS NULL;

Note

Non NULL values in a table can be listed using IS NOT NULL.

12.7.5.5 ORDER BY clause

The **ORDER BY** clause in SQL is used to sort the data in either ascending or descending based on one or more columns.

1. By default **ORDER BY** sorts the data in ascending order.
2. We can use the keyword **DESC** to sort the data in descending order and the keyword **ASC** to sort in ascending order.

The ORDER BY clause is used as :

SELECT <column-name>[,<column-name>,....] FROM <table-name>ORDER BY <column1>,<column2>,...ASC| DESC ;

XII Std - CS EM Chapter-12.indd 223 24-01-2020 11:20:33

For example :

To display the students in alphabetical order of their names, the command is used as

SELECT * FROM Student ORDER BY Name;

The above student table is arranged as follows :

Admno	Name	Gender	Age	Place
104	Abinandh	M	18	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	17	Bangalore
100	Ashish	M	17	Chennai
103	Ayush	M	18	Delhi
106	Devika	F	19	Bangalore
107	Hema	F	17	Chennai
105	Revathi	F	19	Chennai

Note

The ORDER BY clause does not affect the original table.

12.7.5.6 WHERE clause

The **WHERE** clause is used to filter the records. It helps to extract only those records which satisfy a given condition. For example in the student table, to display the list of students of age18 and above in alphabetical order of their names, the command is given as below:

SELECT * FROM Student WHERE Age>=18 ORDER BY Name;

Admno	Name	Gender	Age	Place
104	Abinandh	M	18	Chennai
101	Adarsh	M	18	Delhi
103	Ayush	M	18	Delhi
106	Devika	F	19	Bangalore
105	Revathi	F	19	Chennai

To display the list of students in the descending order of names of those students of age 18 and above the command is given as :

SELECT * FROM Student WHERE Age>=18 ORDER BY Name DESC;

XII Std - CS EM Chapter-12.indd 224 24-01-2020 11:20:33

Admno	Name	Gender	Age	Place
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai

Note

Sorting can be done on multiple fields.

12.7.5.7 GROUP BY clause

The **GROUP BY** clause is used with the **SELECT** statement to group the students on rows or columns having identical values or divide the table in to groups. For example to know the number of male students or female students of a class, the **GROUP BY** clause may be used. It is mostly used in conjunction with aggregate functions to produce summary reports from the database.

The syntax for the **GROUP BY** clause is

SELECT <column-names> FROM <table-name> GROUP BY <column-name>HAVING condition];

To apply the above command on the student table :

SELECT Gender FROM Student GROUP BY Gender;

The following command will give the below given result:

Gender
M
F

The point to be noted is that only two results have been returned. This is because we only have two gender types 'Male' and 'Female'. The GROUP BY clause grouped all the 'M' students together and returned only a single row for it. It did the same with the 'F' students.

For example to count the number of male and female students in the student table, the following command is given :

XII Std - CS EM Chapter-12.indd 225 24-01-2020 11:20:33

SELECT Gender, count(*) FROM Student GROUP BY Gender;

Gender	count(*)
M	5
F	3

Note

The * is used with the COUNT to include the NULL values.

The GROUP BY applies the aggregate functions independently to a series of groups that are defined by having a field value in common. The output of the above SELECT statement gives a count of the number of Male and Female students.

12.7.5.8 HAVING clause

The **HAVING** clause can be used along with GROUP BY clause in the SELECT statement to place condition on groups and can include aggregate functions on them. For example to count the number of Male and Female students belonging to Chennai .

SELECT Gender , count(*) FROM Student GROUP BY Gender HAVING Place = 'Chennai';

Gender	count(*)
M	2
F	2

The above output shows the number of Male and Female students in Chennai from the table student.

12.7.6 TCL commands

12.7.6.1 COMMIT command

The **COMMIT** command is used to permanently save any transaction to the database. When any DML commands like **INSERT, UPDATE, DELETE** commands are used, the changes made by these commands are not permanent. It is marked permanent only after the **COMMIT** command is given from the SQL prompt. Once the **COMMIT** command is given, the changes made cannot be rolled back. The **COMMIT** command is used as

COMMIT;

12.7.6.2 ROLLBACK command

The **ROLLBACK** command restores the database to the last commited state. It is used with **SAVEPOINT** command to jump to a particular savepoint location. The syntax for the **ROLLBACK** command is :

ROLL BACK TO save point name;

XII Std - CS EM Chapter-12.indd 226 24-01-2020 11:20:33

12.7.6.3 SAVEPOINT command

The **SAVEPOINT** command is used to temporarily save a transaction so that you can rollback to the point whenever required. The different states of our table can be saved at anytime using different names and the rollback to that state can be done using the **ROLLBACK** command.

```
SAVEPOINT savepoint_name;
```

Example showing **COMMIT, SAVEPOINT** and **ROLLBACK** in the student table having the following data:

Admno	Name	Gender	Age	Place
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai

```
INSERT INTO Student  VALUES (107, 'Beena', 'F', 20 , 'Cochin');
COMMIT;
```

Admno	Name	Gender	Age	Place
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai
107	Beena	F	20	Cochin

```
UPDATE Student SET Name = 'Mini' WHERE  Admno=105;
SAVEPOINT A;
```

Admno	Name	Gender	Age	Place
105	Mini	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai
107	Beena	F	20	Cochin

XII Std - CS EM Chapter-12.indd 227 24-01-2020 11:20:33

INSERT INTO Student VALUES(108, 'Jisha', 'F', 19, 'Delhi');

SAVEPOINT B;

Admno	Name	Gender	Age	Place
105	Mini	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	vAbinandh	M	18	Chennai
107	Beena	F	20	Cochin
108	Jisha	F	19	Delhi

ROLLBACK TO A;

Admno	Name	Gender	Age	Place
105	Mini	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai
107	Beena	F	20	Cochin

Points to remember:

- ***SQL*** is a language that helps to create and operate relational databases.
- ***MySQL*** is a database management system.
- The various components of ***SQL*** are Data Definition Language (***DDL***), Data Manipulation Language (***DML***), Data Query Language (***DQL***), Transactional Control Language (***TCL***), Data Control Language (***DCL***).
- The ***DDL*** provides statements for creation and deletion of tables.
- The ***DML*** provides statements to insert, update and delete data of a table.
- The ***DCL*** provides authorization commands to access data.
- The ***TCL*** commands are used to manage transactions in a database.
- The ***DQL*** commands help to generate queries in a database.
- The ***CREATE TABLE*** command creates a new table.

XII Std - CS EM Chapter-12.indd 228 24-01-2020 11:20:33

Points to remember:

- ***SQL*** is a language that helps to create and operate relational databases.
- ***MySQL*** is a database management system.
- The various components of ***SQL*** are Data Definition Language (***DDL***), Data Manipulation Language (***DML***), Data Query Language (***DQL***), Transactional Control Language (***TCL***), Data Control Language (***DCL***).
- The ***DDL*** provides statements for creation and deletion of tables.
- The ***DML*** provides statements to insert, update and delete data of a table.
- The ***DCL*** provides authorization commands to access data.
- The ***TCL*** commands are used to manage transactions in a database.
- The ***DQL*** commands help to generate queries in a database.
- The ***CREATE TABLE*** command creates a new table.

Hands on Experience

1. Create a query of the student table in the following order of fields name, age, place and admno.
2. Create a query to display the student table with students of age more than 18 with unique city.

Hands on Experience

1. Create a employee table with the following fields employee number, employee name, designation, date of joining and basic pay.
2. In the above table set the employee number as primary key and check for NULL values in any field.
3. Prepare a list of all employees who are Managers.

XII Std - CS EM Chapter-12.indd 229 24-01-2020 11:20:33

LA6AGJ

Evaluation

Part - I

I Choose the best answer **1X 5 = 5**

1. Which commands provide definitions for creating table structure, deleting relations, and modifying relation schemas.
   a. DDL b. DML
   c. DCL d. DQL
2. Which command lets to change the structure of the table?
   a. SELECT b. ORDER BY
   c. MODIFY d. ALTER
3. The command to delete a table is
   A) DROP B) DELETE
   C) DELETE ALL D) ALTER TABLE
4. Queries can be generated using
   a. SELECT b. ORDER BY
   c. MODIFY d. ALTER
5. The clause used to sort data in a database
   a. SORT BY b. ORDER BY
   c. GROUP BY d. SELECT

Part -II

II Write Short Answers **2 x 5 = 10**

1. Write a query that selects all students whose age is less than 18 in order wise.
2. Differentiate Unique and Primary Key constraint.
3. Write the difference between table constraint and column constraint?
4. Which component of SQL lets insert values in tables and which lets to create a table?
5. What is the difference between SQL and MySQL?

XII Std - CS EM Chapter-12.indd 230 24-01-2020 11:20:33

Part -III

III Explain 3 x 5 = 15

1. What is a constraint? Write short note on Primary key constraint.
2. Write a SQL statement to modify the student table structure by adding a new field.
3. Write any three DDL commands.
4. Write the use of Savepoint command with an example.
5. Write a SQL statement using DISTINCT keyword.

Part -IV

IV Explain in Detail 5 x 5 = 25

1. Write the different types of constraints and their functions.
2. Consider the following employee table. Write SQL commands for the qtns.(i) to (v).

EMP CODE	NAME	DESIG	PAY	ALLO WANCE
S1001	Hariharan	Supervisor	29000	12000
P1002	Shaji	Operator	10000	5500
P1003	Prasad	Operator	12000	6500
C1004	Manjima	Clerk	8000	4500
M1005	Ratheesh	Mechanic	20000	7000

(i) To display the details of all employees in descending order of pay.

(ii) To display all employees whose allowance is between 5000 and 7000.

(iii) To remove the employees who are mechanic.

(iv) To add a new row.

(v) To display the details of all employees who are operators.

3. What are the components of SQL? Write the commands in each.
4. Construct the following SQL statements in the student table-
   (i) SELECT statement using GROUP BY clause.
   (ii) SELECT statement using ORDER BY clause.
5. Write a SQL statement to create a table for employee having any five fields and create a table constraint for the employee table.

XII Std - CS EM Chapter-12.indd 231 24-01-2020 11:20:33

Unit IV

CHAPTER 13

PYTHON AND CSV FILES

Learning Objectives

After the completion of this chapter, the student will be able to

- Understand what is CSV?
- Able to import CSV files in python programs
- Execute and debug python programs

M4Y4HJ

13.1 Introduction

Python has a vast library of modules that are included with its distribution. One among the module is the **CSV module** which gives the Python programmer the ability to parse **CSV (Comma Separated Values)** files. **A CSV file is a human readable text file where each line has a number of fields, separated by commas or some other delimiter.** You can assume each line as a row and each field as a column. The CSV module will be able to read and write the vast majority of CSV files.

13.2 Difference between CSV and XLS file formats

The difference between **Comma-Separated** Values (CSV) and e**X**ce**L** **S**heets(XLS) file formats is

Excel	CSV
Excel is a binary file that holds information about all the worksheets in a file, including both content and formatting	CSV format is a plain text format with a series of values separated by commas.
XLS files can only be read by applications that have been especially written to read their format, and can only be written in the same way.	CSV can be opened with any text editor in Windows like notepad, MS Excel, OpenOffice, etc.
Excel is a spreadsheet that saves files into its own proprietary format viz. xls or xlsx	CSV is a format for saving tabular information into a delimited text file with extension .csv
Excel consumes more memory while importing data	Importing CSV files can be much faster, and it also consumes less memory

XII Std - CS EM Chapter-13.indd 232 24-01-2020 12:50:36

DO YOU KNOW?

Files saved in excel cannot be opened or edited by text editors.

13.3 Purpose Of CSV File

CSV is a simple **file format** used to store tabular data, such as a spreadsheet or database. Since they're plain text, they're easier to import into a spreadsheet or another storage database, regardless of the specific software you're using.

You can open CSV files in a spreadsheet program like Microsoft Excel or in a text editor or through a database which make them easier to read.

Note

CSV File cannot store charts or graphs. It stores data but does not contain formatting, formulas, macros, etc.

DO YOU KNOW?

A CSV file is also known as a Flat File. Files in the CSV format can be imported to and exported from programs that store data in tables, such as ***Microsoft Excel*** or ***OpenOfficeCalc***

13.4 Creating a CSV file using Notepad (or any text editor)

A CSV file is a text file, so it can be created and edited using any text editor, But more frequently a CSV file is created by exporting a spreadsheet or database in the program that created it.

13.4.1 Creating CSV Normal File

To create a CSV file in Notepad, First open a new file using

File →New or ctrl +N.

Then enter the data you want the file to contain, separating each value with a comma and each row with a new line.

For example consider the following details

```
Topic1,Topic2,Topic3
one,two,three
Example1,Example2,Example3
```

Save this content in a file with the extension .csv . You can then open the same using Microsoft Excel or any other spreadsheet program. Here we have opened using Microsoft Excel. It would create a table of data similar to the following:

XII Std - CS EM Chapter-13.indd 233 24-01-2020 12:50:36

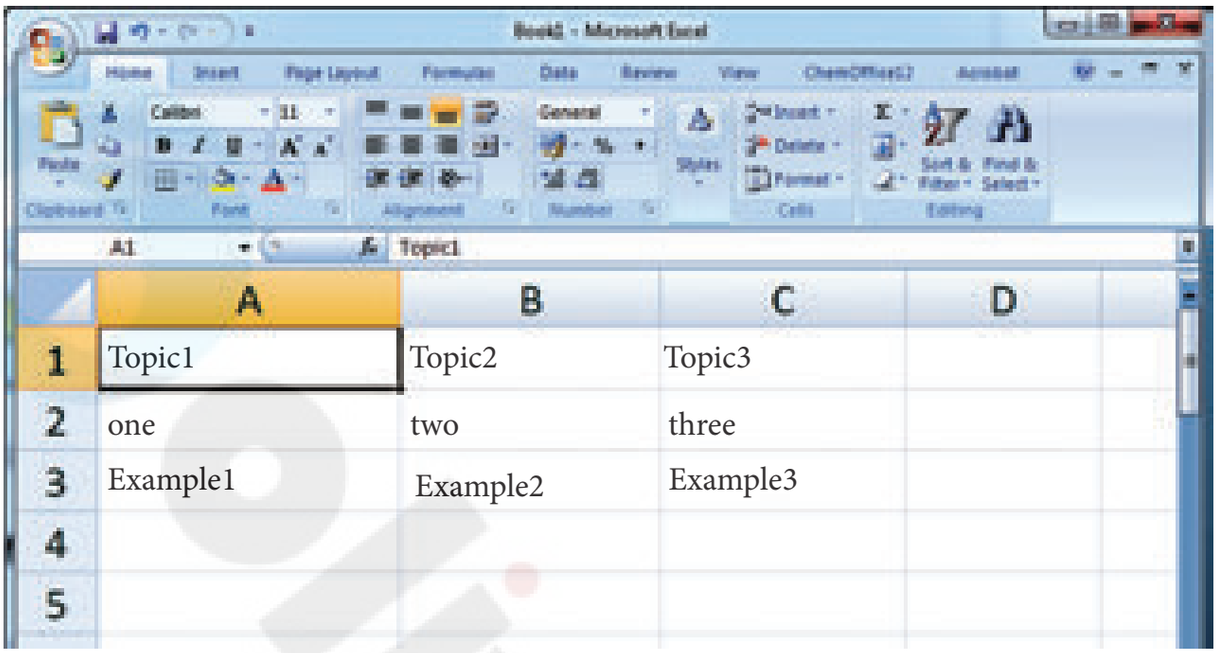


Fig. 13.4.1 CSV file when opened in MS-Excel

In the above CSV file, you can observe the fields of data were separated by commas. But what happens if the data itself contains commas in it?

If the fields of data in your CSV file contain commas, you can protect them by enclosing those data fields in double-quotes (“). ***The commas that are part of your data will then be kept separate from the commas which delimit the fields themselves.***

13.4.2 Creating CSV File That contains Comma With Data

For example, let’s say that one of our fields contain commas in the description. If our data looked like the below example:

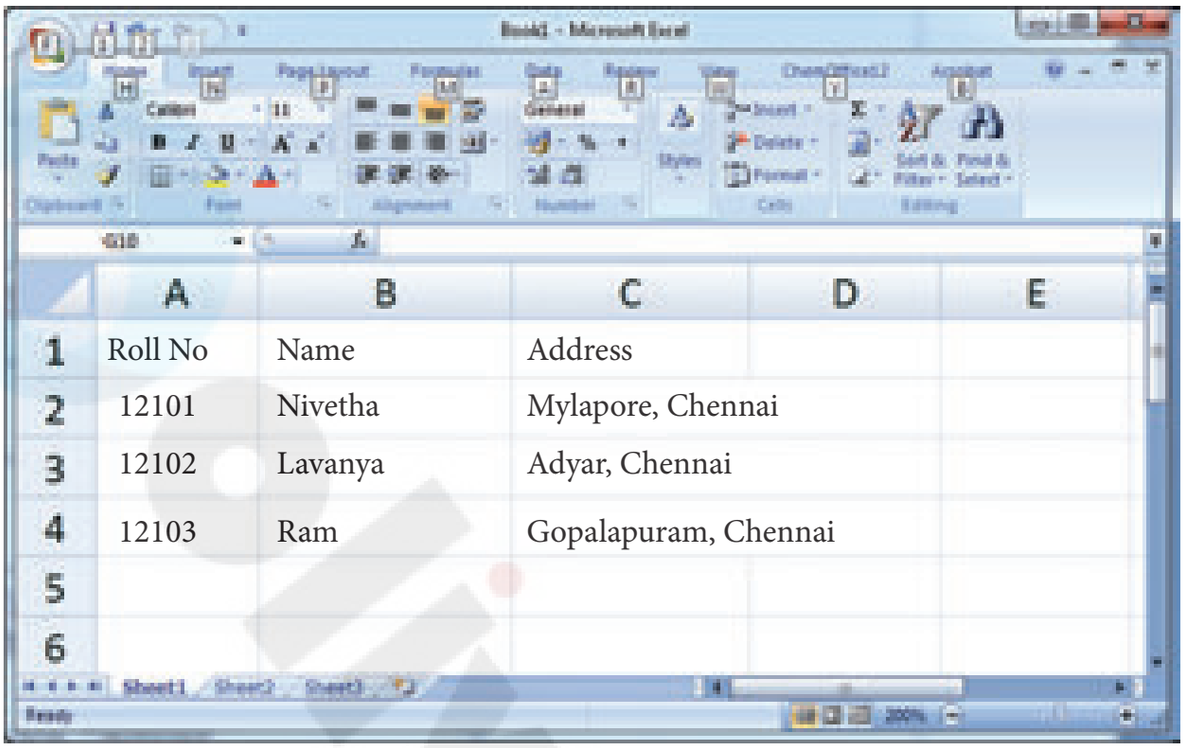
RollNo	Name	Address
12101	Nivetha	Mylapore, Chennai
12102	Lavanya	Adyar, Chennai
12103	Ram	Gopalapuram, Chennai

To retain the commas in “Address” column, you can enclose the fields in quotation marks. For example:

```
RollNo, Name, Address
12101, Nivetha, “Mylapore, Chennai”
12102, Lavanya, “Adyar, Chennai”
12103, Ram, “Gopalapuram, Chennai”
```

As you can see, only the fields that contain commas are enclosed in quotes. If you open this in MS Excel, It looks like as follows

XII Std - CS EM Chapter-13.indd 234 24-01-2020 12:50:36



	A	B	C	D	E
1	Roll No	Name	Address		
2	12101	Nivetha	Mylapore, Chennai		
3	12102	Lavanya	Adyar, Chennai		
4	12103	Ram	Gopalapuram, Chennai		
5					
6					

Fig 13.4.2 (a) CSV Field data with comma in Excel

The same goes for newlines (display data in more than one line example Address column) which may be part of your field data. Any fields containing a newline as part of its data need to be enclosed in double-quotes.

For Example

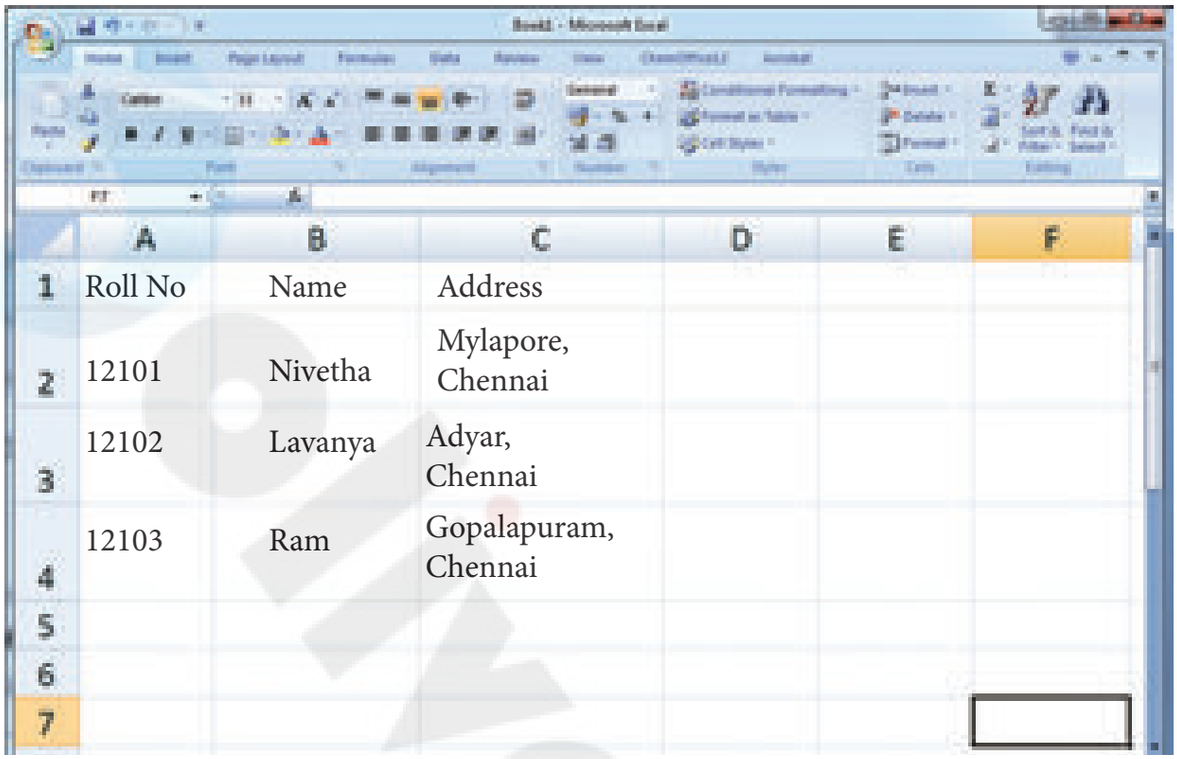
RollNo	Name	Address
12101	Nivetha	Mylapore, Chennai
12102	Lavanya	Adyar, Chennai
12103	Ram	Gopalapuram, Chennai

It should be written in CSV file as

```
RollNo, Name, Address
12101, Nivetha, "Mylapore
Chennai"
12102, Lavanya, "Adyar
Chennai"
12103, Ram, "Gopalapuram
Chennai"
```


XII Std - CS EM Chapter-13.indd 235 24-01-2020 12:50:36

The Result will look like this



	A	B	C	D	E	F
1	Roll No	Name	Address			
2	12101	Nivetha	Mylapore, Chennai			
3	12102	Lavanya	Adyar, Chennai			
4	12103	Ram	Gopalapuram, Chennai			
5						
6						
7						

Fig 13.4.2 (b) CSV Field Data with newline in Excel

13.4.3 Creating CSV File That contains Double Quotes With Data

If your fields contain double-quotes as part of their data, **the internal quotation marks need to be doubled so that they can be interpreted correctly.** For Example, given the following data:

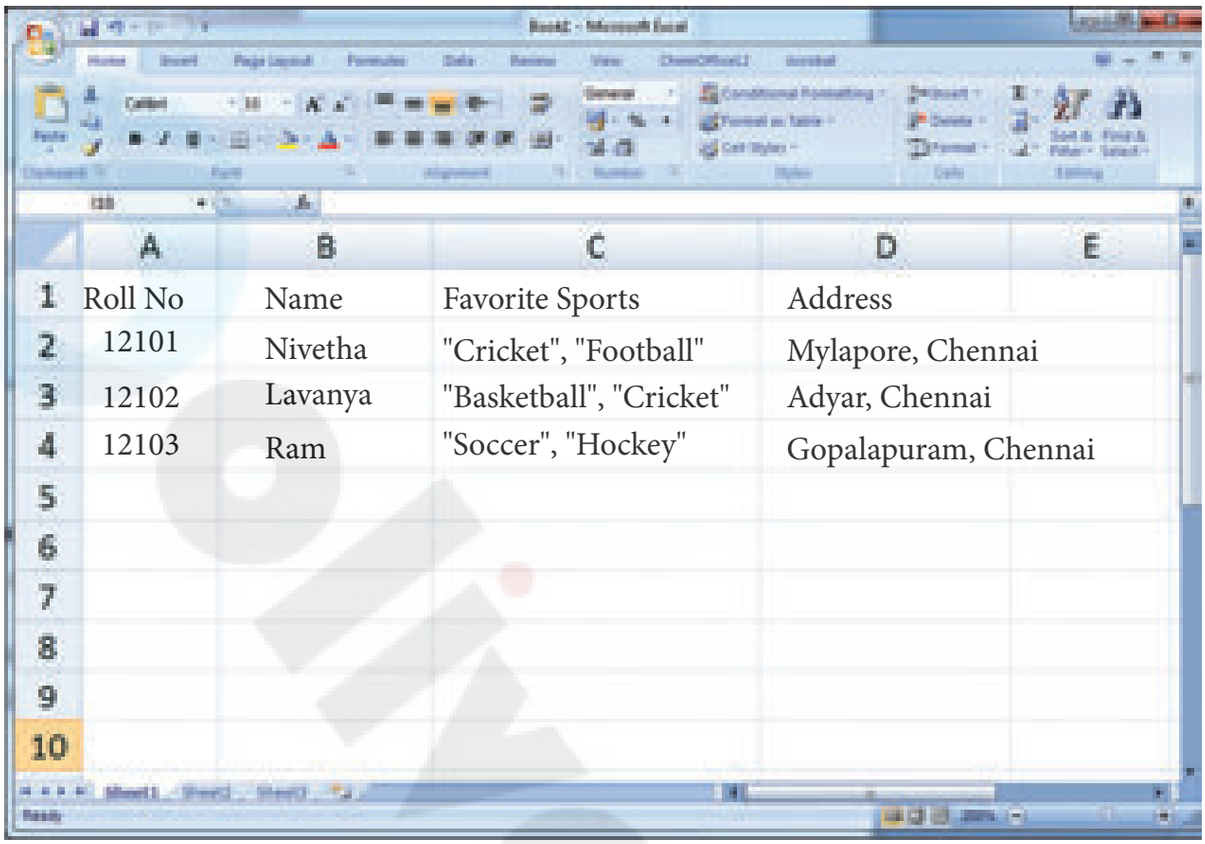
Roll No	Name	Favorite Sports	Address
12101	Nivetha	"Cricket", "Football"	Mylapore Chennai
12102	Lavanya	"Basketball", "Cricket"	Adyar Chennai
12103	Ram	"Soccer", "Hockey"	Gopala puram Chennai

It should be written in csv file as

```
RollNo, Name, FavoriteSports, Address
12101, Nivetha,"""" Cricket "","" Football """", Mylapore chennai
12102, Lavanya,"""" Basketball "","" Cricket """", Adyar chennai
12103, Ram,"""" Soccer"","" Hockey"""", Gopalapuram chennai
```

The output will be

XII Std - CS EM Chapter-13.indd 236 24-01-2020 12:50:36



	A	B	C	D	E
1	Roll No	Name	Favorite Sports	Address	
2	12101	Nivetha	"Cricket", "Football"	Mylapore, Chennai	
3	12102	Lavanya	"Basketball", "Cricket"	Adyar, Chennai	
4	12103	Ram	"Soccer", "Hockey"	Gopalapuram, Chennai	
5					
6					
7					
8					
9					
10					

Fig 13.4.3 CSV Field Data with Double quotes in Excel

13.4.4 Rules to be followed to format data in a CSV file

1. Each record (row of data) is to be located on a separate line, delimited by a line break by pressing enter key. For example:↵

   xxx,yyy↵

   ↵ denotes enter Key to be pressed

2. The last record in the file may or may not have an ending line break. For example:

   ppp, qqq ↵

   yyy, xxx

3. There may be an optional header line appearing as the first line of the file with the same format as normal record lines. The header will contain names corresponding to the fields in the file and should contain the same number of fields as the records in the rest of the file. For example: field_name1,field_name2,field_name3 ↵

   aaa,bbb,ccc ↵

   zzz,yyy,xxx CRLF(**Carriage Return and Line Feed)**

XII Std - CS EM Chapter-13.indd 237 24-01-2020 12:50:36

4. Within the header and each record, there may be one or more fields, separated by commas. Spaces are considered part of a field and should not be ignored. The last field in the record must not be followed by a comma. For example: Red , Blue

5. Each field may or may not be enclosed in double quotes. If fields are not enclosed with double quotes, then double quotes may not appear inside the fields. For example:

```
"Red","Blue","Green"↵      #Field data with doule quotes
Black,White,Yellow         #Field data without doule quotes
```

6. Fields containing line breaks (CRLF), double quotes, and commas should be enclosed in double-quotes. For example:

```
Red, ",", Blue CRLF    # comma itself is a field value.so it is enclosed with double quotes
Red, Blue , Green
```

7. If double-quotes are used to enclose fields, then a double-quote appearing inside a field must be preceded with another double quote. For example:

```
"Red, " "Blue", "Green",    # since double quotes is a field value it is enclosed with another double quotes
, ,  White
```

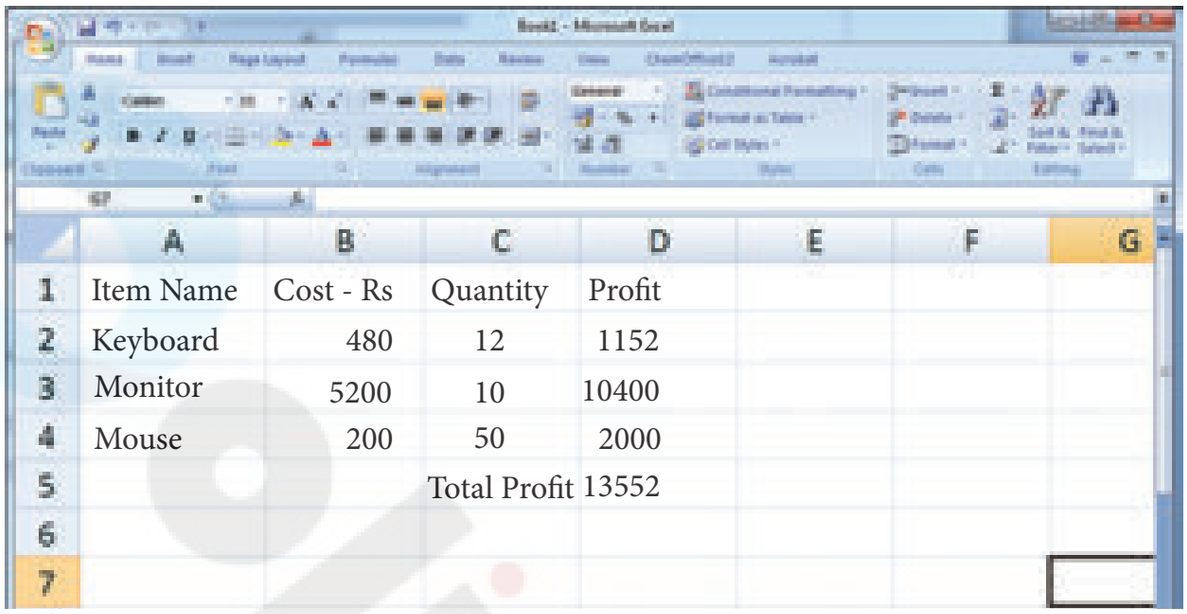
Note

The last row in the above example (, , White) begins with two commas because the first two fields of that row were empty in our spreadsheet. Don't delete them — the two commas are required so that the fields correspond from row to row. They cannot be omitted.

13.5 Create A CSV File Using Microsoft Excel

To create a CSV file using Microsoft Excel, launch Excel and then open the file you want to save in CSV format. For example, below is the data contained in our sample Excel worksheet:

XII Std - CS EM Chapter-13.indd 238 24-01-2020 12:50:36



	A	B	C	D
1	Item Name	Cost - Rs	Quantity	Profit
2	Keyboard	480	12	1152
3	Monitor	5200	10	10400
4	Mouse	200	50	2000
5			Total Profit	13552
6				
7				

Fig 13.5 Sample Worksheet Data

Once the data is entered in the worksheet, select **File → Save** As option, and for the "Save as type option", select CSV (Comma delimited) or type the file name along with extension .csv.

Saving excel file as CSV

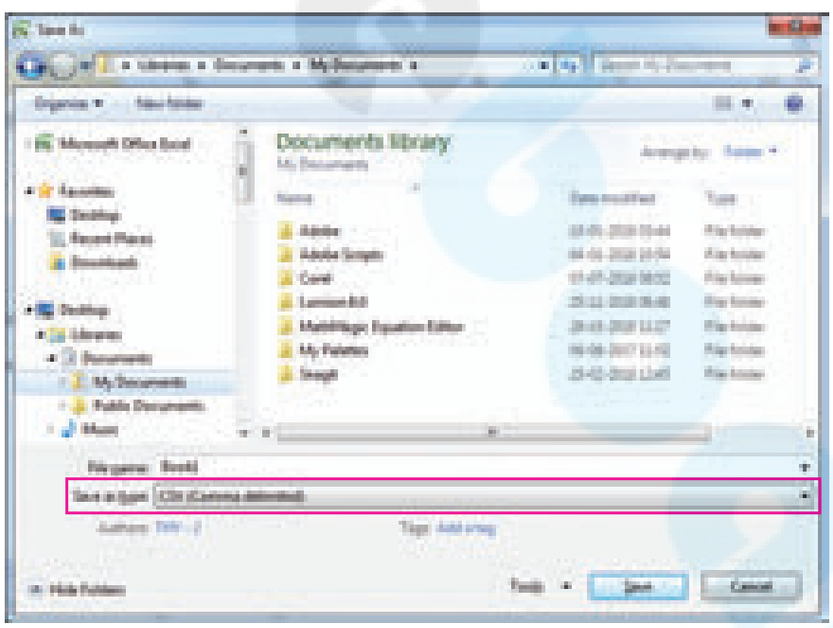


Fig 13.6 Save As dialog box

After you save the file, you are free to open it up in a text editor to view it or to edit it manually. Its contents will resemble the following:

XII Std - CS EM Chapter-13.indd 239 24-01-2020 12:50:36

```
Item Name, Cost-Rs, Quantity, Profit
Keyboard, 480, 12, 1152
Monitor, 5200, 10, 10400
Mouse, 200, 50, 2000
,,Total Profit =,13552
```

13.5.1 Microsoft Excel to open a CSV file

If Microsoft Excel has been installed on the computer, **by default CSV files should open automatically in Excel when the file is double-clicked.** If you are getting an **Open With** prompt when opening the CSV file, choose Microsoft Excel from the available programs to open the file.

Alternatively, you can open Microsoft Excel and in the menu bar, select **File → Open,** and select the CSV file. If the file is not listed, make sure to change the file type to be opened to Text Files (*.prn, *.txt, *.csv).

DO YOU KNOW?

If both MS Excel and Open Office calc is installed in the computer, by default the CSV file will be opened in MS Excel.

13.6 Read and write a CSV file Using Python

Python provides a module named **CSV,** using this you can do several operations on the CSV files. The CSV library contains objects and other code to **read, write, and process** data from and to CSV files.

DO YOU KNOW?

CSV files have been used extensively in e-commerce applications because they are considered very easy to process.

13.6.1 Read a CSV File Using Python

There are two ways to read a CSV file.

1. Use the csv module's reader function
2. Use the DictReader class.

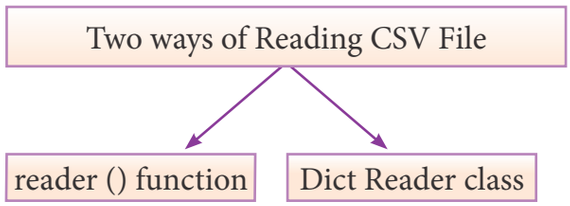


Fig 13.7 Ways to read CSV file

XII Std - CS EM Chapter-13.indd 240 24-01-2020 12:50:36

When you want to read from or write to a file ,you need to open it. Once the reading is over it needs to be closed. So that, resources that are tied with the file are freed. Hence, in Python, a file operation takes place in the following order

Step 1 Open a file

Step 2 Perform Read or write operation

Step 3 Close the file

Note

File name or the complete path name can be represented either with in " " or in ' ' in the open command.

Python has a built-in function open() to open a file. This function returns a **file object,** also called a **handle,** as it is used to read or modify the file accordingly.

For Example

```
>>> f = open("sample.txt")  # open file in current directory and f is file object
>>> f = open('c:\pyprg\ch13sample5.csv')# specifying full path
```

You can specify the mode while opening a file. In mode, you can specify whether you want to read **'r',** write **'w'** or append **'a'** to the file. you can also specify "text or binary" in which the file is to be opened.

The default is reading in text mode. In this mode, while reading from the file the data would be in the format of **strings.**

On the other hand, **binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.**

Python File Modes

Mode	Description
'r'	Open a file for reading. (default)
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.

XII Std - CS EM Chapter-13.indd 241 24-01-2020 12:50:36

'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)

```
f=open("sample.txt")
#equivalent to 'r' or 'rt'
f = open("sample.txt",'w')          # write in text mode
f = open("image1.bmp",'r+b')        # read and write in binary mode
```

Python has a garbage collector to clean up unreferenced objects but, one must not rely on it to close the file.

```
f = open("test.txt")        #  since no mode is specified the default mode rt is used
# perform file operations
f.close()
```

The above method is **not entirely safe.** If an **exception** occurs when you are performing some operation with the file, the code exits without closing the file. The best way to do this is using the **"with"** statement. This ensures that the file is closed when the block inside **with** is exited. You need not to explicitly call the close() method. It is done internally.

```
with open("test.txt",'r') as f:
# f is file object to perform file operations
```

Closing a file will free up the resources that were tied with the file and is done using Python close() method.

```
f = open("sample.txt")
# perform file operations
f.close()
```

13.6.1.1 CSV Module's Reader Function

You can read the contents of CSV file with the help of **csv.reader()** method. **The reader function is designed to take each line of the file and make a list of all columns.** Then, you just choose the column you want the variable data for. Using this method one can read data from csv files of different formats like quotes (" "), pipe (|) and comma (,).

The syntax for csv.reader() is

XII Std - CS EM Chapter-13.indd 242 24-01-2020 12:50:36

```
csv.reader(fileobject,delimiter,fmtparams)
```

where

file object :- passes the path and the mode of the file

delimiter :- an optional parameter containing the standard dilects like , | etc can be omitted

fmtparams: optional parameter which help to override the default values of the dialects like skipinitialspace,quoting etc. Can be omitted

1. CSV file - data with default delimiter comma (,)
2. CSV file - data with Space at the beginning
3. CSV file - data with quotes
4. CSV file - data with custom Delimiters

13.6.1.1.1 CSV file with default delimiter comma (,)

The following program read a file called "sample1.csv" with default delimiter comma (,) and print row by row.

```
#importing csv
import csv
#opening the csv file which is in different location with read mode
with open('c:\pyprg\sample1.csv', 'r') as F:
#other way to open the file is f= ('c:\pyprg\sample1.csv', 'r')
    reader = csv.reader(F)
# printing each line of the Data row by row
for row in reader:
    print(row)
F.close()
```

OUTPUT

```
['SNO', 'NAME', 'CITY']
['12101', 'RAM', 'CHENNAI']
['12102', 'LAVANYA', 'TIRUCHY']
['12103', 'LAKSHMAN', 'MADURAI']
```

13.6.1.1.2. CSV files- data with Spaces at the beginning

Consider the following file "sample2.csv" containing the following data when opened through notepad

```
Topic1, Topic2, Topic3,
one, two, three
Example1, Example2, Example3
```


XII Std - CS EM Chapter-13.indd 243 24-01-2020 12:50:36

The following program read the file through Python using "csv.reader()".

```
import csv
csv.register_dialect('myDialect',delimiter = ',',skipinitialspace=True)
F=open('c:\pyprg\sample2.csv','r')
reader = csv.reader(F, dialect='myDialect')
for row in reader:
        print(row)
F.close()
```

OUTPUT

```
['Topic1', 'Topic2', 'Topic3']
['one', 'two', 'three']
['Example1', 'Example2', 'Example3']
```

As you can see in "sample2.csv" there are spaces after the delimiter due to which the output is also displayed with spaces.

These whitespaces can be removed, by registering new dialects using **csv.register_dialect()** class of csv module. **A dialect describes the format of the csv file that is to be read.** In dialects the parameter **"skipinitialspace"** is used for removing whitespaces after the delimiter.

Note

By default "skipinitialspace" has a value false

The following program reads "sample2.csv" file, which contains spaces after the delimiter.

```
import csv
csv.register_dialect('myDialect',delimiter = ',',skipinitialspace=True)
F=open('c:\pyprg\sample2.csv','r')
reader = csv.reader(F, dialect='myDialect')
for row in reader:
        print(row)
F.close()
```

OUTPUT

```
['Topic1', 'Topic2', 'Topic3']
['one', 'two', 'three']
['Example1', 'Example2', 'Example3']
```


XII Std - CS EM Chapter-13.indd 244 24-01-2020 12:50:36

Note

A dialect is a class of csv module which helps to define parameters for reading and writing CSV. It allows you to create, store, and re-use various formatting parameters for your data.

13.6.1.1.3 CSV File-Data With Quotes

You can read the csv file with quotes, by registering new dialects using csv.register_dialect() class of csv module.

Here, we have quotes.csv file with following data.

```
SNO,Quotes
1, "The secret to getting ahead is getting started."
2, "Excellence is a continuous process and not an accident."
3, "Work hard dream big never give up and believe yourself."
4, "Failure is the opportunity to begin again more intelligently."
5, "The successful warrior is the average man, with laser-like focus."
```

The following Program read "quotes.csv" file, where delimiter is comma (,) but the quotes are within quotes (" ").

```
import csv
csv.register_dialect('myDialect',delimiter = ',',quoting=csv.QUOTE_ALL,
skipinitialspace=True)
f=open('c:\pyprg\quotes.csv','r')
reader = csv.reader(f, dialect='myDialect')
for row in reader:
    print(row)
```

OUTPUT

```
['SNO', 'Quotes']
['1', 'The secret to getting ahead is getting started.']
['2', 'Excellence is a continuous process and not an accident.']
['3', 'Work hard dream big never give up and believe yourself.']
['4', 'Failure is the opportunity to begin again more intelligently.']
['5', 'The successful warrior is the average man, with laser-like focus. ']
```

In the above program, register a dialect with name myDialect. Then, we used **csv. QUOTE_ALL to display all the characters** after double quotes.

13.6.1.1.4 CSV files with Custom Delimiters

You can read CSV file having custom delimiter by registering a new dialect with the help of csv.register_dialect().

In the following file called "sample4.csv",each column is separated with | (Pipe symbol)

Roll No | Name | City
12101 | Arun | Chennai
12102 | Meena | Kovai
12103 | Ram | Nellai

The following program read the file "sample4.csv" with user defined delimiter "|"

```
import csv
csv.register_dialect('myDialect', delimiter = '|')
with open('c:\pyprg\sample4.csv', 'r') as f:
    reader = csv.reader(f, dialect='myDialect')
    for row in reader:
        print(row)
f.close()
```

OUTPUT

```
['RollNo', 'Name', 'City']
['12101', 'Arun', 'Chennai']
['12102', 'Meena', 'Kovai']
['12103', 'Ram', 'Nellai']
```

In the above program, a new dialects called myDialect is registered. Use the delimiter=| where a pipe (|) is considered as column separator.

13.6.2 Read a specific column In a File

To get the specific columns like only Item Name and profit for the "sample5.csv" file . Then you have to do the following:

```
import csv
#opening the csv file which is in different location with read mode
f=open("c:\pyprg\ch13sample5.csv",'r')
#reading the File with the help of csv.reader()
readFile=csv.reader(f)
#printing the selected column
for col in readFile :
    print (col[0],col[3])
f.close()
```

sample5.csv File in Excel

XII Std - CS EM Chapter-13.indd 246 24-01-2020 12:50:37

	A	B	C	D
1	item Nam	Cost-Rs	Quantity	Profit
2	Keyboard	480	12	1152
3	Monitor	5200	10	10400
4	Mouse	200	50	2000

sample5.csv File with selected col

OUTPUT

Item Name Profit
Keyboard 1152
Monitor 10400
Mouse 2000

13.6.3 Read A CSV File And Store It In A List

In this topic you are going to read a CSV file and the contents of the file will be stored as a list. The syntax for storing in the List is

```
list = []                    # Start as the empty list
list.append(element)         # Use append() to add elements
```

For example all the row values of "sample.csv" file is stored in a list using the following program

```
import csv
# other way of declaring the filename
inFile= 'c:\pyprg\sample.csv'
F=open(inFile,'r')
reader = csv.reader(F)
# declaring array
arrayValue = []
# displaying the content of the list
for row in reader:
    arrayValue.append(row)
    print(row)
F.close()
```

sample.csv opened in MS-Excel

XII Std - CS EM Chapter-13.indd 247 24-01-2020 12:50:37

A1	^ fx	Topic 1
A	B	C
Topic 1	Topic 2	Topic 3
One	two	three
Example 1	Example 2	Example 3

sample5.csv File with selected col

OUTPUT

['Topic1', 'Topic2', 'Topic3']
[' one', 'two', 'three']
['Example1', 'Example2', 'Example3']

Note

A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements.

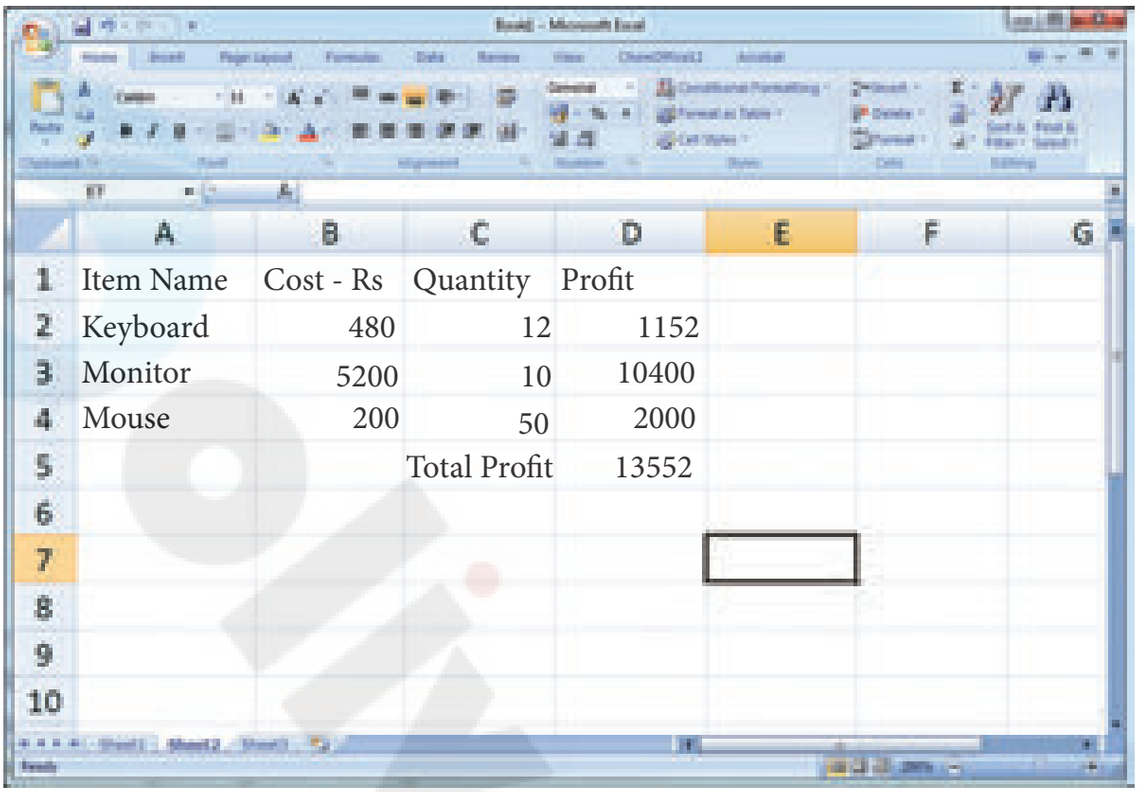
DO YOU KNOW?

List literals are written within square brackets [ ]. Lists work similarly to strings

13.6.4 Read A CSV File And Store A Column Value In A List For Sorting

In this program you are going to read a selected column from the "sample6.csv" file by getting from the user the column number and store the content in a list.

XII Std - CS EM Chapter-13.indd 248 24-01-2020 12:50:37



	A	B	C	D	E	F	G
1	Item Name	Cost - Rs	Quantity	Profit			
2	Keyboard	480	12	1152			
3	Monitor	5200	10	10400			
4	Mouse	200	50	2000			
5			Total Profit	13552			
6							
7							
8							
9							
10							

Fig 13.6.4 CSV file Data for a selected column for sorting

Since the row heading is also get sorted, to avoid that the first row should be skipped. This is can be done by using the command "next()". The list is sorted and displayed.

```
# sort a selected column given by user leaving the header column in
# descending order of value
import csv
# other way of declaring the filename
inFile= 'c:\pyprg\sample6.csv'
# opening the csv file which is in the same location of where the current #
python file
```


XII Std - CS EM Chapter-13.indd 249 24-01-2020 12:50:37

```
F=open(inFile,'r')
#  reading the File with the help of csv.reader()
reader = csv.reader(F)
#  skipping the first row(heading)
next(reader)
#  declaring a list
arrayValue = []
a = int(input ("Enter the column number between 0 to 3:-"))
#  sorting a particular column-cost
for row in reader:
    arrayValue.append(row[a])
arrayValue.sort(reverse=True)
for row in arrayValue:
    print (row)
F.close()
```

OUTPUT

```
Enter the column number between 0 to 3:- 2
50
12
10
```

Read a specific column in a csv file and display its result in Ascending order

DO YOU KNOW

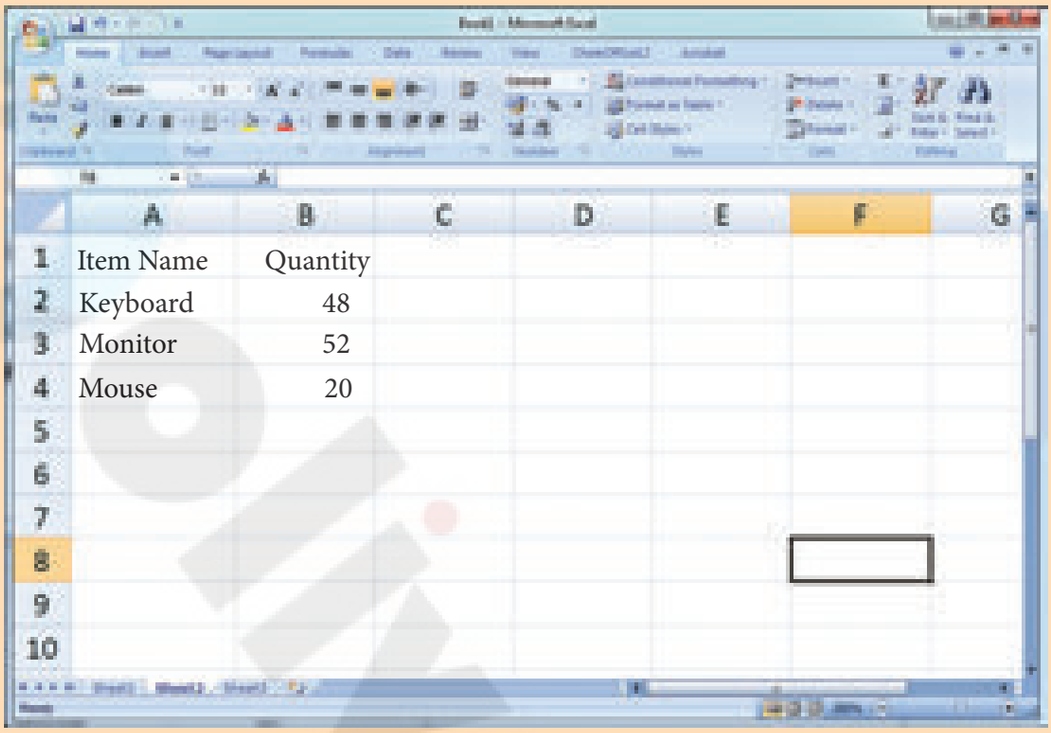
list_name.sort() command arranges a list value in ascending order. list_name.sort(reverse=True) is used to arrange a list in descending order

13.6.5 Sorting A CSV File With A Specified Column

In this program you are going to see the "sample8.csv" file's entire content is transferred to a list. Then the list of rows is sorted and displayed in ascending order of quantity. To sort by more than one column you can use itemgetter with multiple indices: operator .itemgetter (1,2), The content of "sample8.csv" is

XII Std - CS EM Chapter-13.indd 250 24-01-2020 12:50:37

sample8.csv in Excel screen



	A	B
1	Item Name	Quantity
2	Keyboard	48
3	Monitor	52
4	Mouse	20

sample8.csv in Notepad

```
ItemName ,Quantity
Keyboard, 48
Monitor,52
Mouse ,20
```

Fig 13.6.5 CSV file Data into a list for sorting

The following program do the task mentioned above using **operator.itemgetter(col_no)**

```
#Program to sort the entire row by using a specified column.
# declaring multiple header files
import csv ,operator
#One more way to read the file
data = csv.reader(open('c:\pyprg\sample8.csv'))
next(data)                    #(to omit the header)
#using operator module for sorting multiple columns
sortedlist = sorted (data, key=operator.itemgetter(1))    # 1 specifies we want to sort
                                                          # according to  second column

for row in sortedlist:
    print(row)
```

OUTPUT

```
['Mouse ', '20']
['Keyboard ', '48']
['Monitor', '52']
```


XII Std - CS EM Chapter-13.indd 251 24-01-2020 12:50:37

Note

The sorted() method sorts the elements of a given item in a specific order – Ascending or Descending. Sort() method which performs the same way as sorted(). Only difference, sort() method doesn't return any value and changes the original list itself.

Add one more column "cost" in "sample8.csv" and sort it in descending order of cost by using the syntax
sortedlist = sorted(data, key=operator.itemgetter(Col_number),reverse=True)

13.6.6 Reading CSV File Into A Dictionary

To read a CSV file into a dictionary can be done by using **DictReader** class of csv module which works similar to the reader() class but creates an object which maps data to a dictionary. The keys are given by the fieldnames as parameter. **DictReader** works by reading the first line of the CSV and using each comma separated value in this line as a **dictionary key.** The columns in each subsequent row then behave like dictionary values and can be accessed with the appropriate key (i.e. fieldname).

If the first row of your CSV does not contain your column names, you can pass a fieldnames parameter into the DictReader's constructor to assign the dictionary keys manually.

The main difference between the csv.reader() and DictReader() is in simple terms csv.reader and csv.writer work with **list/tuple,** while csv.DictReader and csv.DictWriter work with dictionary. csv.DictReader and csv.DictWriter take additional argument fieldnames that are used as dictionary keys.

For Example Reading "sample8.csv" file into a dictionary

```
import csv
filename = 'c:\pyprg\sample8.csv'
input_file =csv.DictReader(open(filename,'r'))
for row in input_file:
    print(dict(row))                    #dict() to print data
```

OUTPUT

```
{'ItemName ': 'Keyboard ', 'Quantity': '48'}
{'ItemName ': 'Monitor', 'Quantity': '52'}
{'ItemName ': 'Mouse ', 'Quantity': '20'}
```

In the above program, DictReader() is used to read "sample8.csv" file and map into a dictionary. Then, the function dict() is used to print the data in dictionary format without order.

XII Std - CS EM Chapter-13.indd 252 24-01-2020 12:50:37

Remove the dict() function from the above program and use print(row).Check you are getting the following output

```
OrderedDict([('ItemName ', 'Keyboard '), ('Quantity', '48')])
OrderedDict([('ItemName ', 'Monitor'), ('Quantity', '52')])
OrderedDict([('ItemName ', 'Mouse '), ('Quantity', '20')])
```

13.6.7 Reading CSV File With User Defined Delimiter Into A Dictionary

You can also register new dialects and use it in the DictReader() methods. Suppose "sample8.csv" is in the following format

```
ItemName|Quantity
Keyboard|48
Monitor|52
Mouse|20
```

Then "sample8.csv" can be read into a dictionary by registering a new dialect

```
import csv
csv.register_dialect('myDialect',delimiter = '|',skipinitialspace=True)
filename = 'c:\pyprg\ch13\sample8.csv'
with open(filename, 'r') as csvfile:
    reader = csv.DictReader(csvfile, dialect='myDialect')
for row in reader:
    print(dict(row))
csvfile.close()
```

OUTPUT

```
{'ItemName ,Quantity': 'Keyboard ,48'}
{'ItemName ,Quantity': 'Monitor,52'}
{'ItemName ,Quantity': 'Mouse ,20'}
```

Note

DictReader() gives OrderedDict by default in its output. An OrderedDict is a dictionary subclass which saves the order in which its contents are added. To remove the OrderedDict use dict().

13.7 Writing Data Into Different Types in Csv Files

As you know Python provides an easy way to work with CSV file and has csv module to read and write data in the csv file. In the previous topics, You have learned how to read CSV files in Python. In similar way, You can also write a new or edit an existing CSV files in Python.

1. Creating A New Normal CSV File
2. Modifying An Existing File
3. Writing On A CSV File with Quotes
4. Writing On A CSV File with Custom Delimiters
5. Writing On A CSV File with Lineterminator
6. Writing On A CSV File with Quotechars
7. Writing CSV File Into A Dictionary
8. Getting Data At Runtime And Writing In a File

13.7.1 Creating A New Normal CSV File

When you have a set of data that you would like to store inside a CSV file, it's time to do the opposite and use the write function.

The csv.writer() method returns a writer object which converts the user's data into delimited strings on the given file-like object. The writerow() method writes a row of data into the specified file.

The syntax for csv.writer() is

```
csv.writer(fileobject,delimiter,fmtparams)
```

where

fileobject :-	passes the path and the mode of the file
delimiter :-	an optional parameter containing the standard dilects like , | etc can be omitted
fmtparams :	optional parameter which help to override the default values of the dialects like skipinitialspace,quoting etc. can be omitted

You can **create** a normal CSV file using writer() method of csv module having default delimiter comma (,)

Here's an example.

The following Python program converts a List of data to a CSV file called "Pupil.csv" that uses, (comma) as a value separator.

XII Std - CS EM Chapter-13.indd 254 24-01-2020 12:50:37

```
Import csv
csvData = [['Student', 'Age'], ['Dhanush', '17'], ['Kalyani', '18'], ['Ram', '15']]
with open('c:\pyprg\ch13\Pupil.csv', 'w') as CF:
    writer = csv.writer(CF)                 #  CF is the file object
    writer.writerows(csvData)               #  csvData is the List name
CF.close()
```

When you open the "Pupil.csv" file with a text editor, it will show the content as follows.

```
Student, Age
Dhanush, 17
Kalyani, 18
Ram, 15
```

In the above program, csv.writer() method converts all the data in the list "csvData" to strings and create the content as file like object. The writerows () method writes all the data in to the new CSV file "Pupil.csv".

Note

The writerow() method writes one row at a time. If you need to write all the data at once you can use writerows() method.

13.7.2 Modifying An Existing File

Making some changes in the data of the existing file or adding more data is called modification .For example the "student.csv" file contains the following data.

```
Roll No, Name, City
1, Harshini, Chennai
2, Adhith, Mumbai
3, Dhuruv, Bangalore
4, Krishna, Tiruchy
5, Venkat, Madurai
```

The following program modify the "student.csv" file by modifying the value of an existing row in student.csv

XII Std - CS EM Chapter-13.indd 255 24-01-2020 12:50:37

```
import csv
row = ['3', 'Meena','Bangalore']
with open('student.csv', 'r') as readFile:
    reader = csv.reader(readFile)
    lines = list(reader)        #  list()- to store each row of data as a list
    lines[3] = row
with open('student.csv', 'w') as writeFile:
#  returns the writer object which converts the user data with delimiter
    writer = csv.writer(writeFile)
#writerows()method writes multiple rows to a csv file
    writer.writerows(lines)
readFile.close()
writeFile.close()
```

When we open the student.csv file with text editor, then it will show:

Roll No, Name, City

1, Harshini, Chennai

2, Adhith, Mumbai

3, Meena, Bangalore

4, Krishna, Tiruchy

5, Venkat, Madurai

In the above program,the third row of "student.csv" is modified and saved. First the "student.csv" file is read by using csv.reader() function. Then, the list() stores each row of the file. The statement "lines[3] = row", changed the third row of the file with the new content in "row". The file object writer using writerows (lines) writes the values of the list to "student.csv" file.

13.7.2.1 ADDING NEW ROW

Sometimes, you may need to add new rows in the existing CSVfile. Adding a new row at the end of the file is called appending a row.

The following program add a new row to the existing "student.csv" file.

XII Std - CS EM Chapter-13.indd 256 24-01-2020 12:50:37

```
import csv
row = ['6', 'Sajini ', 'Madurai']
with open('student.csv', 'a') as CF:            # append mode to add data at the end
    writer = csv.writer(CF)
    writer.writerow(row)    # writerow() method write a single row of data in file
CF.close()
```

When "student.csv" file is opened with a text editor, it displays as follows

Roll No, Name, City

1, Harshini, Chennai

2, Adhith, Mumbai

3, Meena, Bangalore

4, Krishna, Tiruchy

5, Venkat, Madurai

6, Sajini, Madurai

In the above program, a new row is appended into "student.csv". For this, purpose only the CSV file is opened in 'a' append mode. Append mode write the value of row after the last line of the "student.csv file."

DO YOU KNOW?

The 'w' write mode creates a new file. If the file is already existing 'w' mode over writs it. Where as 'a' append mode add the data at the end of the file if the file already exists otherwise creates a new one

Note

writerow() takes 1-dimensional data (one row), and writerows takes 2-dimensional data (multiple rows) to write in a file.

13.7.3 CSV Files With Quotes

You can write the csv file with quotes, by registering new dialects using **csv.register_dialect()** class of csv module. The following program explains this.

XII Std - CS EM Chapter-13.indd 257 24-01-2020 12:50:37

```
import csv
info = [['SNO', 'Person', 'DOB'],
['1', 'Madhu', '18/12/2001'],
['2', 'Sowmya','19/2/1998'],
['3', 'Sangeetha','20/3/1999'],
['4', 'Eshwar', '21/4/2000'],
['5', 'Anand', '22/5/2001']]
csv.register_dialect('myDialect',quoting=csv.QUOTE_ALL)
with open('c:\pyprg\ch13\person.csv', 'w') as f:
    writer = csv.writer(f, dialect='myDialect')
    for row in info:
        writer.writerow(row)
f.close()
```

When you open "person.csv" file, we get following output :

```
"SNO","Person","DOB"
"1","Madhu","18/12/2001"
"2","Sowmya","19/2/1998"
"3","Sangeetha","20/3/1999"
"4","Eshwar","21/4/2000"
"5","Anand","22/5/2001"
```

In above program, a dialect named myDialect is registered(declared). Quoting=csv.QUOTE_ALL allows to write the double quote on all the values.

13.7.4 CSV Files With Custom Delimiters

A delimiter is a string used to separate fields. The default value is comma(,). You can have custom delimiter in CSV files by registering a new dialect with the help of **csv.register_dialect()**.This example Program is written using the custom delimiter pipe(|)

```
import csv
info = [['SNO', 'Person', 'DOB'],
['1', 'Madhu', '18/12/2001'],
['2', 'Sowmya','19/2/1998'],
['3', 'Sangeetha','20/3/1999'],
['4', 'Eshwar', '21/4/2000'],
['5', 'Anand', '22/5/2001']]
csv.register_dialect('myDialect',delimiter = '|')
with open('c:\pyprg\ch13\dob.csv', 'w') as f:
    writer = csv.writer(f, dialect='myDialect')
    for row in info:
        writer.writerow(row)
f.close()
```


XII Std - CS EM Chapter-13.indd 258 24-01-2020 12:50:37

When we open "dob.csv" file, we get the following output:

```
SNO|Person|DOB
1|Madhu|18/12/2001
2|Sowmya|19/2/1998
3|Sangeetha|20/3/1999
4|Eshwar|21/4/2000
5|Anand|22/5/2001
```

In the above program, a dialect with delimiter as pipe(|)is registered. Then the list "info" is written into the CSV file "dob.csv".

Note

The dialect parameter skipinitialspace when it is True, whitespace immediately following the delimiter is ignored. The default is False.

13.7.5 CSV File With A Line Terminator

A Line Terminator is a string used to terminate lines produced by writer. The default value is \r or \n. We can write csv file with a line terminator in Python by registering new dialects using csv.register_dialect() class of csv module. For Example

```
import csv
Data = [['Fruit', 'Quantity'], ['Apple', '5'], ['Banana', '7'], ['Mango', '8']]
csv.register_dialect('myDialect', delimiter = '|', lineterminator = '\n')
with open('c:\pyprg\ch13\line.csv', 'w') as f:
    writer = csv.writer(f, dialect='myDialect')
    writer.writerows(Data)
f.close()
```

When we open the line.csv file, we get following output with spacing between lines:

```
Fruit|Quantity
Apple|5
Banana|7
Mango|8
```

In the above code, the new dialect "myDialect uses the delimiter='|' where a | (pipe) is considered as column separator. The line terminator='\r\n\r\n' separates each row and display the data after every two lines.

Note

Python's CSV module only accepts \r\n, \n or \r as line terminator

13.7.6 CSV File with quote characters

You can write the CSV file with custom quote characters, by registering new dialects using csv.register_dialect() class of csv module.

```
import csv
csvData = [['SNO','Items'], ['1','Pen'], ['2','Book'], ['3','Pencil']]
csv.register_dialect('myDialect',delimiter = '|',quotechar = '"',
    quoting=csv.QUOTE_ALL)
with open('c:\pyprg\ch13\quote.csv', 'w') as csvFile:
    writer = csv.writer(csvFile, dialect='myDialect')
    writer.writerows(csvData)
print("writing completed")
csvFile.close()
```

When you open the "quote.csv" file in notepad, we get following output:

```
"SNO|"Items
1|"Pen"
2|"Book"
3|"Pencil"
```

In the above program, myDialect uses pipe (|) as delimiter and quotechar as doublequote '"' to write inside the file.

13.7.7 Writing CSV File Into A Dictionary

Using DictWriter() class of csv module, we can write a csv file into a dictionary. It creates an object which maps data into a dictionary. The keys are given by the fieldnames parameter. The following program helps to write the dictionary in to file.

```
import csv
data = [{'MOUNTAIN' : 'Everest', 'HEIGHT': '8848'},
    {'MOUNTAIN' : 'Anamudi ', 'HEIGHT': '2695'},
    {'MOUNTAIN' : 'Kanchenjunga', 'HEIGHT': '8586'}]
with open('c:\pyprg\ch13\peak.csv', 'w') as CF:
fields = ['MOUNTAIN', 'HEIGHT']
    w = csv.DictWriter(CF, fieldnames=fields)
    w.writeheader()
    w.writerows(data)
    print("writing completed")
CF.close()
```


XII Std - CS EM Chapter-13.indd 260 24-01-2020 12:50:37

When you open the "peak.csv" file in notepad, you get the following output:

```
MOUNTAIN,HEIGHT
Everest,8848
Anamudi,2695
Kanchenjunga,8586
```

In the above program, use fieldnames as headings of each column in csv file. Then, use a DictWriter() to write dictionary data into "peak.csv" file.

13.7.7.1 Writing Dictionary Into CSV File With Custom Dialects

```
import csv
csv.register_dialect('myDialect', delimiter = '|', quoting=csv.QUOTE_ALL)
with open('c:\pyprg\ch13\grade.csv', 'w') as csvfile:
    fieldnames = ['Name', 'Grade']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames, dialect="myDialect")
    writer.writeheader()
    writer.writerows([{'Grade': 'B', 'Name': 'Anu'},
                {'Grade': 'A', 'Name': 'Beena'},
                {'Grade': 'C', 'Name': 'Tarun'}])
print("writing completed")
```

When we open grade.csv file, it will contain following output:

```
"Name"|"Grade"
"Anu"|"B"
"Beena"|"A"
"Tarun"|"C"
```

In the above program, a custom dialect called myDialect with pipe (|) as delimiter uses the fieldnames as headings of each column to write in a csv file. Finally, we use a DictWriter() to write dictionary data into "grade.csv" file.

13.7.8 Getting Data At Runtime And Writing It In a CSV File

You can even accept the data through keyboard and write in to a CSV file. For example the following program accept data from the user through key board and stores it in the file called "dynamicfile.csv". It also displays the content of the file.

XII Std - CS EM Chapter-13.indd 261 24-01-2020 12:50:37

```
import csv
with open('c:\pyprg\ch13\dynamicfile.csv', 'w') as f:
    w = csv.writer(f)
    ans='y'
    while (ans=='y'):
        name = input("Name?: ")
        date = input("Date of birth: ")
        place = input("Place: ")
        w.writerow([name, date, place])
        ans=input("Do you want to enter more y/n?: ")
    F=open('c:\pyprg\ch13\dynamicfile.csv','r')
    reader = csv.reader(F)
    for row in reader:
        print(row)
    F.close()
```

OUTPUT

```
Name?: Nivethitha
Date of birth: 12/12/2001
Place: Chennai
Do you want to enter more y/n?: y
Name?: Leena
Date of birth: 15/10/2001
Place: Nagercoil
Do you want to enter more y/n?: y
Name?: Padma
Date of birth: 18/08/2001
Place: Kumbakonam
Do you want to enter more y/n?: n
['Nivethitha', '12/12/2001', 'Chennai']
[]
['Leena', '15/10/2001', 'Nagercoil']
[]
['Padma', '18/08/2001', 'Kumbakonam']
```

MS Excel output

H8 ^ fx

	A	B	C
1	Nivethitha	12/12/2001	Chennai
2			
3	Leena	15/10/2001	Nagercoil
4			
5	Padma	18/08/2001	Kumbakonam
6			

XII Std - CS EM Chapter-13.indd 262 24-01-2020 12:50:37

Points to remember:

- A CSV file is a human readable text file where each line has a number of fields, separated by commas or some other delimiter
- Excel is a binary file whereas CSV format is a plain text format
- The two ways to read a CSV file are using csv.reader() function and using DictReader class.
- The default mode of csv file in reading and writing is text mode
- Binary mode can be be used when dealing with non-text files like image or exe files.
- Python has a garbage collector to clean up unreferenced objects
- close() method will free up the resources that were tied with the file
- By default CSV files should open automatically in Excel
- The CSV library contains objects and other code to read, write, and process data from and to CSV files.
- "skipinitialspace" is used for removing whitespaces after the delimiter
- To sort by more than one column operator.itemgetter() can be used
- DictReader() class of csv module creates an object which maps data to a dictionary
- CSV file having custom delimiter is read with the help of csv.register_dialect().
- To sort by more than one column itemgetter() with multiple indices is used.
- csv.reader and csv.writer work with list/tuple, while csv.DictReader and csv.DictWriter work with dictionary .
- csv.DictReader and csv.DictWriter take additional argument fieldnames that are used as dictionary keys.
- The function dict() is used to print the data in dictionary format without order.
- The csv.writer() method returns a writer object which converts the user's data into delimited strings.
- The writerow() method writes one row at a time. Writerows() method is used to write all the data at once
- Adding a new row at the end of the file is called appending a row.

XII Std - CS EM Chapter-13.indd 263 24-01-2020 12:50:37

Hands on Experience

1. Write a Python program to read the following Namelist.csv file and sort the data in alphabetically order of names in a list and display the output

	A	B	C
1	SNO	NAME	OCCUPATION
2	1	NIVETHITHA	ENGINEER
3	2	ADHITH	DOCTOR
4	3	LAVANYA	SINGER
5	4	VIDHYA	TEACHER
6	5	BINDHU	LECTURER

2. Write a Python program to accept the name and five subjects mark of 5 students .Find the total and store all the details of the students in a CSV file

Evaluation

Part - I

MJGYIV

I Choose the best answer

1. A CSV file is also known as a ….

   (A) Flat File (B) 3D File

   (C) String File (D) Random File

2. The expansion of CRLF is

   (A) Control Return and Line Feed

   (B) Carriage Return and Form Feed

   (C) Control Router and Line Feed

   (D) Carriage Return and Line Feed

3. Which of the following module is provided by Python to do several operations on the CSV files?

   (A) py (B) xls

   (C) csv (D) os

XII Std - CS EM Chapter-13.indd 264 24-01-2020 12:50:37

4. Which of the following mode is used when dealing with non-text files like image or exe files?

(A) Text mode (B) Binary mode

(C) xls mode (D) csv mode

5. The command used to skip a row in a CSV file is

(A) next() (B) skip()

(C) omit() (D) bounce()

6. Which of the following is a string used to terminate lines produced by writer()method of csv module?

(A) Line Terminator (B) Enter key

(C) Form feed (D) Data Terminator

7. What is the output of the following program? import csv

```
d=csv.reader(open('c:\PYPRG\ch13\city.csv'))
next(d)
for row in d:
    print(row)
```

if the file called "city.csv" contain the following details

```
chennai,mylapore
mumbai,andheri
```

A) chennai,mylapore (B) mumbai,andheri

(C) chennai (D) chennai,mylapore

mumba mumbai,andheri

8. Which of the following creates an object which maps data to a dictionary?

(A) listreader() (B) reader()

(C) tuplereader() (D) DictReader ()

9. Making some changes in the data of the existing file or adding more data is called

(A)Editing (B) Appending

(C)Modification (D) Alteration

10. What will be written inside the file test.csv using the following program

```
import csv
D = [['Exam'],['Quarterly'],['Halfyearly']]
```


XII Std - CS EM Chapter-13.indd 265 24-01-2020 12:50:38

```
csv.register_dialect('M',lineterminator = '\n')
with open('c:\pyprg\ch13\line2.csv', 'w') as f:
    wr = csv.writer(f,dialect='M')
    wr.writerows(D)
f.close()
```

(A) Exam Quarterly Halfyearly

(B) Exam Quarterly Halfyearly

(C) E
Q
H

(D) Exam,
Quarterly,
Halfyearly

Part - II

1. What is CSV File?
2. Mention the two ways to read a CSV file using Python.
3. Mention the default modes of the File.
4. What is use of next() function?
5. How will you sort more than one column from a csv file?Give an example statement.

Part - III

1. Write a note on open() function of python. What is the difference between the two methods?
2. Write a Python program to modify an existing file.
3. Write a Python program to read a CSV file with default delimiter comma (,).
4. What is the difference between the write mode and append mode.
5. What is the difference between reader() and DictReader() function?

Part - IV

1. Differentiate Excel file and CSV file.
2. Tabulate the different mode with its meaning.
3. Write the different methods to read a File in Python.
4. Write a Python program to write a CSV File with custom quotes.
5. Write the rules to be followed to format the data in a CSV file.

REFERENCES

1. ***Python for Data Analysis, Data Wrangling with Pandas, NumPy, and IPython By William McKinney***
2. ***CSV File Reading and Writing - Python 3.7.0 documentation***
3. ***https://docs.python.org***

XII Std - CS EM Chapter-13.indd 266 24-01-2020 12:50:38

Unit V

CHAPTER 14

IMPORTING C++ PROGRAMS IN PYTHON

NE9SJV

Learning Objectives

After the completion of this chapter, the student will be able to

- Understand what is wrapping
- Able to import C++ functions and classes in to Python programs
- Create environment to work with both languages
- Execute and debug Python programs

14.1 Introduction

Python and C++ are general-purpose programming language. However, Python is quite different from C++.

S.NO	PYTHON	C++
1	Python is typically an "interpreted" language	C++ is typically a "compiled" language
2	Python is a dynamic-typed language	C++ is compiled statically typed language
3	Data type is not required while declaring variable	Data type is required while declaring variable
4	It can act both as scripting and general purpose language	It is a general purpose language

Yet these two languages complement one another perfectly. Python is mostly used as a scripting or "glue", language. That is, the top level program mostly calls routines written in C or C++. This is useful when the logic can be written in terms of existing code (For example a program written in C++) but can be called and manipulated through Python program.

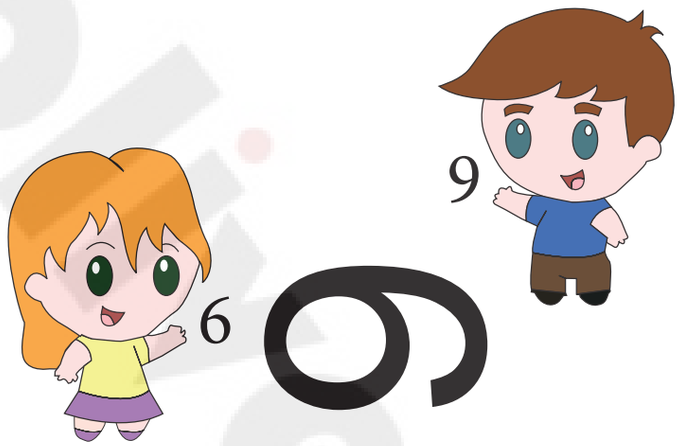
XII Std - CS EM Chapter-14.indd 267 24-01-2020 11:33:48

14.2 Scripting Language

A scripting language is a programming language designed for integrating and communicating with other programming languages. Some of the most widely used scripting languages are JavaScript, VBScript, PHP, Perl, Python, Ruby, ASP and Tcl. Since a scripting language is normally used in conjunction with another programming language, they are often found alongside HTML, Java or C++.

14.2.1 Difference between Scripting and Programming Languages

Scripting Language and Programming Language looks like the following picture.



Basically, all scripting languages are programming languages. The theoretical difference between the two is that scripting languages do not require the compilation step and are rather interpreted. For example, normally, a C++ program needs to be compiled before running whereas, a scripting language like JavaScript or Python need not be compiled. A scripting language requires an interpreter while a programming language requires a compiler. A given language can be called as a scripting or programming language depending on the environment they are put to use.

14.3 Applications of Scripting Languages

1. To automate certain tasks in a program
2. Extracting information from a data set
3. Less code intensive as compared to traditional programming language
4. can bring new functions to applications and glue complex systems together

Python is actually an interpreted, high-level, general-purpose programming language that can be used on any modern computer operating system. It can be used for processing text, numbers, images, scientific data and just about anything else you might save on a computer. Now a days, large applications are written almost exclusively in Python.

XII Std - CS EM Chapter-14.indd 268 24-01-2020 11:33:48

14.4 Features of Python over C++

- Python uses Automatic Garbage Collection whereas C++ does not.
- C++ is a statically typed language, while Python is a dynamically typed language.
- Python runs through an interpreter, while C++ is pre-compiled.
- Python code tends to be 5 to 10 times shorter than that written in C++.
- In Python, there is no need to declare types explicitly where as it should be done in C++
- In Python, a function may accept an argument of any type, and return multiple values without any kind of declaration beforehand. Whereas in C++ return statement can return only one value.

Note

Python deletes unwanted objects (built-in types or class instances) automatically to free the memory space. The process by which Python periodically frees and reclaims blocks of memory that no longer are in use is called Garbage Collection.

14.5 Importing C++ Files in Pythona

Importing C++ program in a Python program is called wrapping up of C++ in Python. Wrapping or creating Python interfaces for C++ programs are done in many ways. The commonly used interfaces are

- Python-C-API (API-**Application Programming Interface** for interfacing with C programs)
- Ctypes (for interfacing with c programs)
- SWIG (Simplified Wrapper Interface Generator- Both C and C++)
- Cython (Cython is both a Python-like language for writing C-extensions)
- Boost. Python (a framework for interfacing Python and C++)
- MinGW (*Minimalist GNU for Windows*)

NTSML7

14.5.1 MinGW Interface

MinGW refers to a set of runtime header files, used in compiling and linking the code of C, C++ and FORTRAN to be run on Windows Operating System.

MinGw-W64 (version of MinGW) is the best compiler for C++ on Windows. To compile and execute the C++ program, you need 'g++' for Windows. MinGW allows to compile and execute C++ program dynamically through Python program using g++.

Python program that contains the C++ coding can be executed through either by using command prompt or by using run terminal.

> g++ is a program that calls GCC (GNU C Compiler) and automatically links the required C++ library files to the object code.

Refer installation of MinGW in Annexure -2

14.5.2 Executing C++ Program through Python

1. Double click on the command prompt or the run terminal.

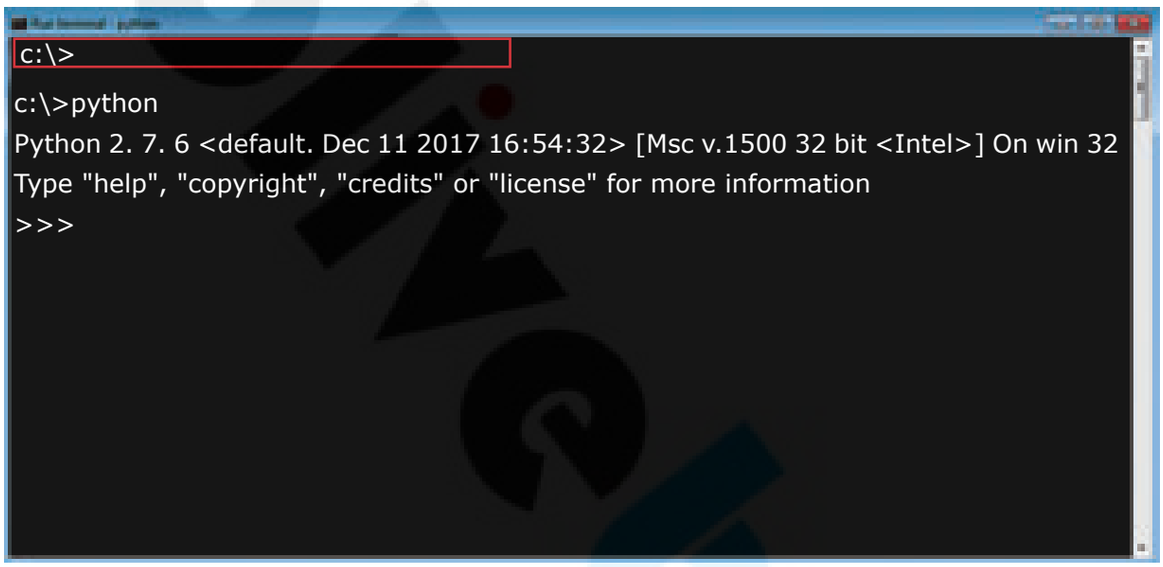


Figure 14.1

In this Example the prompt shows the "C:\>". See that highlighted area in the above window. To change a directory 'cd' command is used. For example to goto the directory pyprg, type the command 'cd pyprg' in the command prompt.

Consider the Example pycpp.py is a Python program which will read the C++program Pali.cpp. The "Pali.cpp" program accepts a number and display whether it is a "Palindrome or Not". For example the entered input number is 232 the output displayed will be "Palindrome". The C++ program Pali is typed in notepad and saved as **pali.cpp.** Same way the Python program **pycpp.py** code is also typed in notepad and saved as pycpp.py.

3. To execute our program double click the run terminal change the path to the Python folder location. The syntax to execute the Python program is

> ***Python <filename.py> -i <C++ filename without cpp extension>***

XII Std - CS EM Chapter-14.indd 270 24-01-2020 11:33:48

Where,

Python	keyword to execute the Python program from command-line
filename.py	Name of the Python program to executed
- i	input mode
C++ filename without cpp extension	name of C++ file to be compiled and executed

For example type Python pycpp.py –i pali in the command prompt and press enter key. If the compilation is successful you will get the desired output. Otherwise the error will be displayed.

Note

In the execution command, the input file doesn't require its extension. For example, it is enough to mention just the name "pali" instead of "pali.cpp".

Now let us will see the execution through our example pycpp.py and pali.cpp. These two programs are stored in the folder c:\pyprg. If the programs are not located in same folder then the complete path must be specified for the files during execution. The output is displayed below

```
C:\Program Files\migw-w64\i686-8.1.0-posix-dwarf-rt_v6-rev0>echo off
Microsoft windows [Version 6.1.7601]
copy right <c> 2009 Microsoft Corporation. All rights reserved.

C:\>cd Pyprg

C:\Pyprg> Python c:\pyprg\pycpp.py-i c:\pyprg\pali
Enter apositive number:232
 The reverse of the number is:232
 The number is a palindrome


C:\Pyprg> Python c:\pyprg\pycpp.py -i c:\pyprg\pali
Enter a positive number:234
 The reverse of the number is:432
 The number is not a palindrome


C:\Pyprg>
```

Fig 14.2

Note

To clear the screen in command window use **cls** command

Now let us will see how to write the Python program for compiling C++ code.

14.6 Python Program to import C++

Python contains many modules. For a problem Python allow programmers to have the flexibility in using different module as per their convenience. The Python program what we have written contains a few new commands which we have not come across in basic Python program. Since our program is an integration of two different languages, we have to import the modules like os, sys and getopt.

14.6.1 MODULE

Modular programming is a software design technique to split your code into separate parts. These parts are called modules. The focus for this separation should have modules with no or just few dependencies upon other modules. In other words: Minimization of dependencies is the goal.

But how do we create modules in Python? Modules refer to a file containing Python statements and definitions. A file containing Python code, for e.g. factorial.py, is called a module and its module name would be factorial. We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code. We can define our most used functions in a module and import it, instead of copying their definitions into different programs.

Example:

```
def fact(n):
        f=1
        if n == 0:
        return 0
elif n == 1:
        return 1
        else:
                for i in range(1, n+1):
                        f= f*i
                print (f)
```

Output:

```
>>>fact (5)
120
```

The above example is named as factorial.py

XII Std - CS EM Chapter-14.indd 272 24-01-2020 11:33:48

14.6.2 How to import modules in Python?

We can import the definitions inside a module to another module. We use the **import** keyword to do this. To import our previously defined module **factorial** we type the following in the Python prompt.

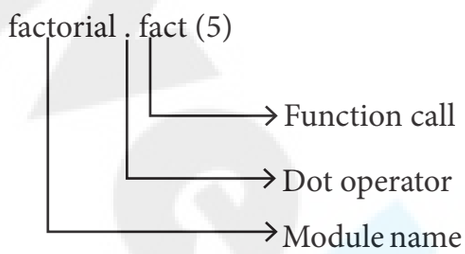
>>> import factorial

Using the module name we can access the functions defined inside the module. The dot (.) operator is used to access the functions. The syntax for accessing the functions from the module is

<module name> . <function name>

For example:

```
>>> factorial.fact(5)
120
```



Python has number of standard (built in) modules. Standard modules can be imported the same way as we import our user-defined modules. We are now going to see the Standard modules which are required for our program to run C++ code.

14.6.2.1 Python's sys module

This module provides access to builtin variables used by the interpreter. One among the variable in **sys** module is **argv**

sys.argv

sys.argv is the list of command-line arguments passed to the Python program. **argv contains** all the items that come via the command-line input, it's basically a list holding the command-line arguments of the program.

To use **sys.argv**, **import sys** should be used. The first argument, sys.argv[0] contains the name of the python program (example pali.py) and sys.argv[1] is the next argument passed to the program (here it is the C++ file). Which will be the argument passed through main (). For example

XII Std - CS EM Chapter-14.indd 273 24-01-2020 11:33:48

main(sys.argv[1])	The input file (C++ file) is send along with its path as a list(array) using **argv[1]**. **argv[0]** contains the Python program which is need not to be passed because by default __main__ contains source code reference

14.6.2.2 Python's OS Module

The ***OS*** module in Python provides a way of using operating system dependent functionality.

The functions that the ***OS*** module allows you to interface with the Windows operating system where Python is running on.

os.system(): Execute the C++ compiling command (a string contains Unix, C command which also supports C++ command) in the shell (Here it is Command Window). For Example to compile C++ program **g++** ***compiler*** should be invoked. To do so the following command is used.

os.system ('g++ ' + <variable_name1> ' -<mode> ' + <variable_name2>

where each argument contains

os.system :-	function system() defined in **os** module to interact with the operating system
g++ :-	General compiler to compile C++ program under Windows Operating system.
variable_name1:-	Name of the C++ file along with its path and without extension .cpp in string format
mode :-	To specify input or output mode. Here it is o prefixed with hyphen.
variable_name2 :-	Name of the executable file without extension .exe in string format

For example the command to compile and execute C++ program is given below

os.system('g++ ' + cpp_file + ' -o ' + exe_file)	g++ compiler compiles the file cpp_file and –o (output) send to exe_file

Note

'+' in os.system() indicates that all strings are concatenated as a single string Therfore give a space after each word for the above argument. For example **'g++ ' + cpp_file + ' -o ' + exe_file**

XII Std - CS EM Chapter-14.indd 274 24-01-2020 11:33:48

14.6.2.3.3 Python getopt module

The getopt module of Python helps you to parse (split) command-line options and arguments. This module provides getopt() method to enable command-line argument parsing.

getopt.getopt method

This method parses command-line options and parameter list. Following is the syntax for this method –

<opts>,<args>=getopt.getopt(argv, options, [long_options])

Here is the detail of the parameters –

argv – This is the argument list of values to be parsed (splited). In our program the complete command will be passed as a list. For example ***c:\pyprg\pali.py -i c:\pyprg\pali_cpp***

options – This is string of option letters that the Python program recognize as, for input or for output, with options (like 'i' or 'o') that followed by a colon (:). Here colon is used to denote the mode.

long_options – This contains a list of strings. Argument of Long options should be followed by an equal sign ('='). In our program the C++ file name along with its path will be passed as string and 'i' i will be also passed to indicate it as the input file.

getopt() method returns value consisting of two elements. Each of these values are stored separately in two different list (arrays) **opts** and **args** .Opts contains list of splitted strings like mode and path. **args** contains error string, if at all the comment is given with wrong path or mode. **args** will be an empty list if there is no error.

For example The Python code which is going to execute the **C++ file p4** in command line will have the getopt() method like the following one.

opts, args = getopt.getopt (argv, "i:",['ifile='])

where **opts** contains	[('-i', 'c:\\pyprg\\p4')]
-i :-	**option - mode should be followed by : (colon)**
'c:\\pyprg\\p4'	**value - absolute path of C++ file.**

In our examples since the entire command line commands are parsed and no leftover argument, the **second argument args** will be empty []. If args is displayed using print() command it displays the output as [].

```
>>>print(args)
[]
```


XII Std - CS EM Chapter-14.indd 275 24-01-2020 11:33:48

Note

You can check out the full list of Python standard modules and what they are for. These files are in the Lib directory inside the location where Python is installed.

Some more command for wrapping C++ code

```
if __name__=='__main__':
        main(sys.argv[1:])
```

__name__ (A Special variable) in Python

Since there is no main() function in Python, when the command to run a Python program is given to the interpreter, the code that is at level 0 indentation (top must line of the program) is to be executed. However, before doing that, interpreter will define a few special variables. **__name__ is one such special variable which by default stores the name of the file.** If the source file is executed as the main program, the interpreter sets the __name__ variable to have the name of the source file as its value. "__main__" by defaut contains the name of the source file (Python program)

__name__ is a built-in variable which evaluates to the name of the current module. Thus it can be used to check whether the current script is being run on its own.

For example consider the following

```
if __name__ == '__main__':
        main (sys.argv[1:])
```

if the command line Python program itself is going to execute first, then __main__ contains the name of that Python program and the Python special variable __name__ also contain the Python program name. If the condition is true it calls the **main** which is passed with C++ file as argument.

Note

sys.argv[1:] - get everything after the script name(file name).
sys.argv[0] is the script name (python program)
Remember "string slicing" you have studied in chapter 8.

14.7 Python program Executing C++ Program using control statement

Now let us write a Python program to read a C++ coding and execute its result. The steps for executing the C++ program to check a given number is palindrome or not is given below

XII Std - CS EM Chapter-14.indd 276 24-01-2020 11:33:49

Step 1 Type the C++ program to check whether the input number is palindrome or not in notepad and save it as **"pali_cpp.cpp".**

Step 2 Type the Python program and save it as pali.py

Step 3 Click the Run Terminal and open the command window

Step 4 Type the command Python pali.py -i pali_cpp

Example:- 14.7.1 - Write a C++ program to enter any number and check whether the number is palindrome or not using while loop.

```
/*. To check whether the number is palindrome or not using while loop.*/
//Now select File->New in Notepad and type the C++ program
#include <iostream>
using namespace std;
int main()
{
int n, num, digit, rev = 0;
cout<< "Enter a positive number: ";
cin>>num;
n = num;
while(num)
{
    digit = num % 10;
    rev = (rev * 10) + digit;
    num = num / 10;
}
cout<< " The reverse of the number is: " << rev <<endl;
if (n == rev)
    cout<< " The number is a palindrome";
else
    cout<< " The number is not a palindrome";
return 0;
}
// Save this file as pali_cpp.cpp
```


XII Std - CS EM Chapter-14.indd 277 24-01-2020 11:33:49

```
#Now select File→New in Notepad and type the Python program
# Save the File as pali.py . Program that compiles and executes a .cpp file
# Python c:\pyprg\pali.py -i c:\pyprg\pali_cpp
import sys, os, getopt
def main(argv):
    opts, args = getopt.getopt(argv, "i:")
    for o, a in opts:
        if o in "-i":
            run(a)

def run(a):
    inp_file=a+'.cpp'
    exe_file=a+'.exe'
    os.system('g++  ' + inp_file + '  -o  ' + exe_file)
    os.system(exe_file)
if __name__=='__main__':
        main(sys.argv[1:])
```

Output of the above program

Output 1

```
C:\Users\Dell>python c:\pyprg\pali.py -i c:\pyprg\pali_cpp
Enter a positive number: 56765
The reverse of the number is: 56765
The number is a palindrome
```

Output 2

```
C:\Users\Dell>python c:\pyprg\pali.py -i c:\pyprg\pali_cpp
Enter a positive number: 56756
The reverse of the number is: 65765
The number is not a palindrome
```


XII Std - CS EM Chapter-14.indd 278 24-01-2020 11:33:49

Python code	How does it works
import sys, os, getopt	include sys , os and getopt modules to use the required function
def main(argv):	Function main() is defined and 'argv' contains the 'input mode and the c++ program file' in the form of list i.e ['-i', 'c:\ pyprg \pali_cpp']
opts, args = getopt.getopt(argv, "i:")	getopt() splits the command as option and argument. 'opts' contains [('-i', 'c:\pyprg\pali_cpp')]. Since no error 'args' shows []
for o, a in opts:	'o' contains the mode and 'a' contains the path of c++ program i.e print("o = ",o) shows o = -i print("a = ",a) shows a = c:\pyprg\pali_cpp
if o in ("-i"):	Checks o == 'i' if true
run(a)	Calls the function run() passed along with the c++ program
def run(a):	Definition of run() function begins here
inp_file=a+'.cpp'	Variable 'inp_file' contains the joined c++ program name and .cpp print(inp_file) shows c:\pyprg\pali_cpp.cpp
exe_file=a+'.exe'	Variable 'exe_file' contains the joined c++ program name and .exe print(exe_file) shows c:\pyprg\pali_cpp.exe
os.system('g++ ' + inp_file + ' -o ' + exe_file)	g++ compiler compiles the c++ program in inp_file and store the executable file in exe_file
os.system(exe_file)	Executes the exe file
if __name__=='__main__':	__name__ stores name of the python program __ main__ also stores the name of the python program.
main(sys.argv[1:])	If 'name' and 'main' are equal then main() is called and passed with the command line argument omitting the python program name argv[1:] contains –i c:\pyprg\pali_cpp

XII Std - CS EM Chapter-14.indd 279 24-01-2020 11:33:49

The Python script(program) is mainly used to read the C++ file along with the type of mode like 'i'/'o'. 'getopt()' Parses(splits) each value of the command line and passes the options(values) as list to 'opt' and since no error 'args' generates empty list[]. Using 'for loop' the tuple in the list is unpacked - 'o' stores the mode and 'a' stores the name along with the path of the c++ file.

The variable 'inp_file' store the c++ file along with its extension and 'exe_file' stores the executable file with .exe extension. '+' usd in this program helps to concatenate the file name with the extensions. 'os.system()' along with 'g++' compiles the inp_file. Mode 'o' sends the executable file to 'exe_file'.

'__name__' variable directs the program to start from the beginning of the Python script(zero'th line) The "main()" definition does the Parsing and calling the run(). The "run()" invoke the "g++" compiler and creates the exe file. The system() of "os" module executes the .exe file and the desired output will be displayed on the output screen. The file extensions are added by the Python script so it is even possible to execute C programs.

14.8 How Python is handling the errors in C++

Python not only execute the successful C++ program, it also helps to display even errors if any in C++ statement during compilation. For example in the following C++ program an error is there. Let us see what happens when you compile through Python.

Example 14.8.1

```
// C++ program to print the message Hello
//Now select File→New in Notepad and type the C++ program
#include<iostream>
using namespace std;
int main()
{
      std::cout<<"hello"
      return 0;
}
// Save this file as hello.cpp

# Now select File→New in Notepad and type the Python program as main.py
# Program that compiles and executes a .cpp file
# Python main.py -i  hello

import sys, os, getopt
def main(argv):
   opts, args = getopt.getopt(argv, "i:")
   for o, a in opts:
      if o in "-i":
         run(a)
```


XII Std - CS EM Chapter-14.indd 280 24-01-2020 11:33:49

```
def run(a):
    inp_file=a+'.cpp'
    exe_file=a+'.exe'
    os.system('g++  ' + inp_file + '  -o  ' + exe_file)
    os.system(exe_file)
if __name__=='__main__':
        main(sys.argv[1:])
```

Output of the above program

```
C:\Users\Dell>python c:\pyprg\main.py -i c:\pyprg\hello
c:\pyprg\hello.cpp: In function 'int main()':
c:\pyprg\hello.cpp:6:21: error: expected ';' before 'return'
  std::cout<<"hello"
                    ^
                    ;
  return 0;
  ~~~~~~
'c:\pyprg\hello.exe' is not recognized as an internal or external command,
operable program or batch file.
```

Note

In the above program Python helps to display the error in C++. The error is displayed along with its line number. The line number starts from python script.

14.9 Python program Executing C++ Program Containing Arrays

In our previous program to check whether a number is a palindrome number or not you have used only control structure statements. Now you are going to execute a C++ program containing array.

Example :- 14.9.1

Write a C++ program to print Transpose of a matrix(2 D array).*/

```
//Now select File->New in Notepad and type the  C++  program
#include <iostream>
using namespace std;
int main()
{
int a[3][3], i, j;
for(i=0; i<3; i++)
```


XII Std - CS EM Chapter-14.indd 281 24-01-2020 11:33:49

```
{
for(j=0; j<3; j++)
      {  cout<<"enter the value for array["<<i+1<<"]"<<"["<<j+1<<"] :";
         cin>>a[i][j];
      }
}
system("cls");
cout<<"\n\nOriginal Array\n";
for(i=0; i<3; i++)    {
      for(j=0; j<3; j++)
         cout<<a[i][j]<<' ';
      cout<<endl;   }
cout<<"\n\n The Transpose of Matrix\n";
for (int i = 0; i < 3; i++)
{
      for (int j = 0; j < 3; j++)
         cout<<a[j][i]<<' ';
cout<<endl ;
}
      return 0;
}
```

```
// Save this file as trans_cpp.cpp
//Now select File→New in Notepad and type the Python program
# Save the File as transpose.py.Program that compiles and executes a
.cpp file
# Python tanspose.py -i trans_cpp
import sys, os, getopt
def main(argv):
   opts, args = getopt.getopt(argv, "i:")
   for o, a in opts:
      if o in "-i":
         run(a)
def run(a):
   inp_file=a+'.cpp'
   exe_file=a+'.exe'
if __name__=='__main__':
      main(sys.argv[1:])
```


XII Std - CS EM Chapter-14.indd 282 24-01-2020 11:33:49

Output of the above program

```
Original Array
    1 2 3
    4 5 6
    7 8 9
The Transpose of Matrix
    1 4 7
    2 5 8
    3 6 9
```

You would have noticed the Python program used to execute the transpose of a matrix is also the same as what you have used in palindrome program. From this what you have understood?

14.10 Python program Executing C++ Program Containing Functions

Now you are going to test the Python script to run a C++ program having functions and function calls.

Example 14.10.1 - Write a C++ program using a user defined function to function cube of a number

```
/*Write a C++ program using a user defined function to function  cube of a
number.*/
//Now select File→New in Notepad and type the C++ program
#include <iostream>
using namespace std;
// Function declaration
int cube(int num);
int main()
{
int num;
int c;
cout<<"Enter any number: "<<endl;
cin>>num;
c = cube(num);
cout<<"Cube of " <<num<< " is "<<c;
return 0;
}
//Function to find cube of any number
int cube(int num)
{
    return (num * num * num);
}
```

```
// Save this file as cube_file.cpp
#Now select File→New in Notepad and type the Python program
# Save the File as fun.py
# Program that compiles and executes a .cpp file
 # Python fun.py -i c:\pyprg\cube_file.cpp
import sys, os, getopt
def main(argv):
   opts, args = getopt.getopt(argv, "i:")
   for o, a in opts:
      if o in "-i":
         run(a)
def run(a):
   inp_file=a+'.cpp'
   exe_file=a+'.exe'
   os.system('g++  ' + inp_file + '  -o  ' + exe_file)
   os.system(exe_file)
if __name__=='__main__':
      main(sys.argv[1:])
```

Output of the above program

```
Enter any number:
5
Cube of 5 is 125
```

14.11 Python program to Illustrate the inheritance of a Class

Now you are going to execute a class program of C++. The class program is of multilevel inheritance. This also gives output using Python script

Example 14.11.1 - C++ program to implement Multilevel Inheritance

```
// C++ program to implement Multilevel Inheritance
//Now select File→New in Notepad and type the C++ program
#include <iostream>
using namespace std;
// base class
class Vehicle
{
      public:
      Vehicle()
      {
         cout<< "This is a Vehicle" <<endl;
      }
};
```


XII Std - CS EM Chapter-14.indd 284 24-01-2020 11:33:49

```
class threeWheeler: public Vehicle
{  public:
      threeWheeler()
      {
         cout<<"Objects with 3 wheels are vehicles"<<endl;
      }
};
// sub class derived from two base classes
class Auto: public threeWheeler {
      public:
         Auto()
      {
         cout<<"Auto has 3 Wheels"<<endl;
      }
};
// main function
int main()
{
      //creating object of sub class will invoke the constructor of base classes
      Auto obj;
      return 0;
}
```

```
// Save this file as inheri_cpp.cpp
//Now select File → New in Notepad and type the Python program
# Save the File as classpy.py
# Python classpy.py -i  inheri_cpp  command to execute c++ program
import sys, os, getopt
def main(argv):
   opts, args = getopt.getopt(argv, "i:")
   for o, a in opts:
      if o in "-i":
         run(a)
def run(a):
   inp_file=a+'.cpp'
   exe_file=a+'.exe'
   os.system('g++  ' + inp_file + '  -o  ' + exe_file)
   os.system(exe_file)
if __name__=='__main__':
      main(sys.argv[1:])
```

Output of the above program

This is a Vehicle
Objects with 3 wheels are vehicles
Auto has 3 Wheels

From all these example One can understand the Python script(program) used for integrating C++ is common and very simple. Only the name of the Python script file and the C++ (cpp) file have changed. Remember it is not mandatory to type the C++ coding if it already exits. The various types of C++ programs like normal, function, inheritance program can be executed through python. Using the same Python script you can even compile and execute C program. Since python automates the C++ program file to execute without it's IDE, Python can be called as a Scripting Language.

Points to remember:

- C++ is a compiler based language while Python is an interpreter based language.
- C++is compiled statically whereas Python is interpreted dynamically
- A static typed language like C++ requires the programmer to explicitly tell the computer what "data type" each data value is going to use.
- A dynamic typed language like Python, doesn't require the data type to be given explicitly for the data. Python manipulate the variable based on the type of value.
- A scripting language is a programming language designed for integrating and communicating with other programming languages
- MinGW refers to a set of runtime header files, used in compiling and linking the code of C, C++ and FORTRAN to be run on Windows Operating System
- The dot (.) operator is used to access the functions of a imported module
- sys module provides access to some variables used by the interpreter and to functions that interact with the interpreter
- OS module in Python provides a way of using operating system dependent functionality
- The getopt module of Python helps you to parse (split) command-line options and arguments

XII Std - CS EM Chapter-14.indd 286 24-01-2020 11:33:49

Hands on Experience

1. Write a C++ program to create a class called Student with the following details

   Protected member

   Rno integer

   Public members

   void Readno(int); to accept roll number and assign to Rno

   void Writeno(); To display Rno.

   The class Test is derived Publically from the Student class contains the following details

   Protected member

   Mark1 float

   Mark2 float

   Public members

   void Readmark(float, float); To accept mark1 and mark2

   void Writemark(); To display the marks

   Create a class called Sports with the following detail

   Protected members

   score integer

   Public members

   void Readscore(int); To accept the score

   void Writescore(); To display the score

   The class Result is derived Publically from Test and Sports class contains the following details

   Private member

   Total float

   Public member

   void display() assign the sum of mark1, mark2, score in total.

   invokeWriteno(), Writemark() and Writescore(). Display the total also.

   Save the C++ program in a file called hybrid. Write a python program to execute the hybrid.cpp

2. Write a C++ program to print boundary elements of a matrix and name the file as Border.cpp. Write a python program to execute the Border.cpp

XII Std - CS EM Chapter-14.indd 287 24-01-2020 11:33:49

Evaluation

Part - I

1. Which of the following is not a scripting language?

   (A) JavaScript (B) PHP

   (C) Perl (D) HTML

2. Importing C++ program in a Python program is called

   (A) wrapping (B) Downloading

   (C) Interconnecting (D) Parsing

Q44BNI

3. The expansion of API is

   (A) Application Programming Interpreter

   (B) Application Programming Interface

   (C) Application Performing Interface

   (D) Application Programming Interlink

4. A framework for interfacing Python and C++ is

   (A) Ctypes (B) SWIG

   (C) Cython (D) Boost

5. Which of the following is a software design technique to split your code into separate parts?

   (A) Object oriented Programming

   (B) Modular programming

   (C) Low Level Programming

   (D) Procedure oriented Programming

6. The module which allows you to interface with the Windows operating system is

   (A) OS module (B) sys module

   (c) csv module (d) getopt module

7. getopt() will return an empty array if there is no error in splitting strings to

   (A) argv variable (B) opt variable

   (c)args variable (d) ifile variable

XII Std - CS EM Chapter-14.indd 288 24-01-2020 11:33:49

8. Identify the function call statement in the following snippet.

```
if __name__ =='__main__':
        main(sys.argv[1:])
```

(A) main(sys.argv[1:]) (B) __name__

(C) __main__ (D) argv

9. Which of the following can be used for processing text, numbers, images, and scientific data?

(A) HTML (B) C

(C) C++ (D) PYTHON

10. What does __name__ contains ?

(A) c++ filename (B) main() name

(C) python filename (D) os module name

Part - II

1. What is the theoretical difference between Scripting language and other programming language?
2. Differentiate compiler and interpreter.
3. Write the expansion of (i) SWIG (ii) MinGW
4. What is the use of modules?
5. What is the use of cd command. Give an example.

Part - III

1. Differentiate PYTHON and C++
2. What are the applications of scripting language?
3. What is MinGW? What is its use?
4. Identify the module ,operator, definition name for the following

   welcome.display()
5. What is sys.argv? What does it contain?

Part - IV

1 Write any 5 features of Python.

2. Explain each word of the following command.

   Python <filename.py> -<i> <C++ filename without cpp extension>

XII Std - CS EM Chapter-14.indd 289 24-01-2020 11:33:49

3. What is the purpose of sys,os,getopt module in Python.Explain

4. Write the syntax for getopt() and explain its arguments and return values

5. Write a Python program to execute the following c++ coding

```
#include <iostream>
using namespace std;
int main()
{ cout<<"WELCOME";
return(0);
}
```

The above C++ program is saved in a file welcome.cpp

REFERENCES

1. *Learn Python The Hard Way by Zed Shaw*
2. *Python Programming Advanced by Adam Stuart or Powerful Python by Aaron Maxwell*
3. *https://docs.python.org*

XII Std - CS EM Chapter-14.indd 290 24-01-2020 11:33:49

Unit V

CHAPTER 15

DATA MANIPULATION THROUGH SQL

Learning Objectives

After the completion of this chapter, the student will be able to write Python script to

- Create a table and to add new rows in the database.
- Update and Delete record in a table
- Query the table
- Write the Query in a CSV file

QIM6PU

15.1 Introduction

A database is an organized collection of data. The term "database" can both refer to the data themselves or to the database management system. The Database management system is a software application for the interaction between users and the databases. Users don't have to be human users. They can be other programs and applications as well. We will learn how Python program can interact as a user of an SQL database.

15.2 SQLite

SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer. It is designed to be embedded in applications, instead of using a separate database server program such as MySQLor Oracle. SQLite is fast, rigorously tested, and flexible, making it easier to work. Python has a native library for SQLite. To use SQLite,

Step 1 import sqlite3

Step 2 create a connection using connect () method and pass the name of the database File

Step 3 Set the cursor object cursor = connection. cursor ()

- Connecting to a database in step2 means passing the name of the database to be accessed. If the database already exists the connection will open the same. Otherwise, Python will open a new database file with the specified name.

XII Std - CS EM Chapter-15.indd 291 24-01-2020 12:52:57

- Cursor in step 3: is a control structure used to traverse and fetch the records of the database.
- Cursor has a major role in working with Python. All the commands will be executed using cursor object only.

To create a table in the database, create an object and write the SQL command in it.

Example:- sql_comm = "SQL statement"

For executing the command use the cursor method and pass the required sql command as a parameter. Many number of commands can be stored in the sql_comm and can be executed one after other. Any changes made in the values of the record should be saved by the commend "Commit" before closing the "Table connection".

15.3 Creating a Database using SQLite

The following example 15.3 explains how a connection to be made to a database through Python sqlite3

```
# Python code to demonstrate table creation and  insertions with SQL
# importing module
import sqlite3
# connecting to the database
connection = sqlite3.connect ("Academy.db")
# cursor
cursor = connection.cursor()
```

In the above example a database with the name "Academy" would be created. It's similar to the sql command "CREATE DATABASE Academy;" to SQL server."sqlite3.connect ('Academy.db')" is again used in some program, "connect" command just opens the already created database.

15.3.1 Creating a Table

After having created an empty database, you will most probably add one or more tables to this database. The SQL syntax for creating a table "Student" in the database "Academy" looks like as follows :

CREATE TABLE Student (

RollnoINTEGER, SnameVARCHAR(20), GradeCHAR(1), gender CHAR(1),

Average float(5.2), birth_date DATE, PRIMARY KEY (Rollno));

This is the way, somebody might do it on a SQL command shell. Of course, we want to do this directly from Python. To be capable to send a command to "SQL", or SQLite, we need a

XII Std - CS EM Chapter-15.indd 292 24-01-2020 12:52:57

cursor object. Usually, **a cursor in SQL and databases is a control structure to traverse over the records in a database.** So it's used for the fetching of the results.

Note

Cursor is used for performing all SQL commands.

The cursor object is created by calling the cursor() method of connection. The cursor is used to traverse the records from the result set. You **can define a SQL command with a triple quoted string in Python.** The reason behind the triple quotes is sometime the values in the table might contain single or double quotes.

Example 15.3.1

```
sql_command = """
CREATE TABLE Student (
Rollno INTEGER PRIMARY KEY ,
Sname VARCHAR(20),
Grade CHAR(1),
gender CHAR(1),
Average DECIMAL(5,2),
birth_date DATE);"""
```

In the above example the Emp_no field as "INTEGER PRIMARY KEY" A column which is labeled like this will be automatically auto-incremented in SQLite3. To put it in other words: **If a column of a table is declared to be an INTEGER PRIMARY KEY, then whenever a NULL will be used as an input for this column, the NULL will be automatically converted into an integer which will one larger than the highest value so far used in that column.** If the table is empty, the value **1** will be used.

15.3.2 Adding Records

To populate (add record) the table "INSERT" command is passed to SQLite. “execute” method executes the SQL command to perform some action. The following example 15.3.2 is a complete working example. To run the program you will either have to remove the file Academy. db or uncomment the "DROP TABLE" line in the SQL command:

XII Std - CS EM Chapter-15.indd 293 24-01-2020 12:52:57

Example 15.3.2 -1

```
import sqlite3
connection = sqlite3.connect ("Academy.db")
cursor = connection.cursor()
# if the table already exits then delete it using cursor.execute ("""DROP TABLE Student;""")
# where  Student is name of the table.
sql_command = """
CREATE TABLE Student (
Rollno INTEGER PRIMARY KEY , Sname VARCHAR(20), Grade CHAR(1),
gender CHAR(1), Average DECIMAL (5, 2), birth_date DATE);"""
cursor.execute(sql_command)
sql_command = """INSERT INTO Student (Rollno, Sname, Grade, gender, Average,
    birth_date) VALUES (NULL, "Akshay", "B", "M","87.8", "2001-12-12");"""
cursor.execute(sql_command)
sql_command = """INSERT INTO Student (Rollno, Sname, Grade, gender, Average,
    birth_date) VALUES (NULL, "Aravind", "A", "M","92.50","2000-08-17");"""
cursor.execute(sql_command)
# never forget this, if you want the changes to be saved:
connection.commit()
connection.close()
print("STUDENT TABLE CREATED")
```

OUTPUT

```
STUDENT TABLE CREATED
```

Of course, in most cases, you will not literally insert data into a SQL table. You will rather have a lot of data inside of some Python data type e.g. a dictionary or a list, which has to be used as the input of the insert statement.

The following working example, assumes that you have an already existing database Academy.db and a table Student. We have a list with data of persons which will be used in the INSERT statement:

XII Std - CS EM Chapter-15.indd 294 24-01-2020 12:52:57

Example 15.3.2-2

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
student_data = [("BASKAR", "C", "M","75.2","1998-05-17"),
    ("SAJINI", "A", "F","95.6","2002-11-01"),
    ("VARUN", "B", "M","80.6","2001-03-14"),
    ("PRIYA", "A", "F","98.6","2002-01-01"),
    ("TARUN", "D", "M","62.3","1999-02-01") ]
for p in student_data:
    format_str = """INSERT INTO Student (Rollno, Sname, Grade, gender,Average,
    birth_date) VALUES (NULL,"{name}", "{gr}", "{gender}","{avg}","{birthdate}");"""
    sql_command = format_str.format(name=p[0], gr=p[1], gender=p[2],avg=p[3],
    birthdate = p[4])
cursor.execute(sql_command)
connection.commit()
connection.close()
print("RECORDS ADDED TO STUDENT TABLE ")
```

OUTPUT

```
RECORDS ADDED TO STUDENT TABLE
```

In the above program {gr} is a place holder (variable) to get the value. format_str.format() is a function used to format the value to the required datatype.

15.4 SQL Query Using Python

The time has come now to finally query our "Student" table. Fetching the data from record is as simple as inserting them. The execute method uses the SQL command to get all the data from the table.

15.4.1 SELECT Query

"Select" is the most commonly used statement in SQL. The SELECT Statement in SQL is used to retrieve or fetch data from a table in a database. The syntax for using this statement is **"Select * from table_name"** and all the table data can be fetched in an object in the form of list of lists.

If you run this program, saved as "sql_Academy_query.py", you would get the following result, depending on the actual data:

It should be noted that the database file that will be created will be in the same folder as that of the python file. If we wish to change the path of the file, change the path while opening the file.

Example 15.4.1-1

```
import sqlite3
#save the file as "sql_Academy_query.py"
# cursor object
crsr = connection.cursor()
# execute the command to fetch all the data from the table Student
crsr.execute("SELECT * FROM Student")
# store all the fetched data in the ans variable
ans= crsr.fetchall()
# loop to print all the data
for i in ans:
        print(i)
```

15.4.1.1 Displaying all records using fetchall()

The fetchall() method is used to fetch all rows from the database table

Example 15.4.1.1-1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("fetchall:")
result = cursor.fetchall()
for r in result:
    print(r)
```

OUTPUT

```
fetchall:
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
(6, 'PRIYA', 'A', 'F', 98.6, '2002-01-01')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```


XII Std - CS EM Chapter-15.indd 296 24-01-2020 12:52:57

Note

cursor.fetchall() -fetchall () method is to fetch all rows from the database table

cursor.fetchone() - The fetchone () method returns the next row of a query result set or None in case there is no row left.

cursor.fetchmany() method that returns the next number of rows (n) of the result set

15.4.1.2 Displaying A record using fetchone()

The fetchone() method returns the next row of a query result set or None in case there is no row left.

Example 15.4.1.2-1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("\nfetch one:")
res = cursor.fetchone()
print(res)
```

OUTPUT

```
fetch one:
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
```

15.4.1.3 Displaying all records using fetchone()

Using while loop and fetchone() method we can display all the records from a table.

Example 15.4.1.3 -1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("fetching all records one by one:")
result = cursor.fetchone()
while result is not None:
    print(result)
    result = cursor.fetchone()
```

OUTPUT

```
fetching all records one by one:
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
(6, 'PRIYA', 'A', 'F', 98.6, '2002-01-01')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```


XII Std - CS EM Chapter-15.indd 297 24-01-2020 12:52:57

15.4.1.4 Displayingusing fetchmany()

Displaying specified number of records is done by using fetchmany(). This method returns the next number of rows (n) of the result set.

Example 15.4.1.4-1: Program to display the content of tuples using fetchmany()

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("fetching first 3 records:")
result = cursor.fetchmany(3)
print(result)
```

OUTPUT

```
fetching first 3 records:
[(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12'), (2, 'Aravind', 'A', 'M', 92.5, '2000-08-17'), (3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')]
```

Example 15.4.1.4-2: Program to display the content of tuples in newline without using loops

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student")
print("fetching first 3 records:")
result = cursor.fetchmany(3)
print(*result,sep="\n") # * is used for unpacking a tuple.
```

OUTPUT

```
fetching first 3 records:
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
```

Note

symbol is used to print the list of all elements in a single line with space. To print all elements in new lines or separated by space use sep= "\n" or sep= "," respectively.

XII Std - CS EM Chapter-15.indd 298 24-01-2020 12:52:57

15.4.2 CLAUSES IN SQL

SQL provides various clauses that can be used in the SELECT statements. This clauses can be called through python script. Almost all clauses will work with SQLite. The following frequently used clauses are discussed here

- DISTINCT
- WHERE
- GROUP BY
- ORDER BY.
- HAVING

15.4.2.1 SQL DISTINCT CLAUSE

The distinct clause is helpful when there is need of avoiding the duplicate values present in any specific columns/table. When we use distinct keyword only the unique values are fetched. In this example we are going to display the different grades scored by students from "student table".

Example 15.4.2.1-1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT DISTINCT (Grade) FROM student")
result = cursor.fetchall()
print(result)
```

OUTPUT

```
[('B',), ('A',), ('C',), ('D',)]
```

Without the keyword "distinct" in the above example displays 7 records instead of 4, since in the original table there are actually 7 records and some are with the duplicate values.

15.4.2.2 SQL WHERE CLAUSE

The WHERE clause is used to extract only those records that fulfill a specified condition. In this example we are going to display the different grades scored by male students from "student table"

XII Std - CS EM Chapter-15.indd 299 24-01-2020 12:52:58

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT DISTINCT (Grade) FROM student where gender='M'")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT

```
('B',)
('A',)
('C',)
('D',)
```

15.4.2.3 SQL Group By Clause

The SELECT statement can be used along with GROUP BY clause. The GROUP BY clause groups records into summary rows. It returns one records for each group. It is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns. The following example count the number of male and female from the student table and display the result.

Example 15.4.2.3 -1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT gender,count(gender) FROM student Group BY gender")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT

```
('F', 2)
('M', 5)
```

15.4.2.4 SQL ORDER BY Clause

The ORDER BY Clause can be used along with the SELECT statement to sort the data of specific fields in an ordered way. It is used to sort the result-set in ascending or descending order. In this example name and Rollno of the students are displayed in alphabetical order of names

XII Std - CS EM Chapter-15.indd 300 24-01-2020 12:52:58

Example 15.4.2.4 -1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT Rollno,sname  FROM student Order BY sname")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT

```
(1, 'Akshay')
(2, 'Aravind')
(3, 'BASKAR')
(6, 'PRIYA')
(4, 'SAJINI')
(7, 'TARUN')
(5, 'VARUN')
```

15.4.2.5 SQL HAVING Clause

Having clause is used to filter data based on the group functions. This is similar to WHERE condition but can be used only with group functions. Group functions cannot be used in WHERE Clause but can be used in HAVING clause.

Example 15.4.2.5 -1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT GENDER,COUNT(GENDER) FROM Student GROUP BY
GENDER HAVING COUNT(GENDER)>3")
result = cursor.fetchall()
co = [i[0] for i in cursor.description]
print(co)
print(result)
```

OUTPUT

```
['gender', 'COUNT(GENDER)']
[('M', 5)]
```

15.5 The SQL AND, OR and NOT Operators

The WHERE clause can be combined with AND, OR, and NOT operators. The AND and OR operators are used to filter records based on more than one condition. In this example you are going to display the details of students who have scored other than 'A' or 'B' from the "student table"

XII Std - CS EM Chapter-15.indd 301 24-01-2020 12:52:58

Example for WHERE WITH NOT Operator

Example 15.5 -1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT * FROM student where grade<>'A' and Grade<>'B'")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT

```
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```

Example for WHERE WITH AND Operator

In this example we are going to display the name, Rollno and Average of students who have scored an average between 80 to 90% (both limits are inclusive)

Example 15.5 -2

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT Rollno,Same,Average FROM student WHERE
(Average>=80 AND Average<=90)")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT

```
(1, 'Akshay', 87.8)
(5, 'VARUN', 80.6)
```

Example for WHERE WITH OR Operator

In this example we are going to display the name and Rollno of students who have not scored an average between 60 to 70%

XII Std - CS EM Chapter-15.indd 302 24-01-2020 12:52:58

Example 15.5 -3

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT Rollno,sname  FROM student WHERE (Average<60
OR  Average>70)")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT

```
(1, 'Akshay')
(2, 'Aravind')
(3, 'BASKAR')
(4, 'SAJINI')
(5, 'VARUN')
(6, 'PRIYA')
```

15.6 Querying A Date Column

In this example we are going to display the name and grade of students who have born in the year 2001

Example 15.6 -1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT Rollno,sname FROM student
WHERE(Birth_date>='2001-01-01' AND Birth_date<='2001-12-01')")
result = cursor.fetchall()
print(*result,sep="\n")
```

OUTPUT

```
(5, 'VARUN')
```

15.7 Aggregate Functions

These functions are used to do operations from the values of the column and a single value is returned.

XII Std - CS EM Chapter-15.indd 303 24-01-2020 12:52:58

- COUNT()
- AVG()
- SUM()
- MAX()
- MIN()

15.7.1 COUNT() function

The SQL COUNT() function returns the number of rows in a table satisfying the criteria specified in the WHERE clause. COUNT() returns 0 if there were no matching rows.

Example 15.7.1-1

Example 1 : In this example we are going to count the number of records(rows)

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT COUNT(*)  FROM student ")
result = cursor.fetchall()
print(result)
```

Output:

```
[(7,)]
```

EXAMPLE 15.7.1-2

Example 2 : In this example we are going to count the number of records by specifying a column

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT COUNT(AVERAGE)  FROM student ")
result = cursor.fetchall()
print(result
```

Output:

```
[(7,)]
```

Note

NULL values are not counted. In case if we had null in one of the records in student table for example in Average field then the output would be 6

XII Std - CS EM Chapter-15.indd 304 24-01-2020 12:52:58

15.7.2AVG():

The following SQL statement in the python program finds the average mark of all students.

Example 15.7.2-1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT AVG(AVERAGE)  FROM student ")
result = cursor.fetchall()
print(result)
```

OUTPUT

```
[(84.65714285714286,)]
```

Note

NULL values are ignored.

15.7.3 SUM():

The following SQL statement in the python program finds the sum of all average in the Average field of "Student table".

Example 15.7.1-3

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("SELECT SUM(AVERAGE)  FROM student ")
result = cursor.fetchall()
print(result)
```

OUTPUT

```
[(592.6,)]
```

Note

NULL values are ignored.

XII Std - CS EM Chapter-15.indd 305 24-01-2020 12:52:58

15.7.4 MAX() AND MIN() FUNCTIONS

The MAX() function returns the largest value of the selected column.

The MIN() function returns the smallest value of the selected column.

The following example show the highest and least average student's name.

Example 15.7.4-1

```
import sqlite3
connection = sqlite3.connect("Organization.db")
cursor = connection.cursor()
print("Displaying the name of the Highest Average")
cursor.execute("SELECT sname,max(AVERAGE)  FROM student ")
result = cursor.fetchall()
print(result)
print("Displaying the name of the Least Average")
cursor.execute("SELECT sname,min(AVERAGE)  FROM student ")
result = cursor.fetchall()
print(result)
```

OUTPUT

```
Displaying the name of the Highest Average
[('PRIYA', 98.6)]
Displaying the name of the Least Average
[('TARUN', 62.3)]
```

15.8 Updating A Record

You can even update a record (tuple) in a table through python script. The following example change the name "Priya" to "Priyanka" in a record in "student table"

XII Std - CS EM Chapter-15.indd 306 24-01-2020 12:52:58

Example 15.8 -1

```
# code for update operation
import sqlite3
# database name to be passed as parameter
conn = sqlite3.connect("Academy.db")
# update the student record
conn.execute("UPDATE Student SET sname ='Priyanka' where Rollno='6'")
conn.commit()
print ("Total number of rows updated :", conn.total_changes)
cursor = conn.execute("SELECT * FROM Student")
for row in cursor:
        print (row)
conn.close()
```

OUTPUT

```
Total number of rows updated : 1
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
(6, 'Priyanka', 'A', 'F', 98.6, '2002-01-01')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```

Note

Remember throughout this chapter student table what we have created is taken as example to explain the SQL queries .Example 15.3.2 -2 contain the student table with records

XII Std - CS EM Chapter-15.indd 307 24-01-2020 12:52:58

15.9 Deletion Operation

Similar to Sql command to delete a record, Python also allows to delete a record. The following example delete the content of Rollno 2 from "student table"

Example 15.9-1

```
# code for delete operation
import sqlite3
 # database name to be passed as parameter
conn = sqlite3.connect("Academy.db")
# delete student record from database
conn.execute("DELETE from Student where Rollno='2'")
conn.commit()
print("Total number of rows deleted :", conn.total_changes)
cursor =conn.execute("SELECT * FROM Student")
for row in cursor:
        print(row)
conn.close()
```

OUTPUT

```
Total number of rows deleted : 1
(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')
(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')
(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')
(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')
(6, 'Priyanka', 'A', 'F', 98.6, '2002-01-01')
(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')
```


XII Std - CS EM Chapter-15.indd 308 24-01-2020 12:52:58

15.10 Data input by User

In this example we are going to accept data using Python input() command during runtime and then going to write in the Table called "Person"

Example 15.10 -1

```
# code for executing query using input data
import sqlite3
# creates a database in RAM
con =sqlite3.connect("Academy.db")
cur =con.cursor()
cur.execute("DROP Table person")
cur.execute("create table person (name, age, id)")
print("Enter 5 students names:")
who =[input() for i in range(5)]
print("Enter their ages respectively:")
age =[int(input()) for i in range(5)]
print("Enter their ids respectively:")
p_id =[int(input())for i in range(5)]
n =len(who)
for i in range(n):
# This is the q-mark style:
        cur.execute("insert into person values (?, ?, ?)", (who[i], age[i], p_id[i]))
cur.execute("select * from person")
# Fetches all entries from table
print("Displaying All the Records From Person Table")
print (*cur.fetchall(), sep='\n' )
```


XII Std - CS EM Chapter-15.indd 309 24-01-2020 12:52:58

OUTPUT

```
Enter 5 students names:
        RAM
        KEERTHANA
        KRISHNA
        HARISH
        GIRISH
Enter their ages respectively:
        28
        12
        21
        18
        16
Enter their ids respectively:
        1
        2
        3
        4
        5
Displaying All the Records From Person Table
        ('RAM', 28, 1)
        ('KEERTHANA', 12, 2)
        ('KRISHNA', 21, 3)
        ('HARISH', 18, 4)
        ('GIRISH', 16, 5)
```

You can even add records to the already existing table like "Student" Using the above coding with appropriate modification in the Field Name. To do so you should comment the create table statement

Note

Execute (sql[, parameters]) :- Executes a single SQL statement. The SQL statement may be parametrized (i. e. Use placeholders instead of SQL literals). The sqlite3 module supports two kinds of placeholders: question marks? ("qmark style") and named placeholders :name ("named style").

XII Std - CS EM Chapter-15.indd 310 24-01-2020 12:52:58

15.11 Using Multiple Table for Querying

Python allows to query more than one table by joining them. In the following example a new table called "Appointments" which contain the details of students Rollno, Duty, Age is created. The tables "student" and "Appointments" are joined and displayed the result with the column headings.

Example 15.11-1

```
import sqlite3
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
cursor.execute("""DROP TABLE Appointment;""")
sql_command = """
CREATE TABLE Appointment(rollnointprimarkey,Dutyvarchar(10),age int)"""
cursor.execute(sql_command)
sql_command = """INSERT INTO Appointment (Rollno,Duty ,age )
        VALUES ("1", "Prefect", "17");"""
cursor.execute(sql_command)
sql_command = """INSERT INTO Appointment (Rollno, Duty, age)
        VALUES ("2", "Secretary", "16");"""
cursor.execute(sql_command)
#  never forget this, if you want the changes to be saved:
connection.commit()
cursor.execute ("SELECT student.rollno,student.sname,
        Appointment.Duty, Appointment.Age FROM student,Appointment
        where student.rollno=Appointment.rollno")
#print (cursor.description) to display the field names of the table
co = [i[0] for i in cursor.description]
print(co)
#  Field informations can be read from cursor.description.
result = cursor.fetchall()
for r in result:
        print(r)
```

OUTPUT

```
['Rollno', 'Sname', 'Duty', 'age']
(1, 'Akshay', 'Prefect', 17)
(2, 'Aravind', 'Secretary', 16)
```

Note

cursor. description contain the details of each column headings .It will be stored as a tuple and the first one that is 0(zero) index refers to the column name. From index 1 onwards the values of the column(Records) are refered. Using this command you can display the table's Field names.

15.12 Integrating Query With Csv File

You can even store the query result in a CSV file. This will be useful to display the query output in a tabular format. In the following example **(EXAMPLE 15.12 -1)** Using Python script the student table is sorted "gender" wise in descending order and then arranged the records alphabetically. The output of this Query will be written in a CSV file called "SQL.CSV", again the content is read from the CSV file and displayed the result.

Example 15.12 -1

```
import sqlite3
import io                                   #  to access replace()
import csv
                                            #  CREATING CSV FILE
d=open('c:/pyprg/sql.csv','w')
c=csv.writer(d)
connection = sqlite3.connect("Academy.db")
cursor = connection.cursor()
                                            #  a=Connection.cursor()
cursor.execute("SELECT * FROM student ORDER BY GENDER DESC,SNAME")
                                            #  WRITING THE COLUMN HEADING
co = [i[0] for i in cursor.description]
c.writerow(co)
data=cursor.fetchall()
for item in data:
        c.writerow(item)
d.close()
#  Reading the CSV File
#  replace() is used to eliminate the newline character at the end of each row
with open('c:/pyprg/sql.csv', "r", newline=None) as fd:
        #  r = csv.reader(fd)
        for line in fd:
                line = line.replace("\n", "")
                print(line)
cursor.close()
connection.close()
```

OUTPUT

```
Rollno,Sname,Grade,gender,Average,birth_date
1, Akshay, B, M, 87.8, 2001-12-12
2, Aravind, A, M, 92.5, 2000-08-17
3, BASKAR, C, M, 75.2, 1998-05-17
7, TARUN, D, M, 62.3, 1999-02-01
5, VARUN, B, M, 80.6, 2001-03-14
6, PRIYA, A, F, 98.6, 2002-01-01
4, SAJINI, A, F, 95.6, 2002-11-01
```


XII Std - CS EM Chapter-15.indd 312 24-01-2020 12:52:58

Example 15.12 -2 Opening the file ("sqlexcel.csv") through MS-Excel and view the result (Program is same similar to EXAMPLE 15.12 -1 script)

```
import sqlite3
import io                                   #to access replace()
import csv
# database name to be passed as parameter
conn = sqlite3.connect("Academy.db")
print("Content of the table before sorting and writing in CSV file")
cursor = conn.execute("SELECT * FROM Student")
for row in cursor:
        print (row)
#  CREATING CSV FILE
d=open('c:\pyprg\sqlexcel.csv','w')
c=csv.writer(d)
cursor = conn.cursor()
cursor.execute("SELECT * FROM student ORDER BY GENDER DESC,SNAME")
#WRITING THE COLUMN HEADING
co = [i[0] for i in cursor.description]
c.writerow(co)
data=cursor.fetchall()
for item in data:
        c.writerow(item)
d.close()
print("sqlexcel.csv File is created open by visiting c:\pyprg\sqlexcel.csv")
conn.close()
```

Note

By default while writing in a csv file each record ends with \n (newline) to eliminate this newline replace () is used during the reading of csv file.

XII Std - CS EM Chapter-15.indd 313 24-01-2020 12:52:58

OUTPUT

Content of the table before sorting and writing in CSV file

(1, 'Akshay', 'B', 'M', 87.8, '2001-12-12')

(2, 'Aravind', 'A', 'M', 92.5, '2000-08-17')

(3, 'BASKAR', 'C', 'M', 75.2, '1998-05-17')

(4, 'SAJINI', 'A', 'F', 95.6, '2002-11-01')

(5, 'VARUN', 'B', 'M', 80.6, '2001-03-14')

(6, 'Priyanka', 'A', 'F', 98.6, '2002-01-01')

(7, 'TARUN', 'D', 'M', 62.3, '1999-02-01')

sqlexcel.csv File is created open by visiting c:\pyprg\sqlexcel.csv

OUTPUT THROUGH EXCEL

	A	B	C	D	E	F
1	Rollno	Sname	Grade	gender	Average	birth_date
2						
3	1	Akshay	B	M	87.8	2001-12-12
4						
5	2	Aravind	A	M	92.5	2000-08-17
6						
7	3	BASKAR	C	M	75.2	1998-05-17
8						
9	7	TARUN	D	M	62.3	1999-02-01
10						
11	5	VARUN	B	M	80.6	2001-03-14
12						
13	6	PRIYA	A	F	98.6	2002-01-01
14						
15	4	SAJINI	A	F	95.6	200-11-01

15.3 Table List

To show (display) the list of tables created in a database the following program **(Example 15.3 – 1)** can be used.

Example 15.3 – 1

```
import sqlite3
con = sqlite3.connect('Academy.db')
cursor = con.cursor()
cursor.execute("SELECT name FROM sqlite_master WHERE type='table';")
print(cursor.fetchall())
```

OUTPUT

[('Student',), ('Appointment',), ('Person',)]

XII Std - CS EM Chapter-15.indd 314 24-01-2020 12:52:58

The above program (Example 15.3-1) display the names of all tables created in 'Academy.db' database. **The master table holds the key information about your database tables and it is called sqlite_master**

So far, you have been using the Structured Query Language in Python scripts. This chapter has covered many of the basic SQL commands. Almost all sql commands can be executed by Python SQLite module. You can even try the other commands discussed in the SQL chapter.

Points to remember:

- A database is an organized collection of data.
- Users of database can be human users, other programs or applications
- SQLite is a simple relational database system, which saves its data in regular data files.
- Cursor is a control structure used to traverse and fetch the records of the database. All the SQL commands will be executed using cursor object only.
- As data in a table might contain single or double quotes, SQL commands in Python are denoted as triple quoted string.
- "Select" is the most commonly used statement in SQL
- The SELECT Statement in SQL is used to retrieve or fetch data from a table in a database
- The GROUP BY clause groups records into summary rows
- The ORDER BY Clause can be used along with the SELECT statement to sort the data of specific fields in an ordered way
- Having clause is used to filter data based on the group functions.
- Where clause cannot be used along with 'Group by'
- The WHERE clause can be combined with AND, OR, and NOT operators
- The 'AND' and 'OR' operators are used to filter records based on more than one condition
- Aggregate functions are used to do operations from the values of the column and a single value is returned.
- COUNT() function returns the number of rows in a table.
- AVG() function retrieves the average of a selected column of rows in a table.
- SUM() function retrieves the sum of a selected column of rows in a table.
- MAX() function returns the largest value of the selected column.
- MIN() function returns the smallest value of the selected column
- sqlite_master is the master table which holds the key information about your database tables.
- The path of a file can be either represented as '/' or using '\' in Python. For example the path can be specified either as 'c:/pyprg/sql.csv', or c:\pyprg\sql.csv'.

XII Std - CS EM Chapter-15.indd 315 24-01-2020 12:52:58

Hands on Experience

1. Create an interactive program to accept the details from user and store it in a csv file using Python for the following table.

   Database name;- DB1

   Table name : Customer

Cust_Id	Cust_Name	Address	Phone_no	City
C008	Sandeep	14/1 Pritam Pura	41206819	Delhi
C010	Anurag Basu	15A, Park Road	61281921	Kolkata
C012	Hrithik	7/2 Vasant Nagar	26121949	Delhi

2. Consider the following table GAMES. Write a python program to display the records for question (i) to (iv) and give outputs for SQL queries (v) to (viii)

   Table: GAMES

Gcode	Name	GameName	Number	PrizeMoney	ScheduleDate
101	Padmaja	Carom Board	2	5000	01-23-2014
102	Vidhya	Badminton	2	12000	12-12-2013
103	Guru	Table Tennis	4	8000	02-14-2014
105	Keerthana	Carom Board	2	9000	01-01-2014
108	Krishna	Table Tennis	4	25000	03-19-2014

(i) To display the name of all Games with their Gcodes in descending order of their schedule date.

(ii) To display details of those games which are having Prize Money more than 7000.

(iii) To display the name and gamename of the Players in the ascending order of Gamename.

(iv) To display sum of PrizeMoney for each of the Numberof participation groupings (as shown in column Number 4)

(v) Display all the records based on GameName

XII Std - CS EM Chapter-15.indd 316 24-01-2020 12:52:58

Evaluation

RDEZQU

Part - I

1. Which of the following is an organized collection of data?

   (A) Database (B) DBMS (C) Information (D) Records

2. SQLite falls under which database system?

   (A) Flat file database system (B) Relational Database system

   (C) Hierarchical database system (D) Object oriented Database system

3. Which of the following is a control structure used to traverse and fetch the records of the database?

   (A) Pointer (B) Key

   (C) Cursor (D) Insertion point

4. Any changes made in the values of the record should be saved by the command

   (A) Save (B) Save As (C) Commit (D) Oblige

5. Which of the following executes the SQL command to perform some action?

   (A) execute() (B) key() (C) cursor() (D) run()

6. Which of the following function retrieves the average of a selected column of rows in a table?

   (A) Add() (B) SUM() (C) AVG() (D) AVERAGE()

7. The function that returns the largest value of the selected column is

   (A) MAX() (B) LARGE()

   (C) HIGH() (D) MAXIMUM()

8. Which of the following is called the master table?

   (A) sqlite_master (B) sql_master

   (C) main_master (D) master_main

9. The most commonly used statement in SQL is

   (A) cursor (B) select (C) execute (D) commit

10. Which of the following clause avoid the duplicate?

    (A) Distinct (B) Remove (C) Where (D) GroupBy

XII Std - CS EM Chapter-15.indd 317 24-01-2020 12:52:58

Part - II

1. Mention the users who uses the Database.
2. Which method is used to connect a database? Give an example.
3. What is the advantage of declaring a column as "INTEGER PRIMARY KEY"
4. Write the command to populate record in a table. Give an example.
5. Which method is used to fetch all rows from the database table?

Part - III

1. What is SQLite?What is it advantage?
2. Mention the difference between fetchone() and fetchmany()
3. What is the use of Where Clause.Give a python statement Using the where clause.
4. Read the following details.Based on that write a python script to display department wise records

   database name :- organization.db

   Table name :- Employee

   Columns in the table :- Eno, EmpName, Esal, Dept
5. Read the following details.Based on that write a python script to display records in desending order of

   Eno

   database name :- organization.db

   Table name :- Employee

   Columns in the table :- Eno, EmpName, Esal, Dept

Part - IV

1. Write in brief about SQLite and the steps used to use it.
2. Write the Python script to display all the records of the following table using fetchmany()

Icode	ItemName	Rate
1003	Scanner	10500
1004	Speaker	3000
1005	Printer	8000
1008	Monitor	15000

XII Std - CS EM Chapter-15.indd 318 24-01-2020 12:52:58

1010	Mouse	700

3. hat is the use of HAVING clause. Give an example python script
4. Write a Python script to create a table called ITEM with following specification.

   Add one record to the table.

   Name of the database :- ABC

   Name of the table :- Item

   Column name and specification :-

Icode	:-	integer and act as primary key
Item Name	:-	Character with length 25
Rate	:-	Integer
Record to be added	:-	1008, Monitor,15000

5. Consider the following table Supplier and item .Write a python script for (i) to (ii)

SUPPLIER				
Suppno	**Name**	**City**	**Icode**	**SuppQty**
S001	Prasad	Delhi	1008	100
S002	Anu	Bangalore	1010	200
S003	Shahid	Bangalore	1008	175
S004	Akila	Hydrabad	1005	195
S005	Girish	Hydrabad	1003	25
S006	Shylaja	Chennai	1008	180
S007	Lavanya	Mumbai	1005	325

   i) Display Name, City and Itemname of suppliers who do not reside in Delhi.

   ii) Increment the SuppQty of Akila by 40

References

1. *The Definitive Guide to SQLite by Michael Owens*
2. *Programming for Beginners: 2 Manuscripts: SQL & Python by Byron Francis*
3. *Tutorialspoint.com*

Unit V

CHAPTER 16

DATA VISUALIZATION USING PYPLOT: LINE CHART, PIE CHART AND BAR CHART

Learning Objectives

RSXUS6

After learning this chapter, the learners will be able to

- Define the term Data Visualization.
- List the types of Data Visualization.
- List the uses of Data Visualization.
- List the types of Visualizations in Matplotlib.
- Practice installing Matplotlib.
- Explore importing Matplotlib.
- Classify the types of Data Visualization plots.
- Practice creating various types of plots using Matplotlib.

16.1 Data Visualization Definition

Data Visualization is the graphical representation of information and data. The objective of Data Visualization is to communicate information visually to users. For this, data visualization uses statistical graphics. Numerical data may be encoded using dots, lines, or bars, to visually communicate a quantitative message.

General types of Data Visualization

- Charts
- Tables
- Graphs
- Maps
- Infographics
- Dashboards

Data visualization - Uses

- Data Visualization help users to analyze and interpret the data easily.

XII Std - CS EM Chapter-16.indd 320 24-01-2020 12:56:33

- It makes complex data understandable and usable.
- Various Charts in Data Visualization helps to show relationship in the data for one or more variables.

DO YOU KNOW?

Infographics → An infographic (information graphic) is the representation of information in a graphic format.

Dashboard → A dashboard is a collection of resources assembled to create a single unified visual display. Data visualizations and dashboards translate complex ideas and concepts into a simple visual format. Patterns and relationships that are undetectable in text are detectable at a glance using dashboard.

Introduction to Matplotlib — Data Visualization in Python

Matplotlib is the most popular data visualization library in Python. It allows you to create charts in few lines of code.

Types of Visualizations in Matplotlib

There are many types of Visualizations under Matplotlib. Some of them are:

- Line plot
- Scatter plot
- Histogram
- Box plot
- Bar chart and
- Pie chart

DO YOU KNOW?

Scatter plot: A scatter plot is a type of plot that shows the data as a collection of points. The position of a point depends on its two-dimensional value, where each value is a position on either the horizontal or vertical dimension.

Box plot: The box plot is a standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum.

Installing Matplotlib

You can install matplotlib using pip. Pip is a management software for installing python packages.

XII Std - CS EM Chapter-16.indd 321 24-01-2020 12:56:33

Note

Detailed installation procedures given in Annexure

16.2 Getting Started

After installing Matplotlib, we will begin coding by importing Matplotlib using the command:

```
import matplotlib.pyplot as plt
```

Now you have imported Matplotlib in your workspace. You need to display the plots. Using Matplotlib from within a Python script, you have to add plt.show() method inside the file to display your plot.

Example

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.show()
```

Output

This window is a matplotlib window, which allows you to see your graph. You can hover the graph and see the coordinates in the bottom right.

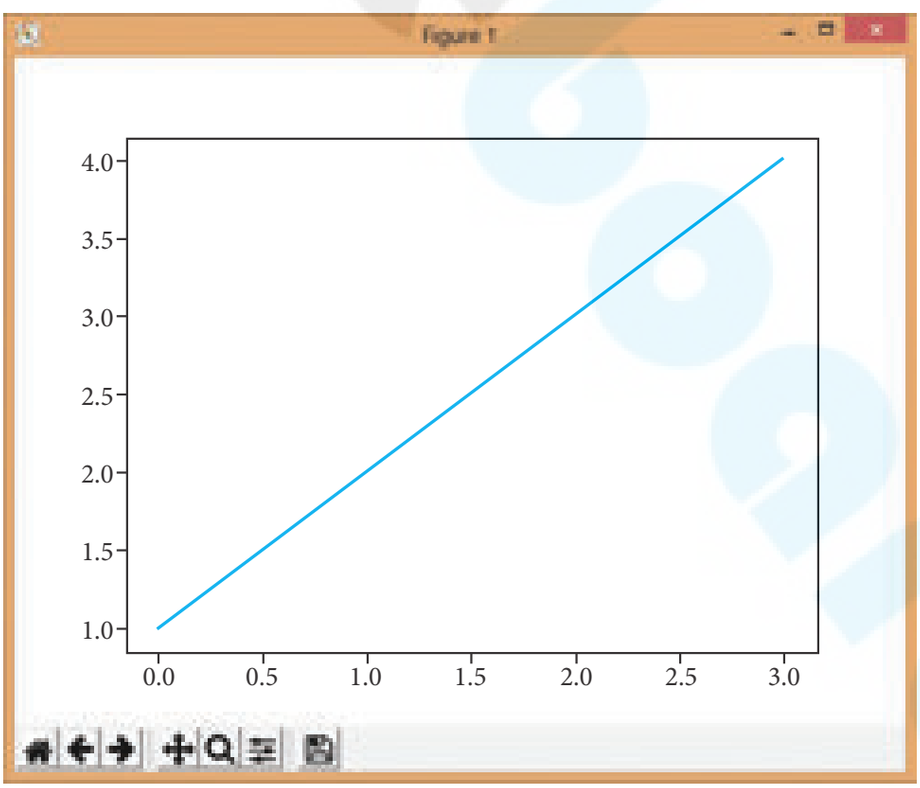


Figure 16.1

XII Std - CS EM Chapter-16.indd 322 24-01-2020 12:56:33

You may be wondering why the x-axis ranges from 0-3 and the y-axis from 1-4. If you provide a single list or array to the plot () command, matplotlib assumes it is a sequence of y values, and automatically generates the x values for you. Since python ranges start with 0, the default x vector has the same length as y but starts with 0. Hence the x data are [0, 1, 2, 3].

plot() is a versatile command, and will take an arbitrary number of arguments.

Program

For example, to plot x versus y, you can issue the command:

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16])
plt.show()
```

This .plot takes many parameters, but the first two here are 'x' and 'y' coordinates. This means, you have 4 co-ordinates according to these lists: (1,1), (2,4), (3,9) and (4,16).

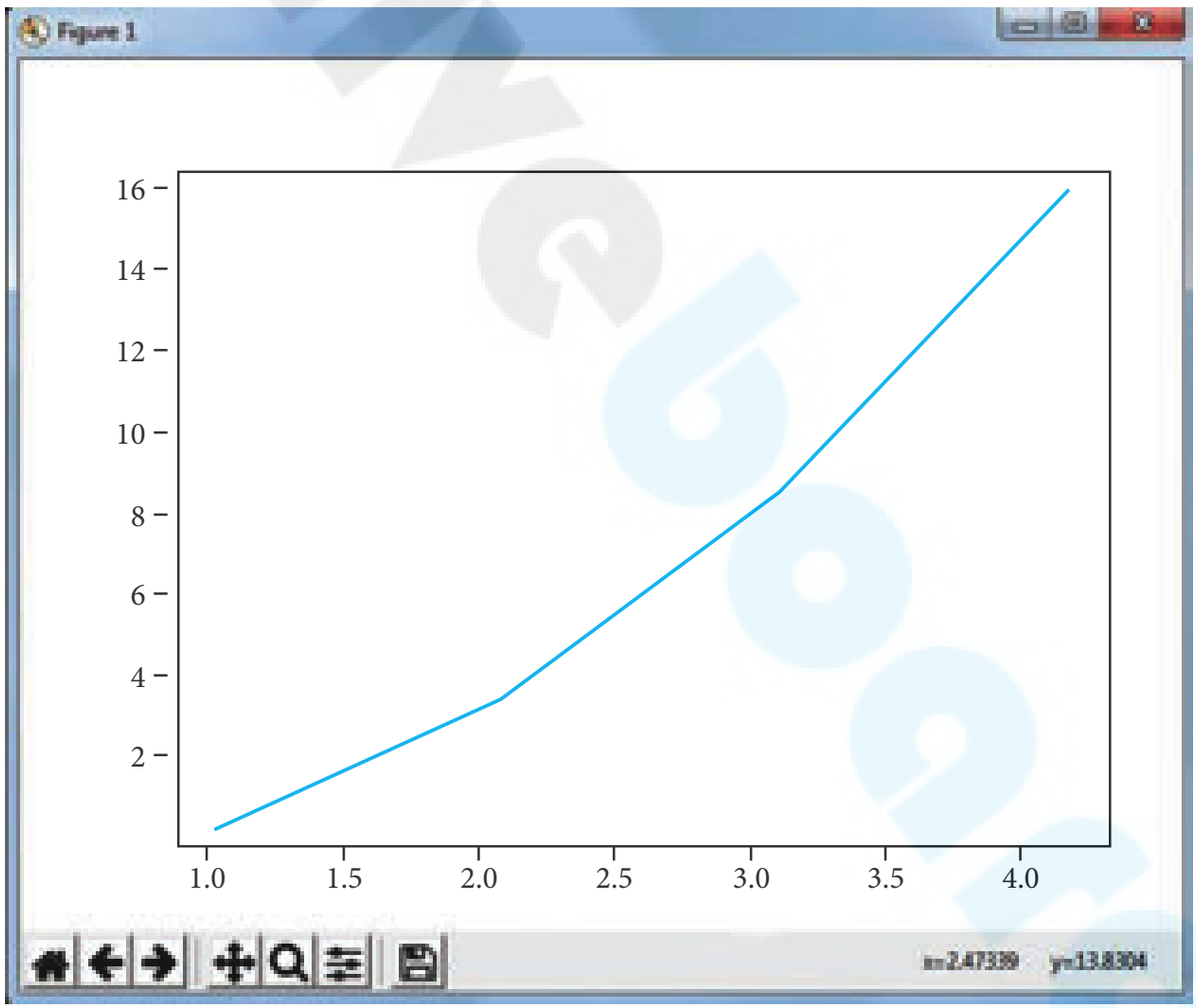


Figure 16.2

XII Std - CS EM Chapter-16.indd 323 24-01-2020 12:56:34

Plotting Two Lines

To plot two lines, use the following code:

```
import matplotlib.pyplot as plt
    x = [1,2,3]
    y = [5,7,4]
    x2 = [1,2,3]
    y2 = [10,14,12]
    plt.plot(x, y, label='Line 1')
    plt.plot(x2, y2, label='Line 2')
    plt.xlabel('X-Axis')
    plt.ylabel('Y-Axis')
    plt.title('LINE GRAPH')
    plt.legend()
    plt.show()
```

Output

With plt.xlabel and plt.ylabel, you can assign labels to those respective axis. Next, you can assign the plot's title with plt.title, and then you can invoke the default legend with plt. legend().

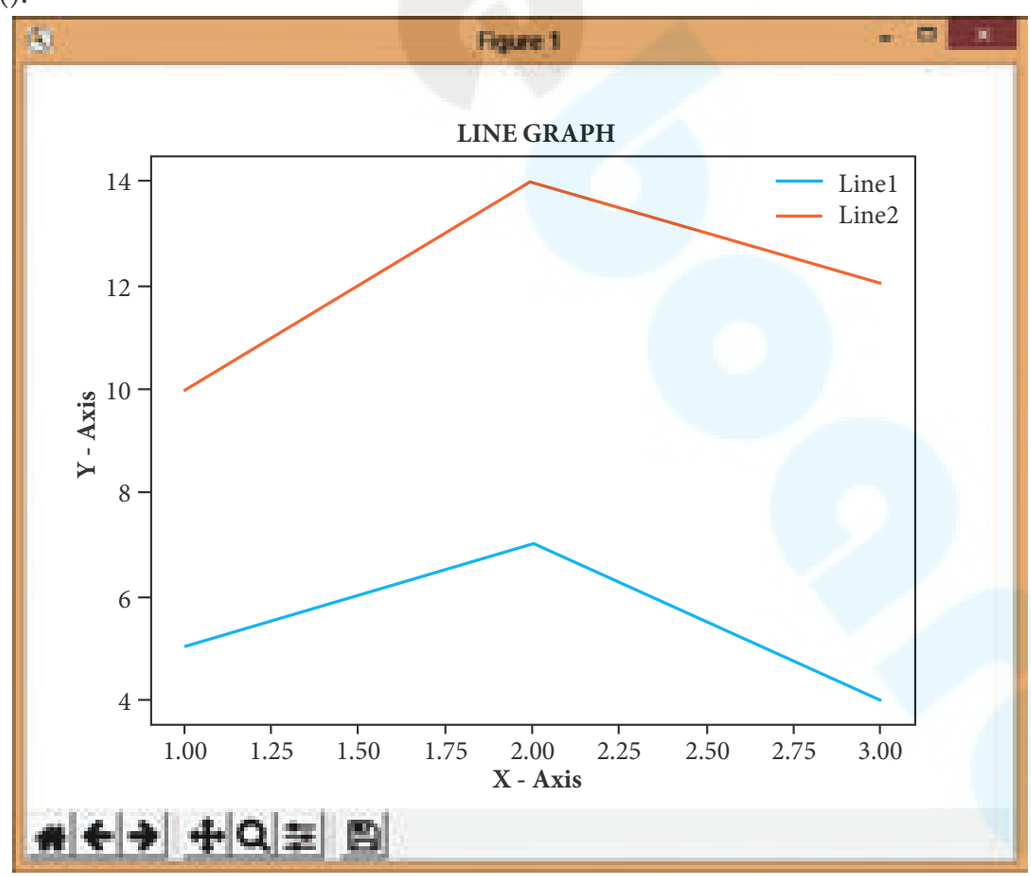


Figure 16.3

XII Std - CS EM Chapter-16.indd 324 24-01-2020 12:56:34

Buttons in the output

In the output figure, you can see few buttons at the bottom left corner. Let us see the use of these buttons.

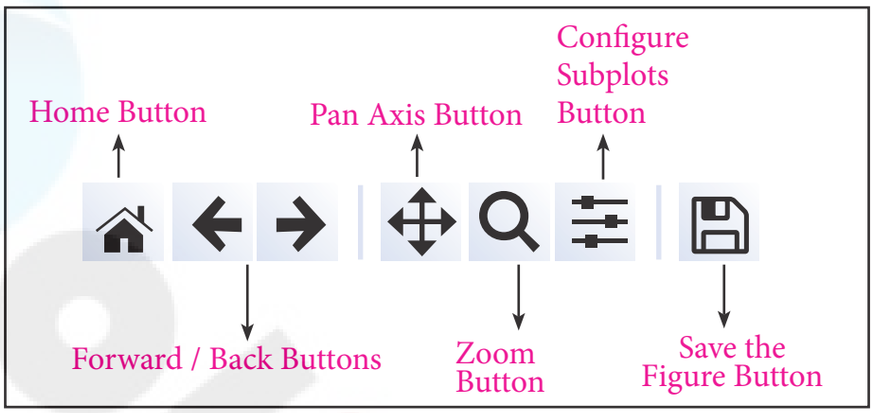


Figure 16.4

Home Button → The Home Button will help once you have begun navigating your chart. If you ever want to return back to the original view, you can click on this.

Forward/Back buttons → These buttons can be used like the Forward and Back buttons in your browser. You can click these to move back to the previous point you were at, or forward again.

Pan Axis → This cross-looking button allows you to click it, and then click and drag your graph around.

Zoom → The Zoom button lets you click on it, then click and drag a square that you would like to zoom into specifically. Zooming in will require a left click and drag. You can alternatively zoom out with a right click and drag.

Configure Subplots → This button allows you to configure various spacing options with your figure and plot.

Save Figure → This button will allow you to save your figure in various forms.

16.3 Special Plot Types

Matplotlib allows you to create different kinds of plots ranging from histograms and scatter plots to bar graphs and bar charts.

Line Chart

A Line Chart or Line Graph is a type of chart which displays information as a series of data points called 'markers' connected by straight line segments. A Line Chart is often used to visualize a trend in data over intervals of time – a time series – thus the line is often drawn chronologically.

XII Std - CS EM Chapter-16.indd 325 24-01-2020 12:56:34

Example: Line plot

```
import matplotlib.pyplot as plt
years = [2014, 2015, 2016, 2017, 2018]
total_populations = [8939007, 8954518, 8960387, 8956741, 8943721]
plt.plot (years, total_populations)
plt.title ("Year vs Population in India")
plt.xlabel ("Year")
plt.ylabel ("Total Population")
plt.show()
```

In this program,

Plt.title() → specifies title to the graph

Plt.xlabel() → specifies label for X-axis

Plt.ylabel() → specifies label for Y-axis

Output

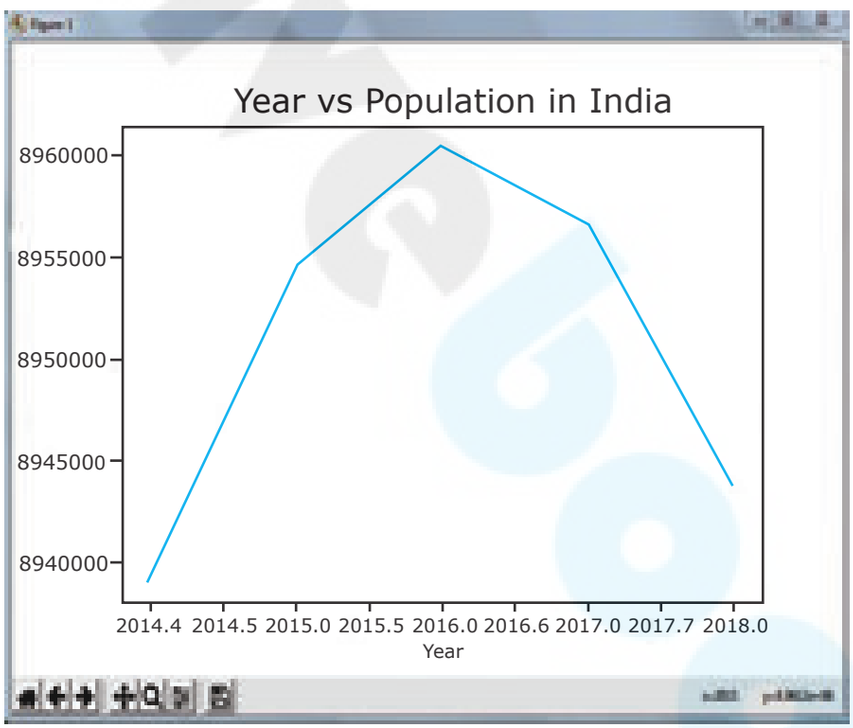


Figure 16.5

Bar Chart

A BarPlot (or BarChart) is one of the most common type of plot. It shows the relationship between a numerical variable and a categorical variable.

Bar chart represents categorical data with rectangular bars. Each bar has a height corresponds to the value it represents. The bars can be plotted vertically or horizontally. It's useful when we want to compare a given numeric value on different categories. To make a bar chart with Matplotlib, we can use the plt.bar() function.

XII Std - CS EM Chapter-16.indd 326 24-01-2020 12:56:34

Example

```
import matplotlib.pyplot as plt
# Our data
labels = ["TAMIL", "ENGLISH", "MATHS", "PHYSICS", "CHEMISTRY", "CS"]
usage = [79.8, 67.3, 77.8, 68.4, 70.2, 88.5]
# Generating the y positions. Later, we'll use them to replace them with labels.
y_positions = range (len(labels))
# Creating our bar plot
plt.bar (y_positions, usage)
plt.xticks (y_positions, labels)
plt.ylabel ("RANGE")
plt.title ("MARKS")
plt.show()
```

Output

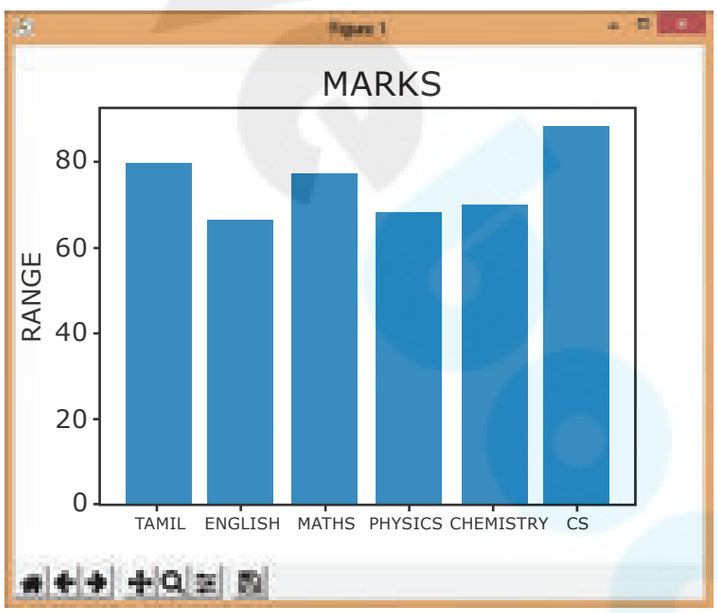


Figure 16.6

The above code represents the following:

Labels → Specifies labels for the bars.

Usgae → Assign values to the labels specified.

Xticks → Display the tick marks along the x-axis at the values represented. Then specify the label for each tick mark.

Range → Create sequence of numbers.

DO YOU KNOW?

Bar Graph and Histogram are the two ways to display data in the form of a diagram.

Key Differences Between Histogram and Bar Graph

The differences between Histogram and bar graph are as follows

1. Histogram refers to a graphical representation; that displays data by way of bars to show the frequency of numerical data. A bar graph is a pictorial representation of data that uses bars to compare different categories of data.
2. A histogram represents the frequency distribution of continuous variables. Conversely, a bar graph is a diagrammatic comparison of discrete variables.
3. Histogram presents numerical data whereas bar graph shows categorical data.
4. The histogram is drawn in such a way that there is no gap between the bars. On the ot her hand, there is proper spacing between bars in a bar graph that indicates discontinuity.
5. Items of the histogram are numbers, which are categorised together, to represent ranges of data. As opposed to the bar graph, items are considered as individual entities.
6. In the case of a bar graph, it is quite common to rearrange the blocks, from highest to lowest. But with histogram, this cannot be done, as they are shown in the sequence of classes.
7. The width of rectangular blocks in a histogram may or may not be same while the width of the bars in a bar graph is always same.

Pie Chart

Pie Chart is probably one of the most common type of chart. It is a circular graphic which is divided into slices to illustrate numerical proportion. The point of a pie chart is to show the relationship of parts out of a whole.

To make a Pie Chart with Matplotlib, we can use the ***plt.pie()*** function. The autopct parameter allows us to display the percentage value using the Python string formatting.

Example

```
import matplotlib.pyplot as plt

sizes = [89, 80, 90, 100, 75]

labels = ["Tamil", "English", "Maths", "Science", "Social"]

plt.pie (sizes, labels = labels, autopct = "%.2f")

plt.axes().set_aspect ("equal")

plt.show()
```


XII Std - CS EM Chapter-16.indd 328 24-01-2020 12:56:34

Output

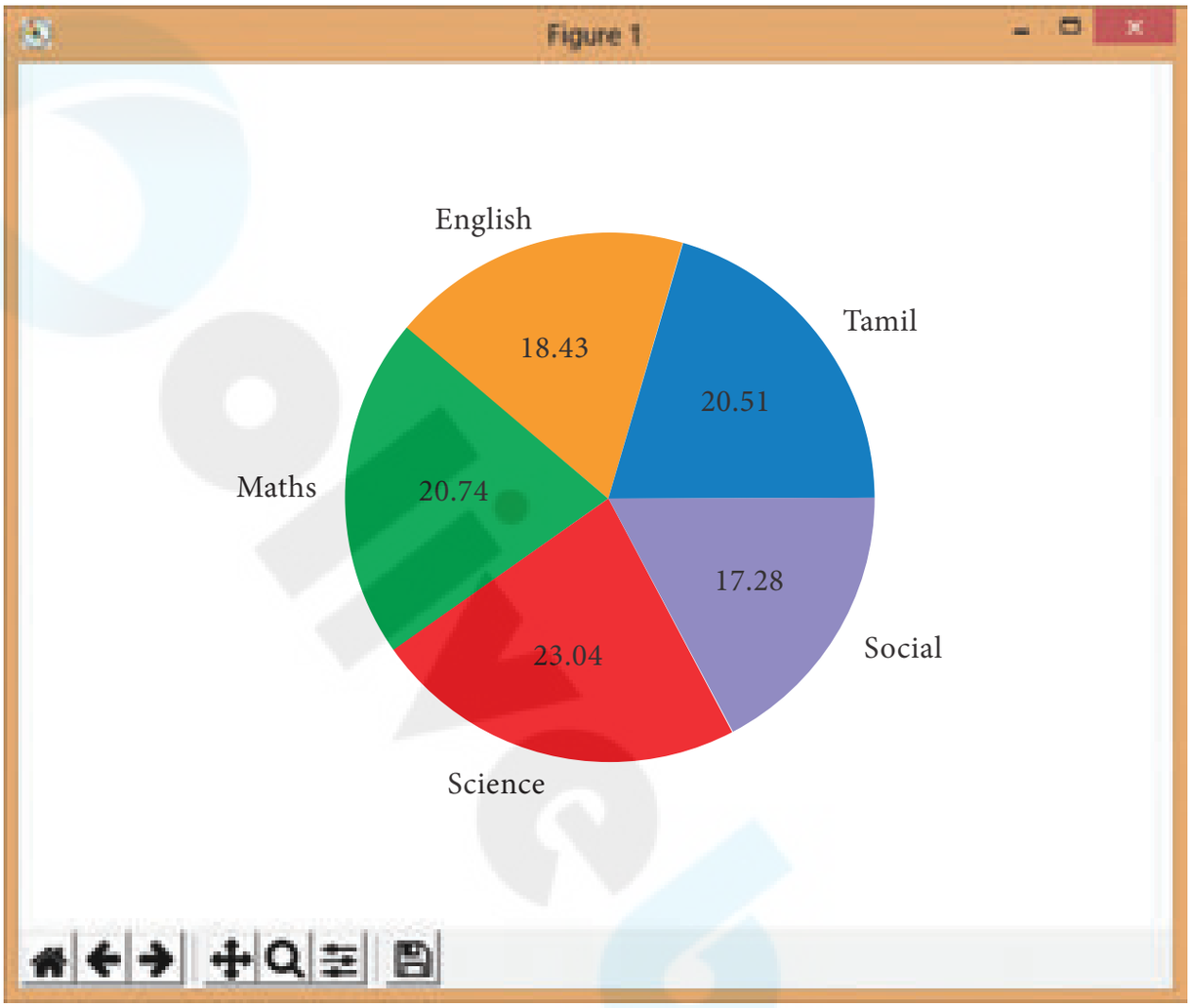


Figure 16.7

Hands on Practice

1. Create a plot. Set the title, the *x* and *y* labels for both axes.
2. Plot a pie chart for your marks in the recent examination.
3. Plot a line chart on the academic performance of Class 12 students in Computer Science for the past 10 years.
4. Plot a bar chart for the number of computer science periods in a week.

XII Std - CS EM Chapter-16.indd 329 24-01-2020 12:56:34

Evaluation

T39IUH

Part - I

1. Which is a python package used for 2D graphics?

   a. matplotlib.pyplot b. matplotlib.pip

   c. matplotlib.numpy d. matplotlib.plt

2. Identify the package manager for Python packages, or modules.

   a. Matplotlib b. PIP c. plt.show() d. python package

3. Read the following code: Identify the purpose of this code and choose the right option from the following.

   C:\Users\YourName\AppData\Local\Programs\Python\Python36-32\Scripts>pip –version

   a. Check if PIP is Installed b. Install PIP

   c. Download a Package d. Check PIP version

4. Read the following code: Identify the purpose of this code and choose the right option from the following.

   C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip list

   a. List installed packages b. list command

   c. Install PIP d. packages installed

5. To install matplotlib, the following function will be typed in your command prompt. What does "-U"represents?

   Python –m pip install –U pip

   a. downloading pip to the latest version

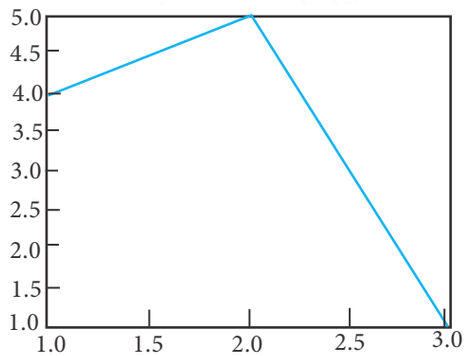
   b. upgrading pip to the latest version

   c. removing pip

   d. upgrading matplotlib to the latest version

6. Observe the output figure. Identify the coding for obtaining this output.

XII Std - CS EM Chapter-16.indd 330 24-01-2020 12:56:34



a. import matplotlib.pyplot as plt
   plt.plot([1,2,3],[4,5,1])
   plt.show()

b. import matplotlib.pyplot as plt
   plt.plot([1,2],[4,5])
   plt.show()

c. import matplotlib.pyplot as plt
   plt.plot([2,3],[5,1])
   plt.show()

d. import matplotlib.pyplot as plt
   plt.plot([1,3],[4,1])
   plt.show()

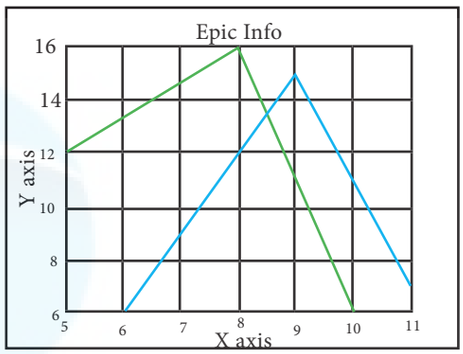
7. Read the code:

   a. import matplotlib.pyplot as plt
   b. plt.plot(3,2)
   c. plt.show()

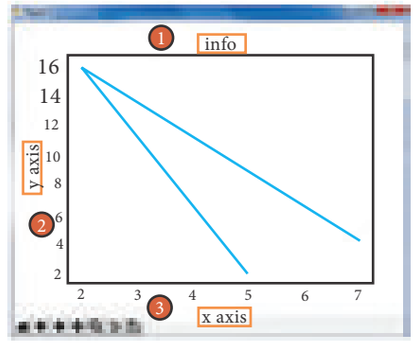
   Identify the output for the above coding.

XII Std - CS EM Chapter-16.indd 331 24-01-2020 12:56:34

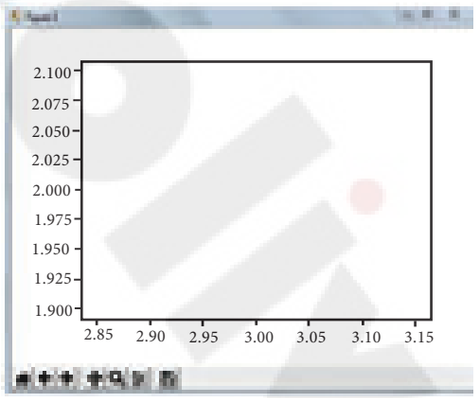
a.



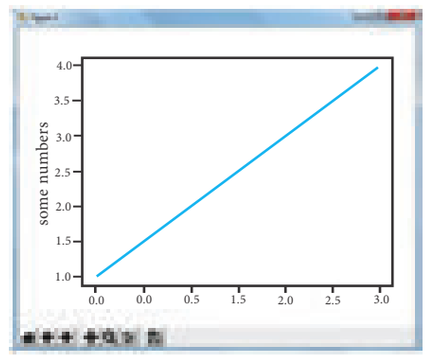
b.



c.



d.



8. Which key is used to run the module?

   a. F6 b. F4 c. F3 d. F5

9. Identify the right type of chart using the following hints.

   Hint 1: This chart is often used to visualize a trend in data over intervals of time.

   Hint 2: The line in this type of chart is often drawn chronologically.

   a. Line chart b. Bar chart c. Pie chart d. Scatter plot

10. Read the statements given below. Identify the right option from the following for pie chart.

    Statement A: To make a pie chart with Matplotlib, we can use the plt.pie() function.

    Statement B: The autopct parameter allows us to display the percentage value using the Python string formatting.

    a. Statement A is correct b. Statement B is correct

    c. Both the statements are correct d. Both the statements are wrong

Part - II

1. Define: Data Visualization.
2. List the general types of data visualization.
3. List the types of Visualizations in Matplotlib.
4. How will you install Matplotlib?

XII Std - CS EM Chapter-16.indd 332 24-01-2020 12:56:34

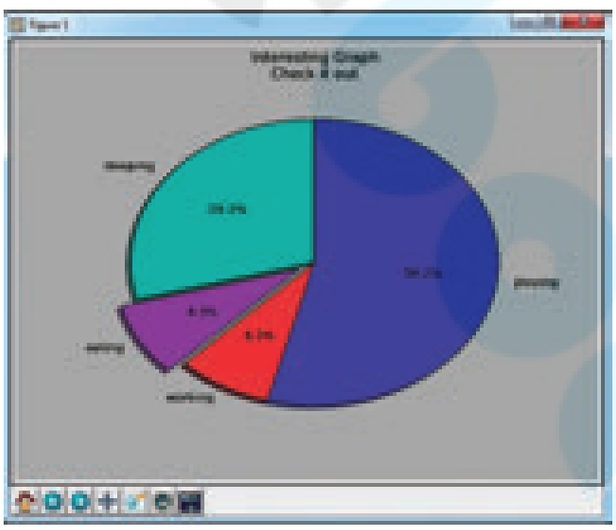
5. Write the difference between the following functions: plt.plot([1,2,3,4]), plt.plot([1,2,3,4], [1,4,9,16]).

Part - III

1. Draw the output for the following data visualization plot.

```
import matplotlib.pyplot as plt
plt.bar([1,3,5,7,9],[5,2,7,8,2], label="Example one")
plt.bar([2,4,6,8,10],[8,6,2,5,6], label="Example two", color='g')
plt.legend()
plt.xlabel('bar number')
plt.ylabel('bar height')
plt.title('Epic Graph\nAnother Line! Whoa')
plt.show()
```

2. Write any three uses of data visualization.
3. Write the coding for the following:
   a. To check if PIP is Installed in your PC.
   b. To Check the version of PIP installed in your PC.
   c. To list the packages in matplotlib.
4. Write the plot for the following pie chart output.



Part - IV

1. Explain in detail the types of pyplots using Matplotlib.
2. Explain the various buttons in a matplotlib window.
3. Explain the purpose of the following functions:
   a. plt.xlabel

XII Std - CS EM Chapter-16.indd 333 24-01-2020 12:56:34

b. plt.ylabel

c. plt.title

d. plt.legend()

e. plt.show()

Reference

1. https://towards datascience.com / data - science - with - python - intro -to- data -visualization-and-matplotlib-5f799b7c6d82.
2. https://heartbeat.fritz.ai/introduction-to-matplotlib-data-visualization-in-python-d9143287ae39.
3. https://python programming.net / legends - titles - labels - matplotlib - tutorial/?completed=/matplotlib-intro-tutorial/.
4. https://keydifferences.com/difference-between-histogram-and-bar-graph.html.

XII Std - CS EM Chapter-16.indd 334 24-01-2020 12:56:34

GLOSSARY

THSDVT

Terminology	Meaning
Access control	*security technique that regulates who or what can view or use resources in a computing environment*
Access modifiers	*Private , Protected and Public*
Alternative	*One of two choices*
append()	*Used to add an element in a list*
Argument	*Argument is the actual value of this variable that gets passed to function.*
argv	*An array containing the values passed through command line argument*
Attribute	*Data items that makes up an object*
Authorization	*Giving permission or access*
Block	*Set of Statements*
Boolean	*means Logical*
break	*Exit the control*
c = sqlite3.connect('test. db')	*create a database connection to the SQLite database 'test.db'. You can also supply the special name :memory: to create a database in RAM.*
c.close()	*To release the connection of the database*
c.commit()	*To save the changes made in the table*
c.execute()	*Executes all SQL commands .Accepts two kinds of placeholders: question marks ? ("qmark style") and named placeholders :name ("named style").*
Cartesian product	*Cartesian operation is helpful to merge columns from two relations*
cd	*cd command refers to change directory*
Class	*Template of creating objects.*
Class variable	*An ordinary variable declared inside a class*
cls	*To clear the screen in command window*
Comma(,)	*Comma is used to separate each data in a csv file*

XII Std - CS EM Glossary.indd 335 1/19/2020 10:49:50 AM

compiler	*Scans the entire program and translates it as a whole into machine code. It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.*
Conjunction	*Concurrence, coincidence*
Constraint	*Restriction or limitation*
Constructor	*A special function get execution automatically when an object enter into scope.*
continue	*To skip the remaining part and start with next iteration.*
CRUD	*Create, Read, Update and Delete*
csv.reader()	*The reader function is designed to take each line of the file and make a list of all columns*
csv.register_dialect()	*A dialect describes the format of the csv file that is to be read*
CsvQuote All	*If quoting is set to csvquote all, then writerow() will quote all fields.*
cur = c.cursor()	*Creating cursor object*
cur.fetchall()	*method to get a list of the matching rows.*
cur.fetchmany()	*method that returns the next number of rows (n) of the result set*
cur.fetchone()	*method to retrieve a single matching row*
CWI	*Centrum Wiskunde & Informatica*
DBA	*DataBase Adminstrator*
DBMS	*Database Management System*
def	*This keyword is used to define function.*
Destructor	*A special function get execution automatically when an object exit from its scope.*
dict()	*It is used to print the data in dictionary format without orderdict*
Dictionary	*Collection of Key-Value pairs*
DictReader()	*Works by reading in the first line of the CSV and using each column comma separated value in this line as a dictionary key.*
Dictwriter()	*Write dictionary data into a CSV file*
elif	*else…if*
Embedded	*Firmly attached*
Enter key	*Enter key or newline is used to create rows in a csv file*

XII Std - CS EM Glossary.indd 336 1/19/2020 10:49:50 AM

eval()	*This function is used to evaluate the value of a string.*
Father of Relational Database	*Dr. Edgar Frank Codd*
g++	*compiler to compile c++ program*
GIS	*Geographic Information System*
global Scope	*A variable, with global scope can be used anywhere in the program.*
Glue language	*You do not write the complete application in the language, but rather, you use the language to orchestrate(organize) modules written in (possibly many different) other languages, making them work together to form the application. A glue language makes it easy to do that (convenient syntax, good support for inter-process communication and data managing, no compilation step etc).*
id ()	*It returns the memory address of the given object.*
IDLE	*Integrated Development Environment*
immutable	*unchangeable*
Implementation	*Implementation carries out the operation declared in the interfac*
import	*Import in python is similar to #include header_file in C++. Python modules can get access to code from another module by importing the file/function using "import" statement.*
Impure Functions	*Any function that changes the internal state of one of its arguments or the value of some external variable is an impure function.*
Instantiation	*Process of creating an object*
Integrity	*Whole and undivided*
Interactive Mode	*A way of using the Python interpreter by typing command and expressions at the prompt.*
Interface	*Interface defines what an object can do, but doesn't actually do it*
interpreter	*Translates program one statement at a time. It continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.*
Intersection	*Intersection defines a relation consisting of a set of all tuple that are in both A and B.*
Key	*Data that is mapped to a value in a dictionary*

XII Std - CS EM Glossary.indd 337 1/19/2020 10:49:50 AM

lambda	*Lambda function is mostly used for creating small and one- time anonymous function.*
LEGB rule	*Local → Enclosed → Global → Built-in scope*
List	*Mutable ordered collection of values*
local Scope	*A variable declared inside the function's body or in a block is called local scope.*
Looping	*Repetition*
Mapping	*The process of binding a variable name with an object*
Method	*A function declared and defined inside a class.*
module	*A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended. Within a module, the module's name (as a string) is available as the value of the global variable name .*
Namespaces	*containers for mapping names of variables to objects*
Nested Block	*A block within a block is called nested block.*
next()	*The next() function returns the next item from the iterator. It can also be used to skip a row of the csv file*
Object	*Collection of Data and Functions.*
Object Oriented Programming	*Computer Programming concept based on real world objects.*
operator.itemgetter(col_ no)	*To sort by more than one column from a csv file*
os.system()	*Used to execute system command and here in our python program is used to compile the c++ program using g++*
Parameter	*Parameter is variable in the declaration of function definition.*
parse	*To split an input into pieces of data that can be easily stored or manipulated.*
pass	*Can be used as placeholder in functions and loops.*
Projection (π)	*The projection eliminates all attributes of the input relation but those mentioned in the projection list*
Prompt	*Character (<<<) displayed by the interpreter to indicate that it is ready to take input from the user.*
Pure Functions	*Pure functions always returns the same result if the same arguments are passed in*
Python prompt	*>>>*

XII Std - CS EM Glossary.indd 338 1/19/2020 10:49:50 AM

Queue	*Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data(enqueue) and the other is used to remove data(dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first.*
RDBMS	*Relational Database Management System*
recursion	*When a function calls itself is known as recursion.*
Redundant	*Duplication of data*
Routines	*routines are otherwise called as functions or methods. In Python it is also called as definition*
Schema	*Structure or model*
Scope	*Visibility of variables, parameters and functions in one part of a program to another part of the same program.*
Script	*A Python program stored in a file.*
Script Mode	*A way of using the Python interpreter by typing command and expressions at the prompt.*
Select (σ)	*The SELECT operation is used for selecting a subset of the tuples according to a given selection condition*
Selectors	*Functions that retrieve information from the data type.*
Sequential	*One after another*
Set difference(-)	*(-) symbol denotes it. The result of A-B is a relation which includes all tuples that are in A but not in B.*
skipinitialspace=true	*When true, whitespace immediately following the delimeter is ignored. The default is false*
slicing	*cut*
Stack	*A stack (sometimes called a "push-down stack") is an ordered collection of items where the addition of new items and the removal of existing items always takes place at the same end. This end is commonly referred to as the "top." The end opposite the top is known as the "base.This ordering principle is sometimes called LIFO, last- in first-out.*
stride	*a long step*
string	*sequence of letters, numbers or symbols*
subscript	*an index number*

XII Std - CS EM Glossary.indd 339 1/19/2020 10:49:50 AM

Syntax	*The structure of a program*
Syntax Error	*An error in a program that makes it impossible to parse.*
Token	*One of the basic elements of the syntactic structure of a program.*
Tuple	*It is a sequence of immutable(not changeable) objects. Tuples are sequences, just like lists.Tuples are defined by having values between parentheses ( ).*
Union operation(U)	*Union is symbolized by symbol. It includes all tuples that are in tables A or in B.*
variable	*Memory box to store values*
writer ow()	*Method to write a single row of data in a file*
writer ows()	*Method to write multiple rows of data in a file*
FALSE	*Logical value 0*
TRUE	*Logical value 1*

XII Std - CS EM Glossary.indd 340 1/19/2020 10:49:50 AM

ANNEXURE - 1

List of Python Functions

I. Built-in Functions

Function	Description
abs()	returns absolute value of a number
all()	returns true when all elements in iterable is true
any()	Checks if any Element of an Iterable is True
ascii()	Returns String Containing Printable Representation
bin()	converts integer to binary string
bool()	Converts a Value to Boolean
bytearray()	returns array of given byte size
bytes()	returns immutable bytes object
callable()	Checks if the Object is Callable
chr()	Returns a Character (a string) from an Integer
classmethod()	returns class method for given function
compile()	Returns a Python code object
complex()	Creates a Complex Number
delattr()	Deletes Attribute From the Object
dir()	Tries to Return Attributes of Object
divmod()	Returns a Tuple of Quotient and Remainder
enumerate()	Returns an Enumerate Object
eval()	Runs Python Code Within Program
exec()	Executes Dynamically Created Program
filter()	constructs iterator from elements which are true
float()	returns floating point number from number, string
format()	returns formatted representation of a value
getattr()	returns value of named attribute of an object
globals()	returns dictionary of current global symbol table
hasattr()	returns whether object has named attribute
hash()	returns hash value of an object
help()	Invokes the built-in Help System
hex()	Converts to Integer to Hexadecimal
id()	Returns Identify of an Object
isinstance()	Checks if a Object is an Instance of Class
issubclass()	Checks if a Object is Subclass of a Class
iter()	returns iterator for an object
len()	Returns Length of an Object
locals()	Returns dictionary of a current local symbol table
map()	Applies Function and Returns a List
max()	returns largest element
memoryview()	returns memory view of an argument
min()	returns smallest element
next()	Retrieves Next Element from Iterator
object()	Creates a Featureless Object
oct()	converts integer to octal

Annexure I.indd 341 24-01-2020 11:21:44

open()	Returns a File object
ord()	returns Unicode code point for Unicode character
pow()	returns x to the power of y
print()	Prints the Given Object
property()	returns a property attribute
range()	return sequence of integers between start and stop
repr()	returns printable representation of an object
reversed()	returns reversed iterator of a sequence
round()	rounds a floating point number to ndigits places.
set()	returns a Python set
setattr()	sets value of an attribute of object
slice()	creates a slice object specified by range()
sorted()	returns sorted list from a given iterable
staticmethod()	creates static method from a function
str()	returns informal representation of an object
sum()	Add items of an Iterable
super()	Allow you to Refer Parent Class by super
type()	Returns Type of an Object
vars()	Returns __dict__ attribute of a class
__import__()	Advanced Function Called by import

II. String Functions

Function	Description
capitalize()	Converts first character to Capital Letter
casefold()	converts to casefolded strings
center()	Pads string with specified character
count()	returns occurrences of substring in string
encode()	returns encoded string of given string
endswith()	Checks if String Ends with the Specified Suffix
expandtabs()	Replaces Tab character With Spaces
find()	Returns the index of first occurrence of substring
format()	formats string into nicer output
format_map()	Formats the String Using Dictionary
index()	Returns Index of Substring
input()	reads and returns a line of string
int()	returns integer from a number or string
isalnum()	Checks Alphanumeric Character
isalpha()	Checks if All Characters are Alphabets
isdecimal()	Checks Decimal Characters
isdigit()	Checks Digit Characters
isidentifier()	Checks for Valid Identifier
islower()	Checks if all Alphabets in a String are Lowercase
isnumeric()	Checks Numeric Characters
isprintable()	Checks Printable Character
isspace()	Checks Whitespace Characters

Annexure I.indd 342 24-01-2020 11:21:44

istitle()	Checks for Titlecased String
isupper()	returns if all characters are uppercase characters
join()	Returns a Concatenated String
ljust()	returns left-justified string of given width
lower()	returns lowercased string
lstrip()	Removes Leading Characters
maketrans()	returns a translation table
partition()	Returns a Tuple
replace()	Replaces Substring Inside
rfind()	Returns the Highest Index of Substring
rindex()	Returns Highest Index of Substring
rjust()	returns right-justified string of given width
rpartition()	Returns a Tuple
rsplit()	Splits String From Right
rstrip()	Removes Trailing Characters
slice()	creates a slice object specified by range()
split()	Splits String from Left
splitlines()	Splits String at Line Boundaries
startswith()	Checks if String Starts with the Specified String
strip()	Removes Both Leading and Trailing Characters
swapcase()	swap uppercase characters to lowercase; vice versa
title()	Returns a Title Cased String
translate()	returns mapped charactered string
upper()	returns uppercased string
zfill()	Returns a Copy of The String Padded With Zeros

III. List Functions

Function	Description
append()	Add Single Element to The List
clear()	Removes all Items from the List
copy()	Returns Shallow Copy of a List
count()	returns occurrences of element in a list
extend()	Add Elements of a List to Another List
index()	returns smallest index of element in list
insert()	Inserts Element to The List
list() Function	creates list in Python
pop()	Removes Element at Given Index
remove()	Removes Element from the List
reverse()	Reverses a List
slice()	creates a slice object specified by range()
sort()	sorts elements of a list

Annexure I.indd 343 24-01-2020 11:21:44

IV. Tuple Functions

Function	Description
count()	returns occurrences of element in a tuple
index()	returns smallest index of element in tuple
slice()	creates a slice object specified by range()
tuple() Function	Creates a Tuple
zip()	Returns an Iterator of Tuples

V. Set Functions

Function	Description
add()	adds element to a set
clear()	remove all elements from a set
copy()	Returns Shallow Copy of a Set
difference()	Returns Difference of Two Sets
difference_update()	Updates Calling Set With Intersection of Sets
discard()	Removes an Element from The Set
frozenset()	returns immutable frozenset object
intersection()	Returns Intersection of Two or More Sets
intersection_update()	Updates Calling Set With Intersection of Sets
isdisjoint()	Checks Disjoint Sets
issubset()	Checks if a Set is Subset of Another Set
issuperset()	Checks if a Set is Superset of Another Set
pop()	Removes an Arbitrary Element
remove()	Removes Element from the Set
set()	returns a Python set
symmetric_difference()	Returns Symmetric Difference
symmetric_difference_update()	Updates Set With Symmetric Difference
union()	Returns Union of Sets
update()	Add Elements to The Set.

VI. Dictionary Functions

Function	Description
clear()	Removes all Items
copy()	Returns Shallow Copy of a Dictionary
dict()	Creates a Dictionary
fromkeys()	creates dictionary from given sequence
get()	Returns Value of The Key
items()	returns view of dictionary's (key, value) pair
keys()	Returns View Object of All Keys
pop()	removes and returns element having given key
popitem()	Returns & Removes Element From Dictionary
setdefault()	Inserts Key With a Value if Key is not Present
update()	Updates the Dictionary
values()	returns view of all values in dictionary

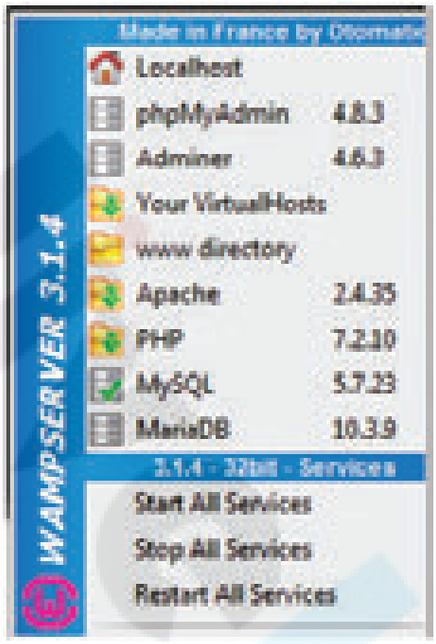
Annexure I.indd 344 24-01-2020 11:21:44

ANNEXURE - 2

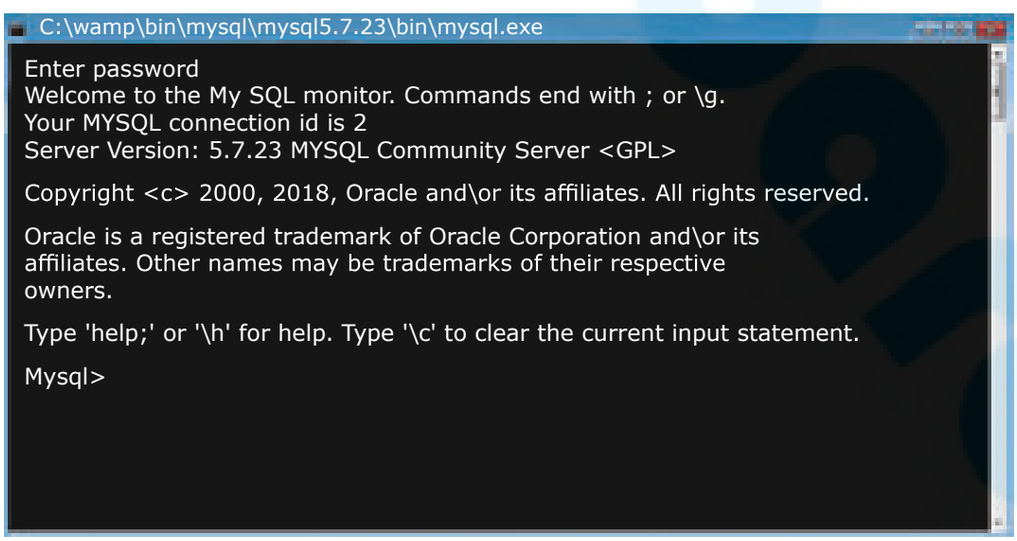
UCK7WT

Installation of MySQL and Creating a database

- Download WAMP based on system configuration and install it in your computer.



- log on to MySQL console as user (root).
- Enter password appears (No need to enter password), press Enter key.
- MySQL prompt appears from where you can enter the SQL commands.



Annexure II.indd 345 24-01-2020 10:53:08

ANNEXURE-3

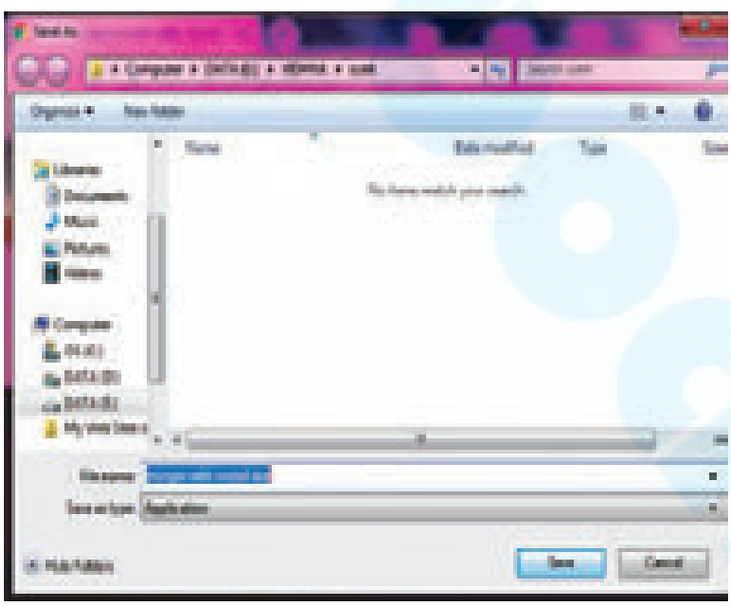
Installation of MinGW

US42Y5

STEPS TO INSTALL MinGW-w64 - for 32 and 64 bit Windows

STEP1: Type https://sourceforge.net/projects/mingw-w64 in any search engine(www.google.in) and double click it.

Step 2 : Click the Download button which appears in the home page. The file should start downloading in your standard download folder or it prompts the save as dialog box



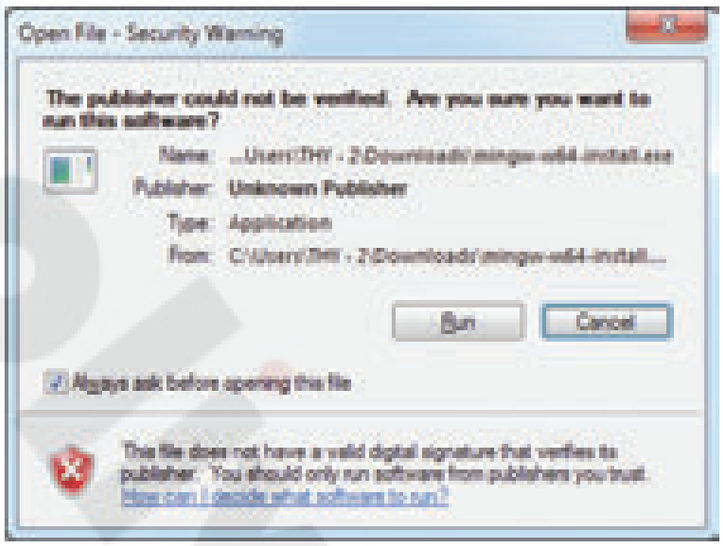
Step 3 : The file should appear as mingw-get-setup.exe . Terminate the window browsing the SourceForge web site. Move this file to a more permanent location, so that you can install MinGW (and reinstall it later, if necessary).

Annexure III.indd 346 24-01-2020 10:55:37

step 4: Start the Installation as per the instructions given bellow.

Installing:

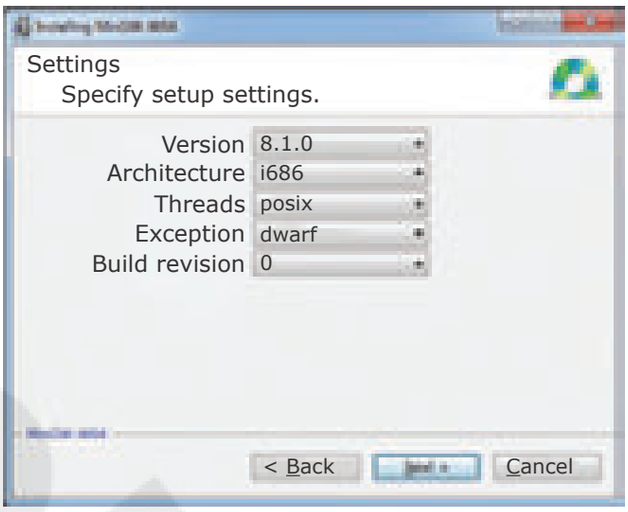
1. Double-click the mingw-get-setup.exe . icon. The following pop-up window will appear.



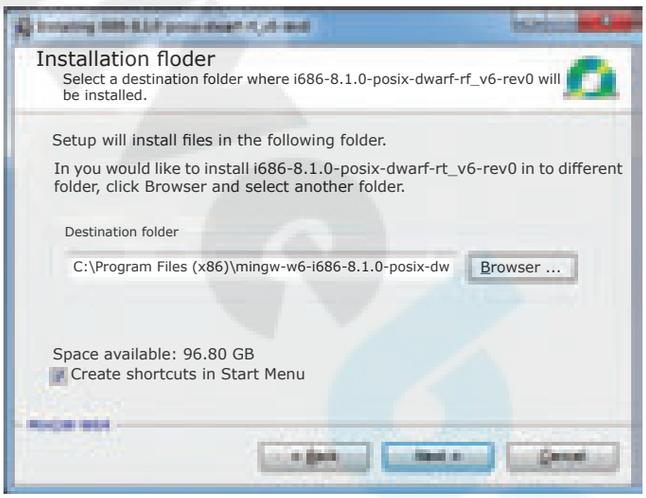
2. Click Run. The following pop-up window will appear. Click next button in the following window

3. The following pop-up window will appear, which specify the Setup settings

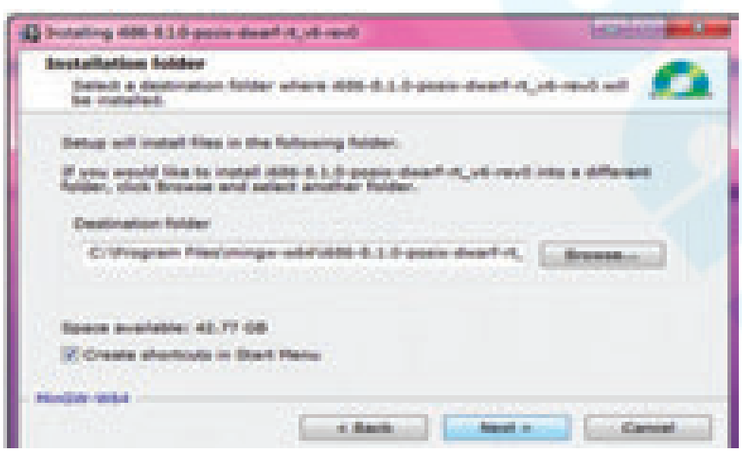
Annexure III.indd 347 24-01-2020 10:55:38



4. Select the destination folder in the following window to install MinGW-W64. Click next button

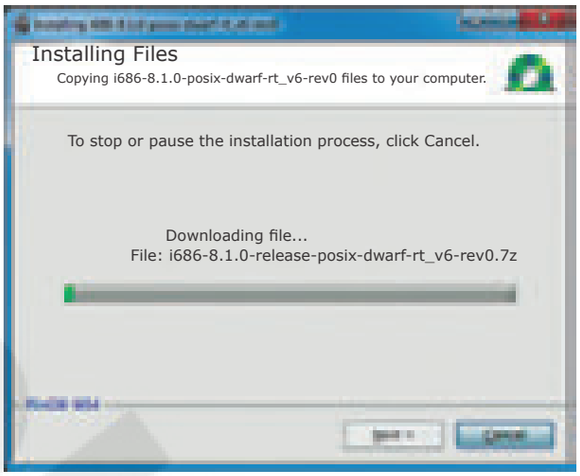


5. Select the destination folder in the following window to install MinGW-W64.

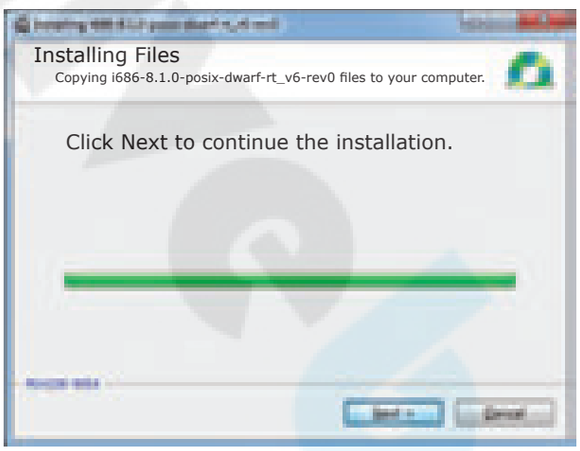


Annexure III.indd 348 24-01-2020 10:55:38

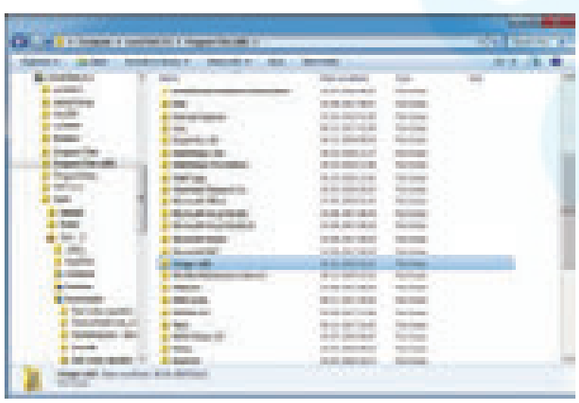
6. Wait for the completion of installation



7 Click the button once it gets highlighted.

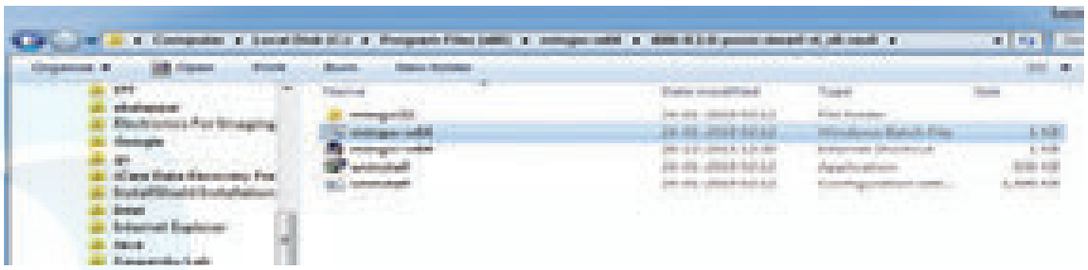


8. Locate the folder in your PC. For example here the mingw64 is present in the following path ***c:\Program Files\mingw-w64.***

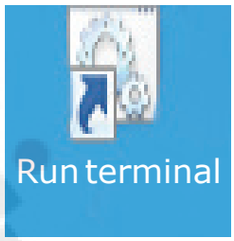


9. Open the folder and double click the batch file. The program will get executed.

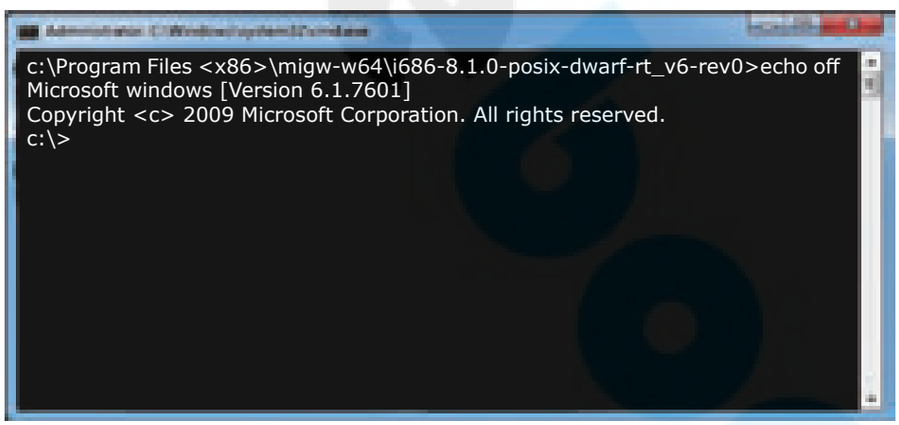
Annexure III.indd 349 24-01-2020 10:55:38



10 A shortcut for run will be created in the desktop. Double click and open the "command window".



11 Through this command **window only we have to execute the Python program** because it contains the other programming language program ie is C++. This command window dynamically invoke the g++ compiler to compile and execute C++ program using Python program



Annexure III.indd 350 24-01-2020 10:55:38

ANNEXURE-4

Installation of pip

W2ER1G

First of all you need to identify whether pip is installed in your PC. If so, upgrade the pip in your system. To do this, you need to launch the command prompt. Before trying to install or upgrade the pip, the command will work only if you have appended the path of python directory in the path variable of the windows.

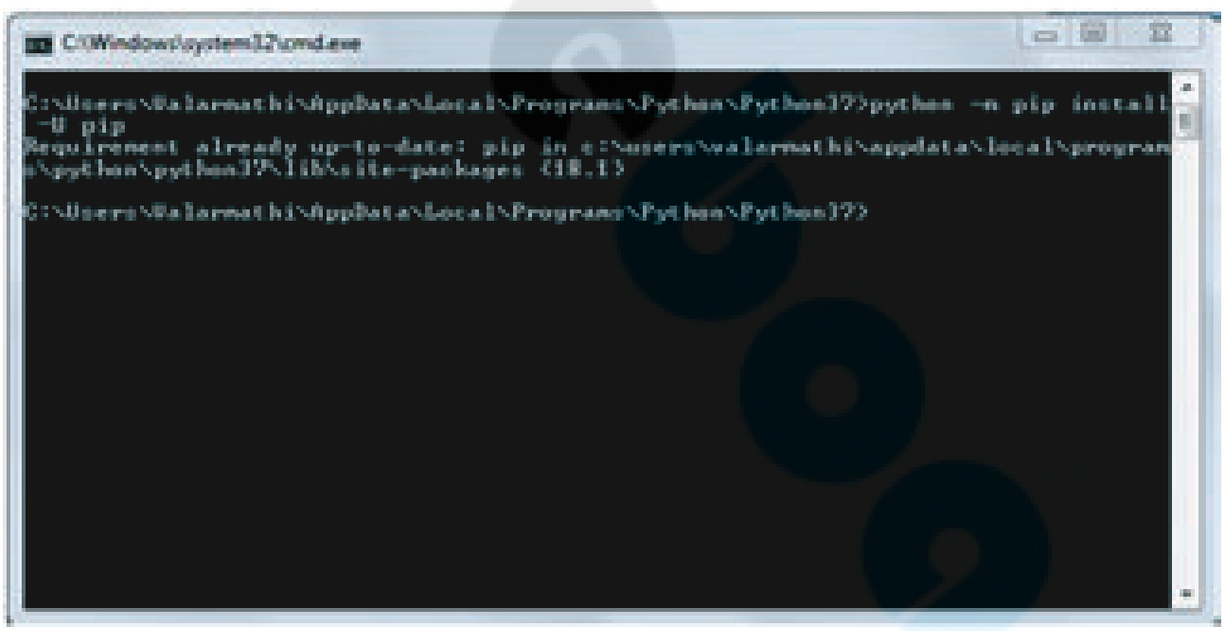
Check if pip is Installed

To check if pip is already installed in your system, navigate your command line to the location of Python's script directory.

You can install the latest version of pip from your command prompt using the following command:

Python –m pip install –U pip

-U represents upgrading pip to the latest version.



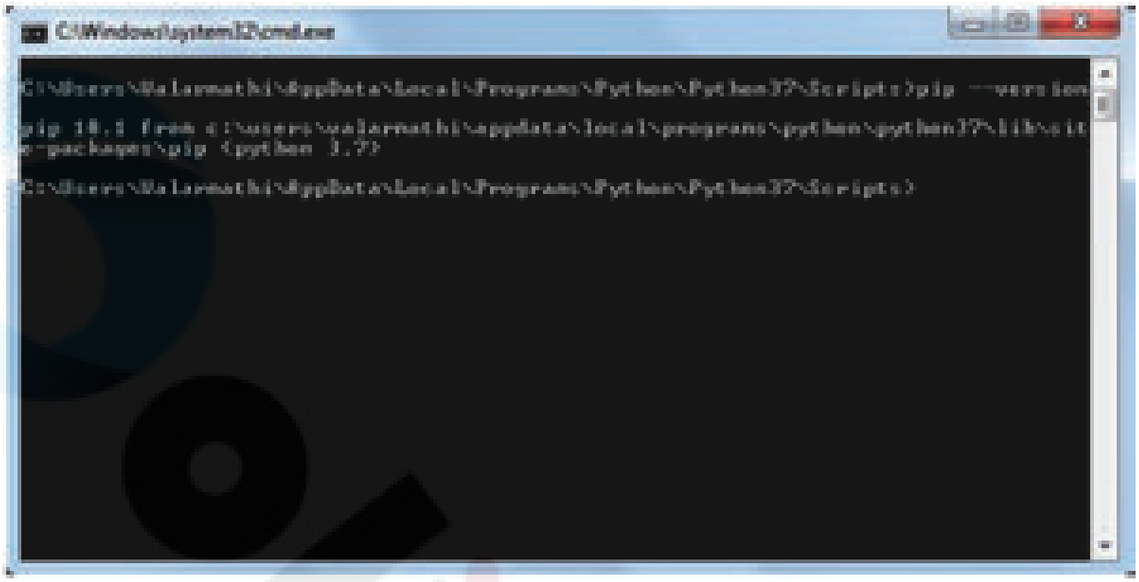
To Check pip version:

To check the version of pip in your system, type the following command:

C:\Users\Your Name\AppData\Local\Programs\Python\Python36-32\Scripts>pip --version

The output in command prompt will look like this:

Annexure IV.indd 351 24-01-2020 10:56:46

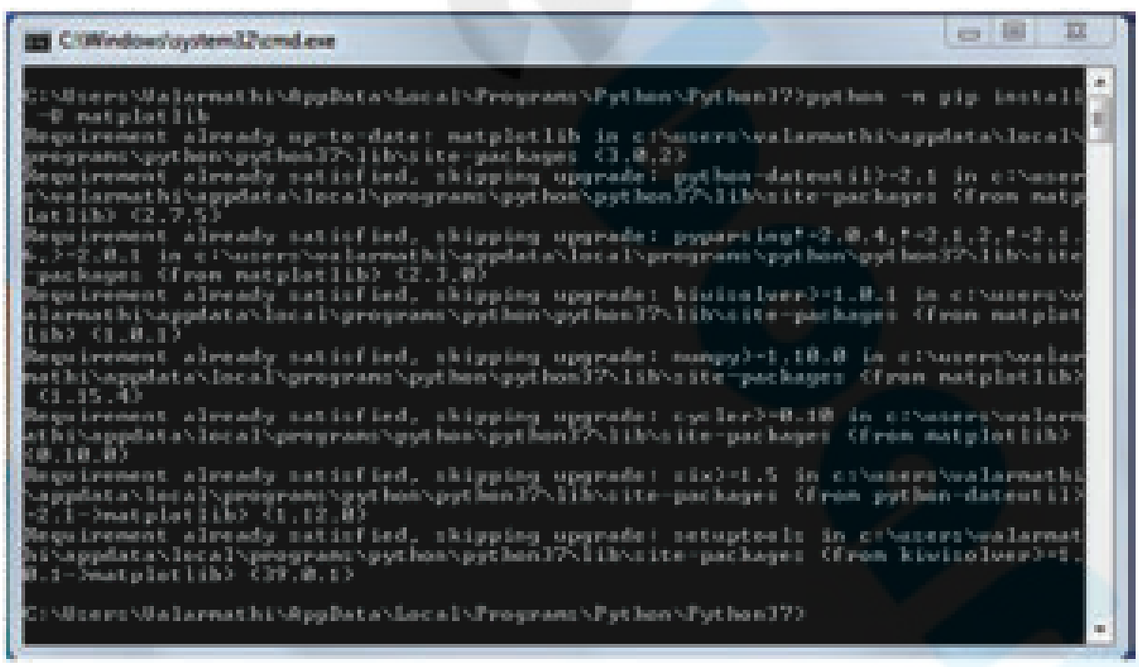


You can see the version of pip installed from the output screen.

To install matplotlib, type the following in your command prompt:

Python –m pip install –U matplotlib

This command will download matplotlib from the source library. If it is already installed the screen will look like the following:



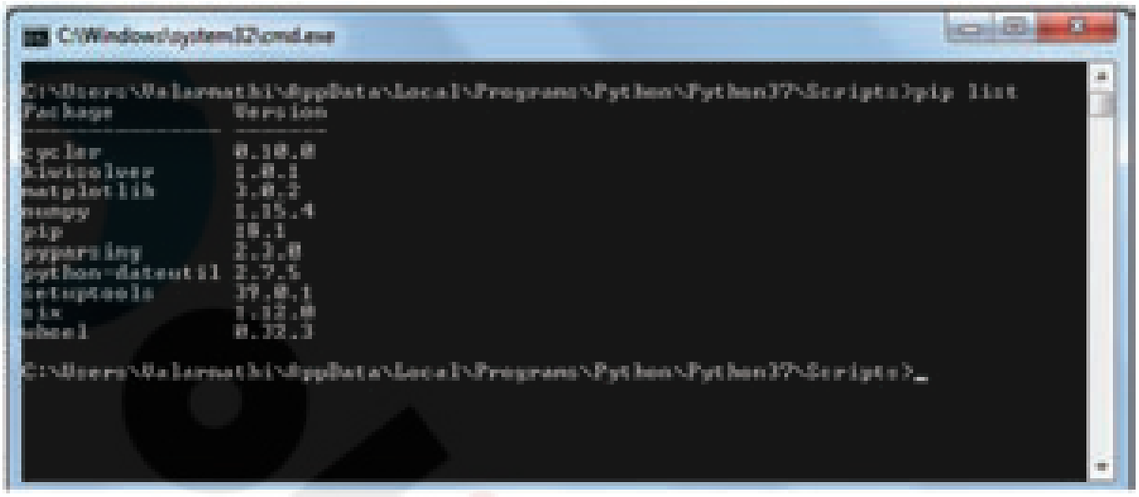
List Packages

To view the list of installed packages on your system, use the List command:

C:\Users\YourName\AppData\Local\Programs\Python\Python36-32\Scripts>pip list The

Annexure IV.indd 352 24-01-2020 10:56:46

screen will display the list of all the packages installed on your system.



Annexure IV.indd 353 24-01-2020 10:56:46

COMPUTER SCIENCE – XII
List of Authors and Reviewers

Domain Expert

Dr. T.V.Gopal
Professor, Dept of Computer Science and Technology,
College of Engineering, Guindy, Anna University, Chennai.

Reviewers

Dr. Ranjani Parthasarathi
Professor, Dept of Info Sci and Tech, College of Engineering, Guindy,
Anna University, Chennai

Content Expert

Mr. Malaiarasu P
Associate Professor, Dept of Computer Science,
Govt. Arts College, Perumpakkam, Chennai

Dr. Ramesh Kumar M
Assistent Professor and Head of the Department,
Dept of Computer Science,
Govt. Arts College, Nandhanam, Chennai

Dr. Radha P
Assistant Professor, Dept of Information Technology,
Govt. Arts & Science College (A), Coimbatore

Dr. Nester Jeyakumar M
Associate Professor and Head of the Department,
Dept of Computer Science, Loyola College, Chennai

Experts Co-ordinator

Mr. Ravikumar Arumugam
Principal, District Institute of Education and Training,
Mayannur, Karur Dt.

Academic Coordinators

Mrs. Tamil Selvi R
B.T. Assistant,
Government High School, Poonampalayam, Trichy District

Authors

Mr. Kannan K
Post Graduate Teacher, Chennai Girls Hr Sec School,
Rotler street , Chennai

Mr. Ramakrishnan V G
Post Graduate Teacher, Karnataka Sangha Hr Sec School,
T Nagar, Chennai

Mrs. Dr. Vidhya H
Post Graduate Teacher,
DAV Boys Senior Secondary School,
Gopalapuram, Chennai.

Mr. Sreenivasan R
Post Graduate Teacher, Santhome Hr Sec School, Mylapore, Chennai

Mr. Gowrisankar N.V
Post Graduate Teacher, Chennai Girls Hr Sec School,
Nungambakkam, Chennai

Mr. Lenin K
Post Graduate Teacher, Chennai Girls Hr Sec School, Saidapet, Chennai

Mrs. Bindhu Mohandas
Post Graduate Teacher, Vijayanta Model Hr Sec School,
H.V.F Estate , Avadi, Chennai

Mrs. Gajalakshmi R
Post Graduate Teacher, Jaigopal Garodia Hindu Vidyalaya Hr Sec School,
West Mambalam, Chennai

Dr. Valarmathi K E
Post Graduate Teacher, Velammal Vidhyashram, Surapet, Chennai

QR Code Management Team

R. Jaganathan,
SGT, PUMS - Ganesapuram,
Polur , Thiruvannamalai.

J.F. Paul Edwin Roy,
B.T.Asst. PUMS -Rakkipatty, Salem.

M. Saravanan,
B.T.Asst, G.G.H.S.S, Puthupalayam,
Vazhapadi, Salem.

Typist

Mrs. Meena T
SCERT, Chennai.

Art and Design Team

Layout

THY designers and computers
Chennai

In-House

QC - Rajesh Thangapan
- Mathan Raj R
- Balasubramani. R
- Wrapper Design - Kathir Arumugam

Co-ordination

Ramesh Munisamy

This book has been printed on 80 G.S.M.
Elegant Maplitho paper.

Printed by offset at:

Annexure IV.indd 354 24-01-2020 10:56:46

FREE Ebooks

Download Now

Current Affairs

Explore Now

Free Notes + Mock Tests + Mindmaps + Current Affairs

For TNPSC Group II/IIA, IV & VAO Exams

Web

APP

BLOG

Your one-stop destination for all exam related information & preparation resources.

Explore Now

FORUM

Interact with peers & experts, exchange scores & improve your preparation.

Explore Now

www.OliveBoard.in