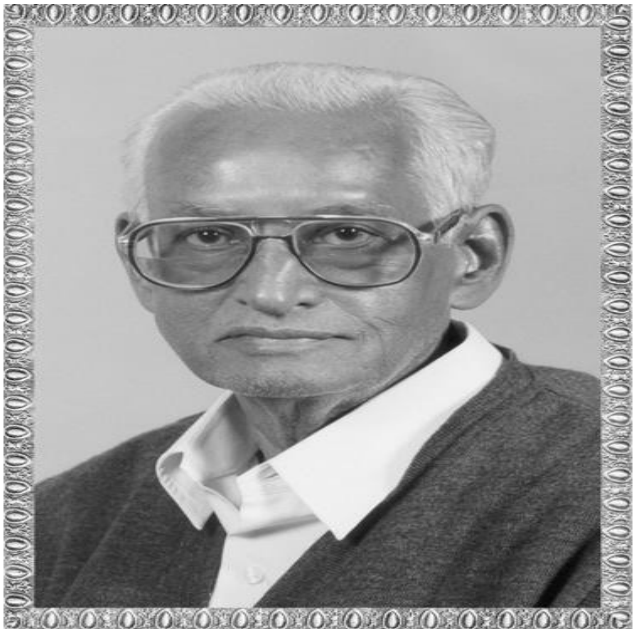
SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.
(AFFILIATED TO SAURASHTRA UNIVERSITY)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645



Lt. Shree Chimanbhai Shukla

B.C.A. – SEMESTER -6

SUB :- CS – 33 :- PROGRAMMING IN PYTHON

Shree H.N.Shukla College
Street No. 2, Vaishali Nagar,
Nr. Amrapali Under Bridge,
Raiya Road, Rajkot.
Ph. (0281)2440478

Shree H.N.Shukla College
Street No. 3, Vaishali Nagar,
Nr. Amrapali Under Bridge,
Raiya Road, Rajkot.
Ph. (0281)2471645

SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.

(AFFILIATED TO SAURASHTRA UNIVERSITY)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

CS – 33 : Programming in Python –B.C.A. –SEM -6

Sr.No.	Topic	Detail
1	**Introduction to Python**	• Basic Element of Python • Branching Programs • String and Input • Iteration • Function and Scoping • Specifications • Recursion • Global Variables • Modules • Files • Tuples • List & Mutability • Functions as Object • Strings • Dictionaries

2	**OOP using Python**	□ Handling Exception □ Exception as Control Flow □ Assertion □ Abstract Data Type □ Class □ Inheritance □ Encapsulation □ Information hiding □ Search Algorithm □ Sorting Algorithm □ Hash table

3	**Plotting using PyLab**	• Plotting using PyLab • Plotting Mortgages • Extended Example • Fibonacci Sequence Revisited • Dynamic Programming • 0/1 Knapsack Algorithm • Dynamic Programming with Divide and Conquer
4	**Network Programming and GUI usingPython**	• Network Programming: • Protocol, Sockets, • Knowing IP Address, • URL, Reading the Source Code of a Web Page, • Downloading a Web Page from Internet, • Downloading an Image from Internet, • A TCP/IP Server, A TCP/IP Client, • A UDP Server, A UDP Client, • File Server, File Client, • Two-Way Communication between Server and Client, • Sending a Simple Mail. • GUI Programming: • Event-driven programming paradigm; • creating simple GUI; • buttons, labels, entry fields, dialogs; • o widget attributes - sizes, fonts, colors ,layouts, nested frames

5	**Connecting with Database**	• Verifying the MySQL dB Interface Installation, • Working with MySQL Database, • Using MySQL from Python, • Retrieving All Rows from a Table, • Inserting Rows into a Table, • Deleting Rows from a Table, • Updating Rows in a Table, • Creating Database Tables through Python

SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.
(AFFILIATED TO SAURASHTRA UNIVERSITY)

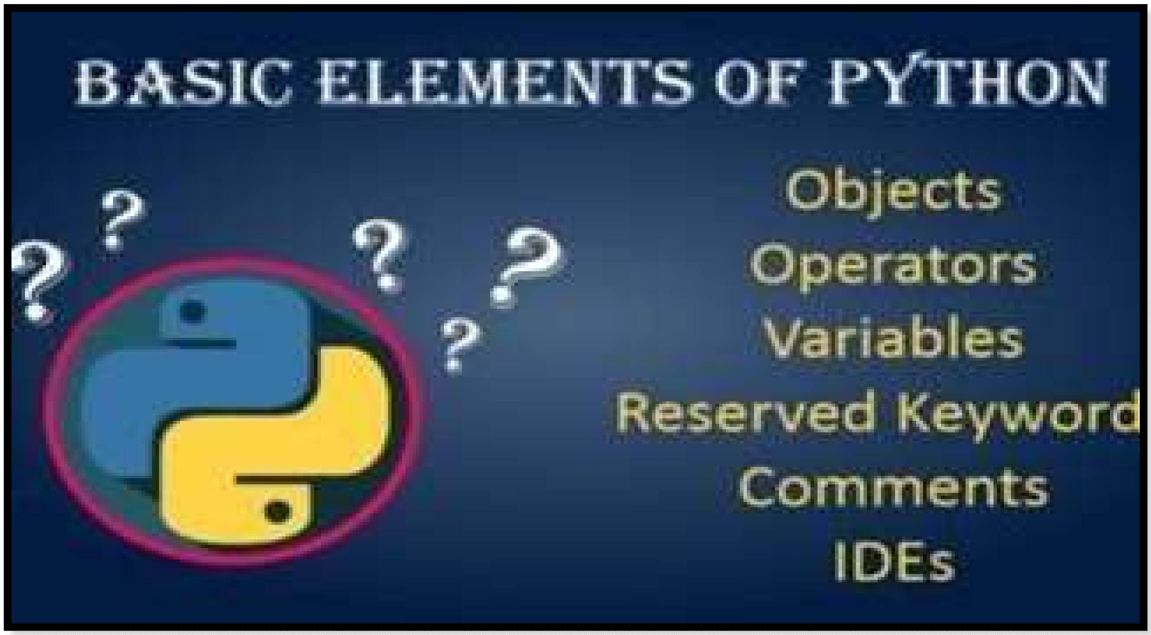
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

CHAPTER-1
Introduction to Python

- Basic Element of Python
- Branching Programs
- String and Input
- Iteration
- Function
- Scoping
- Specifications
- Recursion
- Global Variables
- Modules
- Files
- Tuples
- List & Mutability
- Functions as Object
- Strings
- Dictionaries

Q-1 What is Python ? Explain Basic Elements of Python :



BASIC ELEMENTS OF PYTHON

Objects
Operators
Variables
Reserved Keyword
Comments
IDEs

Detail :-

- Python is popular programming language.
- It was created in 1991 by **Guido Van Rossum**.
- It is used for :
  - Web Development (server side)
  - Software Development
  - System Scripting
  - Mathematics
- Python support following elements to perform perfect programming:

1. **Data Type :**
   - ✓ Python support integer and float data type to hold numbers.
   - ✓ Python interpreter can produce the result of numeric values.

2. **Variable :**

   ✓ Unlike other programming language , python have no command for declaring variable.

   ✓ You can create variable by assigning value directly to it.

   o **EX :- x = 50**
   Name = “hello”

3. **Syntax :**

   ✓ Python syntax can be executed by writing directly at the command line like,

   ✓ >>> **print (“hello”)**

4. **String :-**

   ✓ String is a collection of different characters.

   ✓ You can write string in signle quotes(‘ ‘) as well as in doublequotes (“ “).

   o **EX :- a=**
   ‘hello’b=” “

5. **Tuples :-**

   ✓ In the case of tuples , it is collection of different elements and values supported by python data types.

   ✓ Tuples are enclosed in round brackets ().

   o **EX :- a = (‘abc’ , ‘jkl’ , ‘xyz’,18)**

   o

6. **List :-**

   ✓ In the case of , It is collection of element or values supported by python data types.

   ✓ List are enclosed in square brackets ().

   o **EX :- a = [1,2,3,4,5]**

7. **Dictionary :-**

✓ A Dictionary is a collection which is unordered , changeable and indexed.

✓ Dictionary can be enclosed in curly brackets { } with key & valuepair.

- **EX :- dict = {name = “snehal” , surname = “pandya”}**

8. Operators :-

✓ An operators are used to perform operations on variable or value.

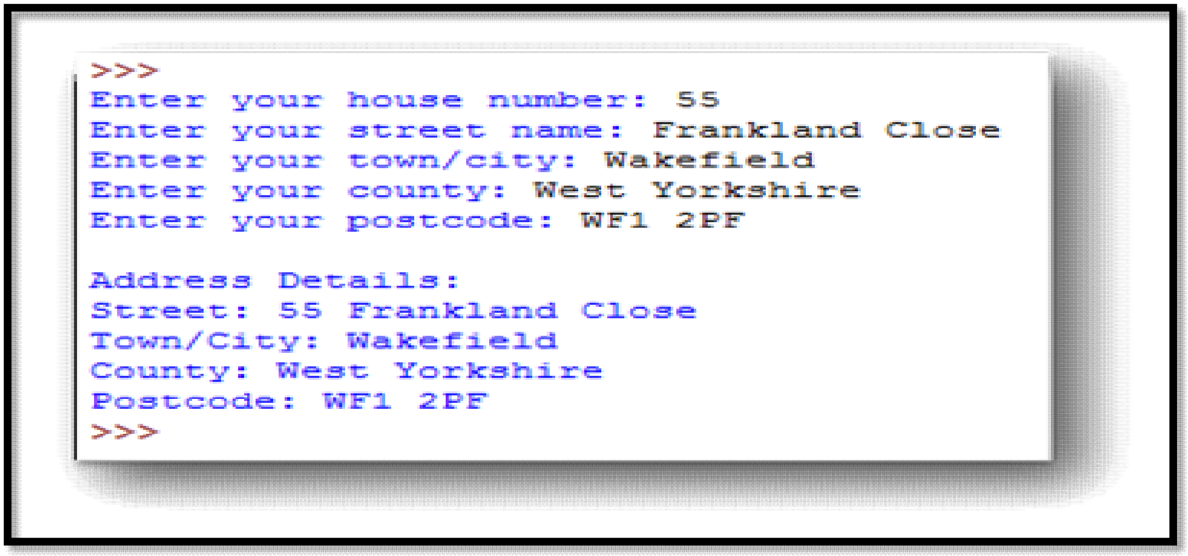
✓ Python support following operators :

- Arithmatic
- Assignment
- Comparison
- Logical
- Membership
- Bitwise
- Identity

1 Word Question – Answer

SR. NO.	QUESTION	ANSWER
1	Python was developed by_____	Guido Van Rossum
2	Python is ___________ language.	Open source

Q-2How to input and output value in the python:



```
>>>
Enter your house number: 55
Enter your street name: Frankland Close
Enter your town/city: Wakefield
Enter your county: West Yorkshire
Enter your postcode: WF1 2PF

Address Details:
Street: 55 Frankland Close
Town/City: Wakefield
County: West Yorkshire
Postcode: WF1 2PF
>>>
```

Detail :-

- In python programming user can input the data as well as get the output asa result of data.

How to input value :

- ✓ To input value python provide one of the built – in function named input().
- ✓ You can ask user to input particular value from userside.
- ✓ You can call input () and wait for user to enter the data.
- ✓ In python2 you can use row_input() to get value from userside , while in python3 you have input() to get value from user.
  - **EX :- num = input (“enter number”)**
    name = input(“enter name”)

How to ouput / print value :

- ✓ Python provide print() to get output to the file.

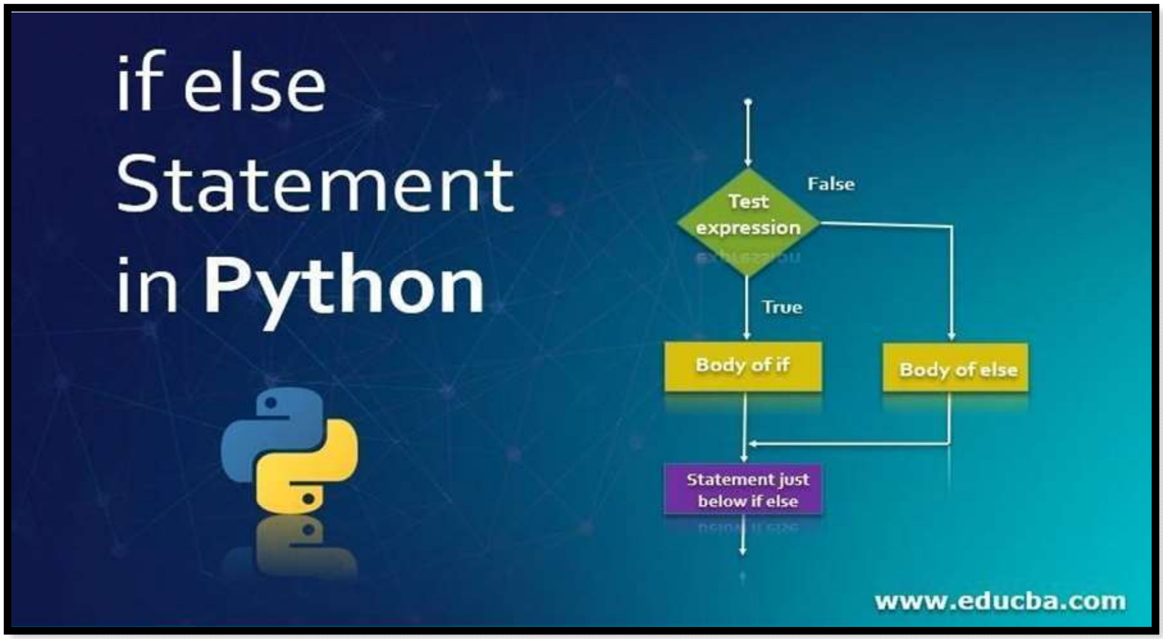
- ✓ Print() support message of the user as well as value of variable they want to print.
  - ○ **EX :- a=5**
    >>> print (a)
    Output :- 5
  - ○ **EX :- b=10**

    >>> print (“value of b is =” ,b)
    Output :- value of b is = 10

1 Word Question – Answer

SR. NO.	QUESTION	ANSWER
1	To input value in the python ____________function can be used.	Input()
2	To print output on the screen_____________ function can be used.	Databse Access
3	Python prompt can be represented by __________	>>>

Q-3Expalin Braching Statements in python:



Detail :-

- If statement support generally the logical expression.
- In this statement first of all we have to give condition.
- If condition become true then statement following if will be execute otherwise condition will be terminated.

✓ **Syntax :-**

```
If  (<condition>):
<statement>
```

✓ **Example :-**

```
A=10
If(A>0):
        Print('yes')
```

If ….. Else :-

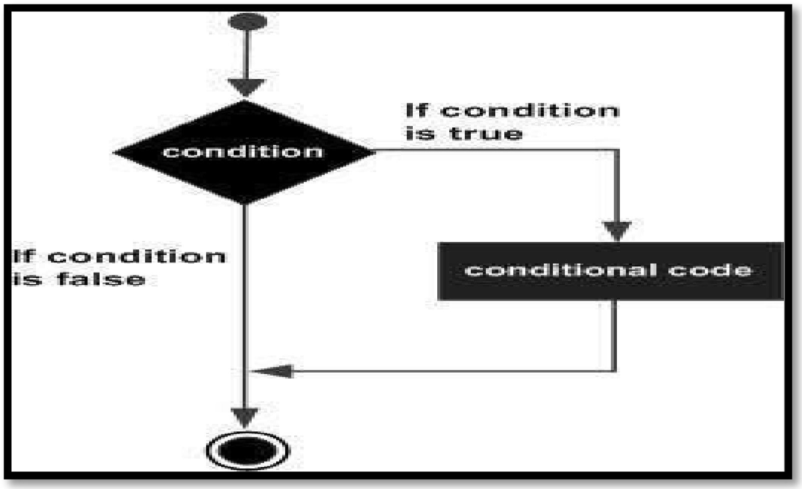
- In this statement first of all condition will be check.
- If condition become true then statement following if will be execute.
- But if condition become false then statement following else will be execute.

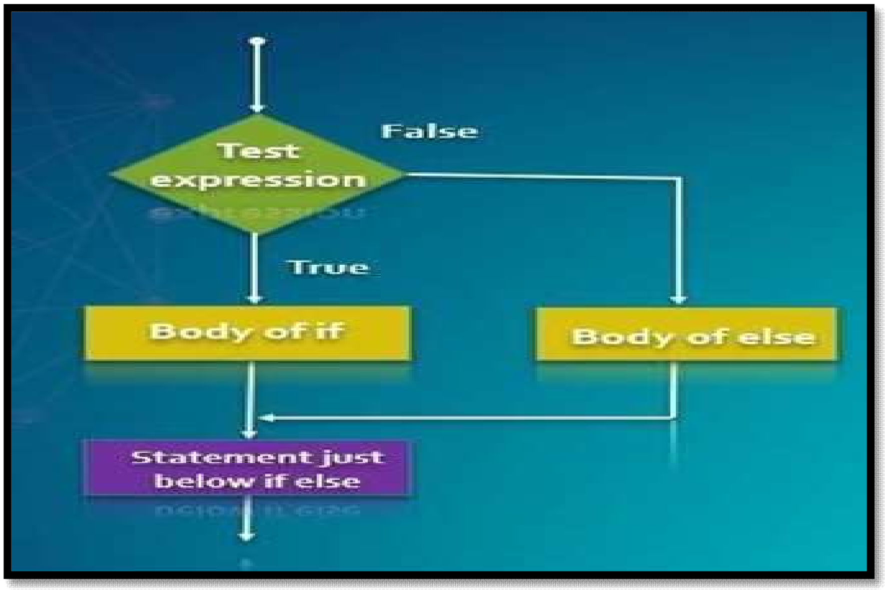
✓ **Syntax :-**

```
If (<condition>):
        <statement>
Else:
        <statement>
```

✓ **Example :-**

```
A=10
If(A>0):
        Print('yes')
Else:
        Print('no')
```





Elif :-

- The elif statement is used to support multiple conditions at the same time.
- At a time only one condition will become true.
- In this case if no any condition become true then finally the statement following else will be execute.

✓ **Syntax :-**

```
If (<condition>):
      <statement>
Elif(<condition>):
      <statement>
Elif(<condition>):
      <statement>
Else:
      <statement>
```

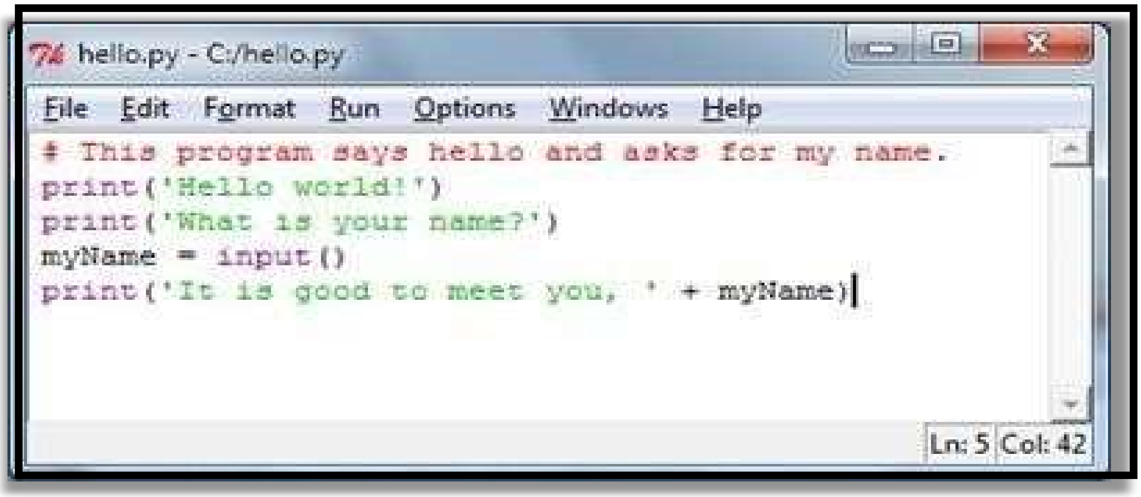
- ✓ **Example :-**

```
a=10,b=20,c=30

 if(a>b and a>c):
         print('a    is    max')
 elif(b>a and b>c):
         print('b is max')
 else:
         print('c is max')
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	Python provide one of the important branching statement that is_______.	If statement
2	In if statement the condition is followed by _____________.	Colon(:)
3	If condition become false then statement followed by___________ will be execute.	Else :
4	__________ statement can be used to support multiple conditions at the same time.	elif

Q-4 Explain String in Python.



```
# This program says hello and asks for my name.
print('Hello world!')
print('What is your name?')
myName = input()
print('It is good to meet you, ' + myName)
```

Detail :-

- Pythondoesnotsupportcharactertype.
- String is mostpopular datatype in python.
- Stringis acollectionof differentcharacterandsymbols.
- We cancreate or declare astring by enclosing “ “(Double Quotes) aswellas ‘ ‘Single Quotes forthestatements.
- Let’ssee, how to declare astringvalue in python:
  - **Example:-** **var1 =“hello”**

How to Access Characters from particular String :-

- ✓ We can accessdifferentcharactersfromgivenstring using indexing.
- ✓ Inpython stringindex willstart from 0(zero).
- ✓ We can notdelete or removethecharacters fromthestringbut deletingentire string is possible using **del** keyword.
  - **Example:- var1 =“helloworld”**
    Var2 =“ pythonprogramming”
    Print (“var1[0]:”, var1[0]) output:- h
    Print(“var2[1:5]”,var2[1:5]) output:- ytho

How to Concatenate (join) two strings :-

✓ To join or concatenatetwo strings in python , (+) plusoperatorcan be used.

- **Example:-** **var1 =“hello ”**
  Var2 =“ pythonprogramming”
  Print(var1 +var2) **output:- hellopythonprogramming**

How to update existing string :-

✓ You can "update" an existing string by (re)assigning a variable to another string.

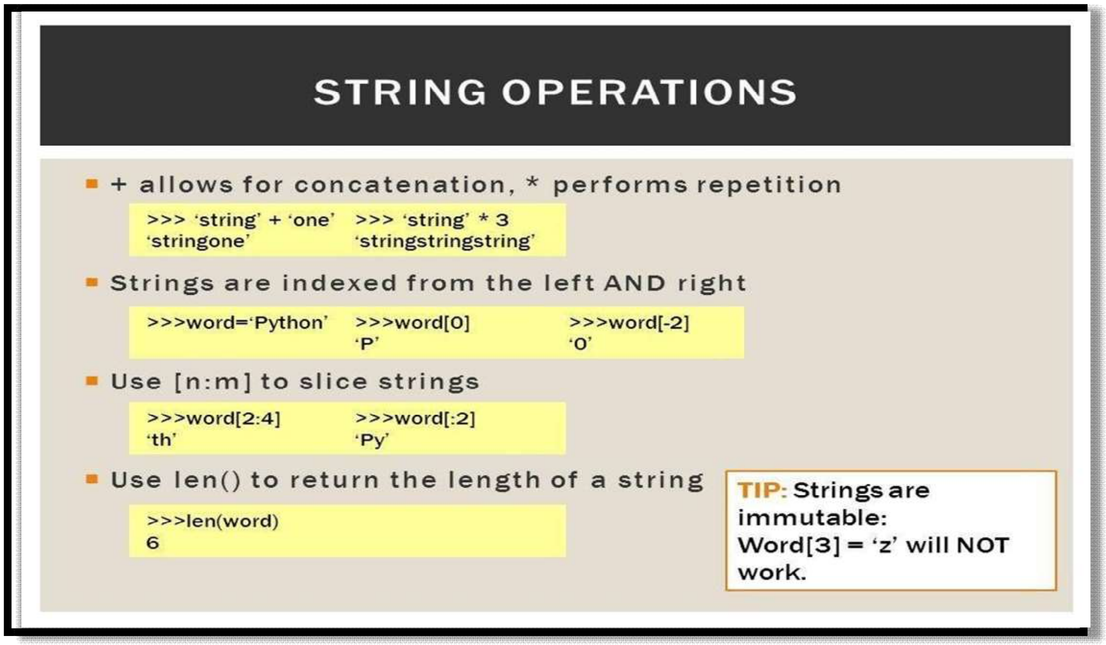
✓ The new value can be related to its previous value or to a completely different string altogether.

- **Example :-** **var1 = ‘hello world’**
  Print (‘updated string:-‘ , var1[:6] + ‘python’)
  output :hello python

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	In python ,String can be represented by	Single quotes & double quotes
2	To access substring ,we have to use index number with___ Operator.	: (Colon)

Q-5 Explain String Operators in Python.



STRING OPERATIONS

- + allows for concatenation, * performs repetition

```
>>> 'string' + 'one'    >>> 'string' * 3
'stringone'             'stringstringstring'
```

- Strings are indexed from the left AND right

```
>>>word='Python'    >>>word[0]    >>>word[-2]
                    'P'           'O'
```

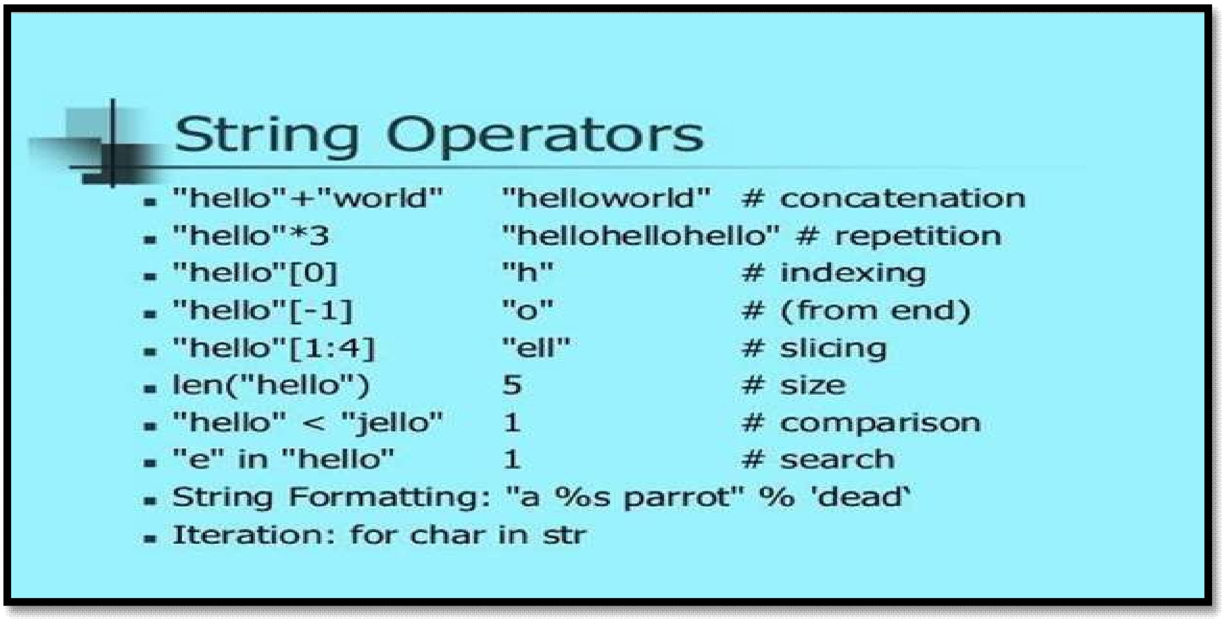
- Use [n:m] to slice strings

```
>>>word[2:4]    >>>word[:2]
'th'            'Py'
```

- Use len() to return the length of a string

```
>>>len(word)
6
```

TIP: Strings are immutable:
Word[3] = 'z' will NOT work.



String Operators

```
"hello"+"world"     "helloworld"    # concatenation
"hello"*3           "hellohellohello" # repetition
"hello"[0]          "h"             # indexing
"hello"[-1]         "o"             # (from end)
"hello"[1:4]        "ell"           # slicing
len("hello")        5               # size
"hello" < "jello"   1               # comparison
"e" in "hello"      1               # search
```

- String Formatting: "a %s parrot" % 'dead'
- Iteration: for char in str

Assume string variable a holds 'Hello' and variable b holds 'Python', then −

Operator	**Description**	**Example**
+	Concatenation - Adds values on either side of the operator	a + b will give HelloPython
*	Repetition - Creates new strings, concatenating multiple copies of the same string	a*2 will give -HelloHello
[]	Slice - Gives the character from the given index	a[1] will give e
[:]	Range Slice - Gives the characters from the given range	a[1:4] will giveell
in	Membership - Returns true if a character exists in the given string	H in a will give 1
not in	Membership - Returns true if a character does not exist in the given string	M not in a willgive 1

Detail :-

- The string operators can be used to perform different types of operations on the string.
- There are three types of string operators supported by python.
  - Basic Operator
  - Membership Operator

Basic Operator:-

- ✓ String operator support two types of basic operators :

  (i) Concatenate Operator :-

  - The concatenate operator can be used to combines two two or more string Values.

- Plus(+) operator can be used as concatenate operator inpython.
  - **Example :-**

```
>>> "hello" + "hi"

Output :- Hellohi
```

(ii) Replication Operator :-

- The Replication operator (*) is used to repeat particular string , character or symbol for given number of time .
- In this operator we have to give one integer paramter and string value.
  - **Example :-**

```
>>> 5  *  "hi "

Output :- hi hi hi hi hi
```

Membership Operator:-

✓ Membership operator is used to indicate possibilities of available members.

✓ There are two types of membership operators:

(i) in Operator :-

- The in operator returns true if particular character or string available in the given string otherwise false.

(ii) Not in Operator :-

- This operator returns true if particular character or string does not exist in given string otherwise returnfalse.

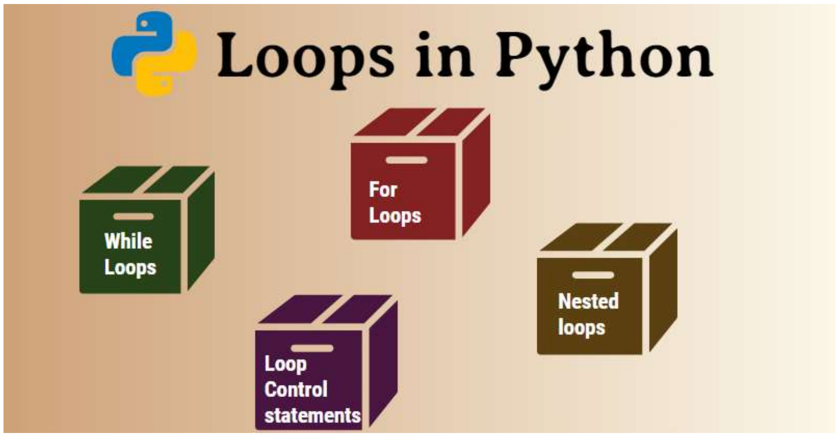
- **Example :-**

```
>>> str1 = “java programming”
>>> str2 = “HNS IT”
>>> str3 = “SEO Material”
>>> str4 = “java”
>>> str5 = “IT”
>>> str4 in str1                Output :-      True
>>> str5 in str2                Output :-      False
>>> str4 not in str1            Output :-      False
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	______string operator can be used for concatenation of multiple strings or words	+
2	______string operator can be used for repetition of multiple strings or characters	*
3	______string operator can be used to return characters from given range.	:
4	_____&_________are known as membership operators.	In & not in

Q- 6 Write note on Iteration OR Looping Statements.



Detail :-

- ➢ Generally the statements are executed sequentially.
- ➢ When user want to execute group of the statements at that time looping statements are used.
- ➢ The main use of loop is to repeat the statements for number of times.
- ➢ Python programming support following types of looping statements:

  - o While Loop
  - o For Loop

Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

While Loop :-

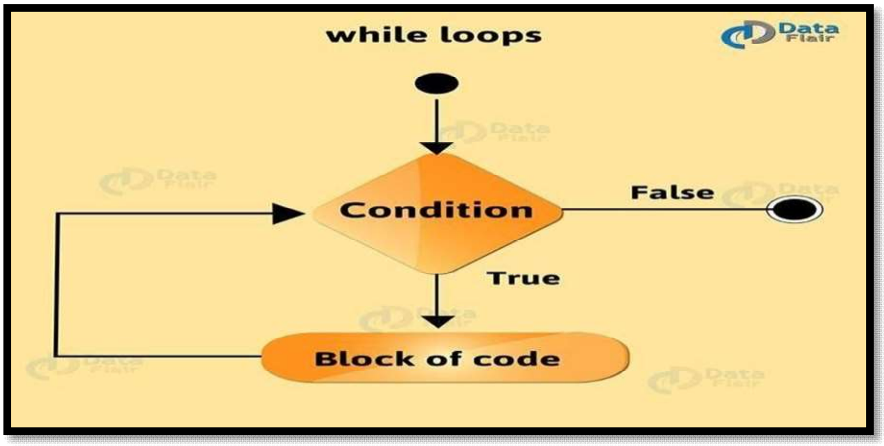
- ➢ While loop is known as entry – control loop.
- ➢ In this loop first of all condition will be checked and then after statementwill be execute.
- ➢ If condition become true then the statement following while will be execute otherwise loop will be terminated.

  - o **Syntax :-**

```
<initialization>
While(<condition>):
   <statement>
   <increment / decrement>
```

- **Example :-**

```
i=1

while(i<=10):
  Print(i)
  i=i+1
```



✓ **For Loop :-**

- ➢ For loop is used to execute block of the code for given number of times.
- ➢ First of all condition will be check and then after statement will be execute.
- ➢ For loop will iterate for the particular collection or list items.

- **Syntax :-**

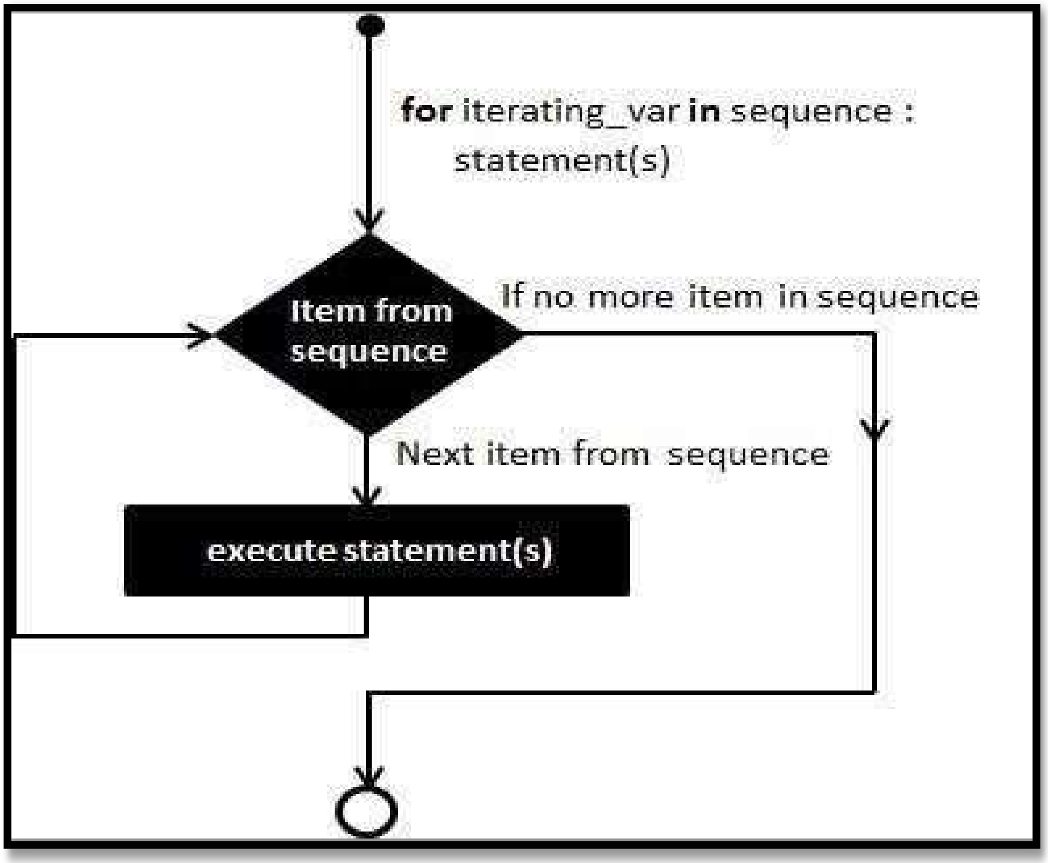
```
For <variable> in  <sequence>:
     <statement>
```

- **Example**

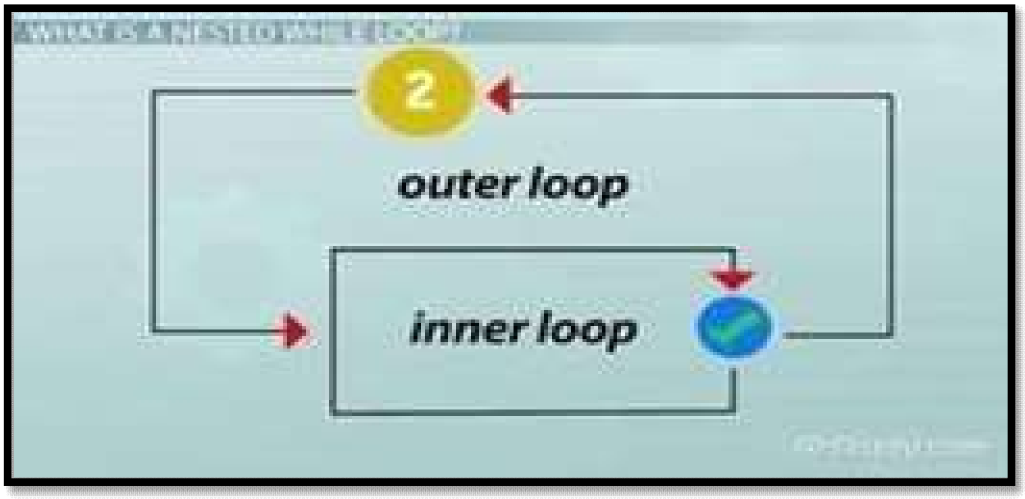
```
:-i=1
fruits = [‘apple’ , ‘banana’ , ‘mango’]
```

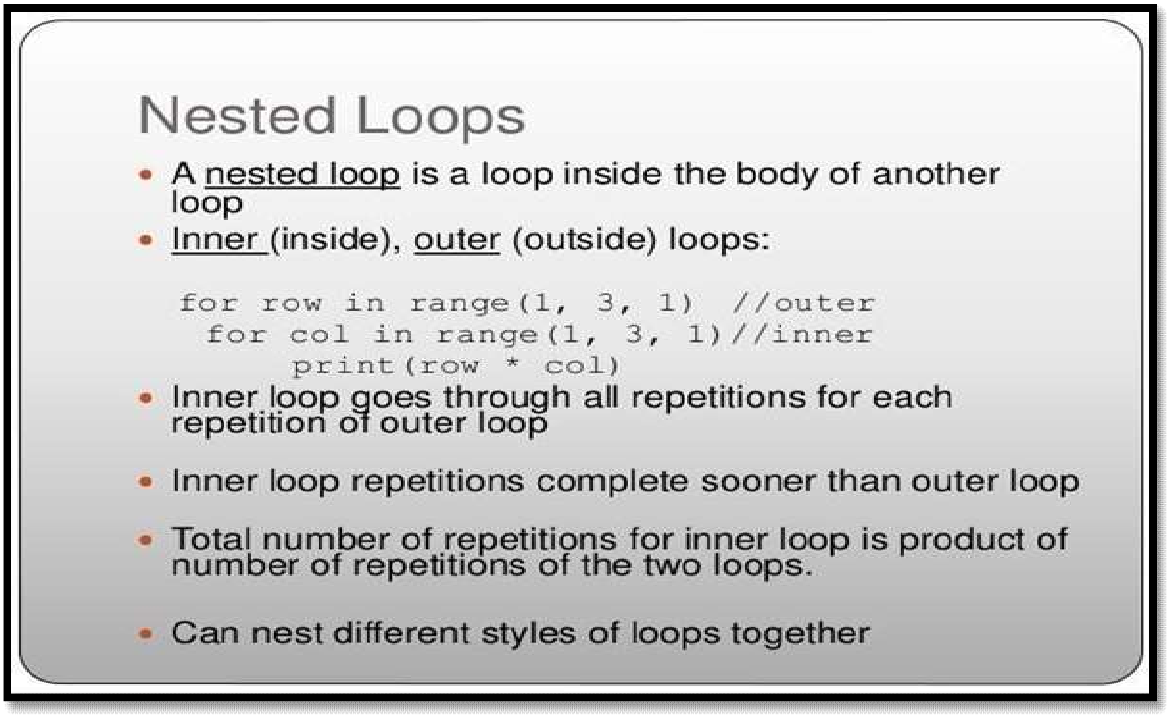
```
for  i  in  fruits:
     print(i)
```

- If a sequence contains an expression list, it is evaluated first.
- Then, the first item in the sequence is assigned to the iterating variable *iterating_var*.
- Next, the statements block is executed.
- Each item in the list is assigned to *iterating_var*, and the statement(s) block is executed until the entire sequence is exhausted.



- ✓ **Nested Loop :-**





Nested Loops

- A nested loop is a loop inside the body of another loop
- Inner (inside), outer (outside) loops:

```
for row in range(1, 3, 1)  //outer
  for col in range(1, 3, 1)//inner
        print(row * col)
```

- Inner loop goes through all repetitions for each repetition of outer loop
- Inner loop repetitions complete sooner than outer loop
- Total number of repetitions for inner loop is product of number of repetitions of the two loops.
- Can nest different styles of loops together

- If you want to create one loop inside another loop then it is called nested loop or nesting of loop.
- First of all outer loop will be execute first and then after inner loop will be execute.
  - **Syntax :-**

```
For <variable> in <sequence>:
        <statements>
  <statements>
```

  - **Example :-**

```
i=1
for I in range(1,10):
        for j in range(1,10):
        print (j)
        print()
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	Python support_________&________looping statements.	While & for
2	____________is loop inside body of another loop	Nested loop
3	In looping statement , the expression or condition must be followed by______.	: (Colon)
4	Looping statements are also known as _________statements	Iterative
5	The loop inside another loop is called _________loop	Inner loop

Q-7 What is Scoping ? Explain.

Detail :-

- Variable is acontainer thatuse to storedifferent values.
- Variable can onlyreachthearea in whichtheyaredefine , which is calledscope.
- Pythonsupporttwotypes of variable scopes:
  - Localscope of variable
  - Globalscope of variable
- Thescope of variable is used to decide placewhere you can access variable.
- If you define variable at the top level of your script or module then it is always global variable.

✓ Local scope of variable :-

- If youdeclarevariable inside the functionthen it is called local scope.
- Local variable have limited scope , it can be accessed by only the function in which it is declare.

  - **Example :-**

```
def
    my_function()
    a=10
    print("a=" , a)
    return
>>>my_func()
>>>print(a)  # name error : name 'a' is not defined
```

✓ Global scope of variable :-

- If you declare variable at the top of your script or module then it is always global.

- Global variable can be easily access anywhere within your script.

o **Example :-**

```
My_var = 20
def my_function()
    print("a=" , a)
    return
>>>print(a)
>>>my_function()
```

1 Word Question – Answer

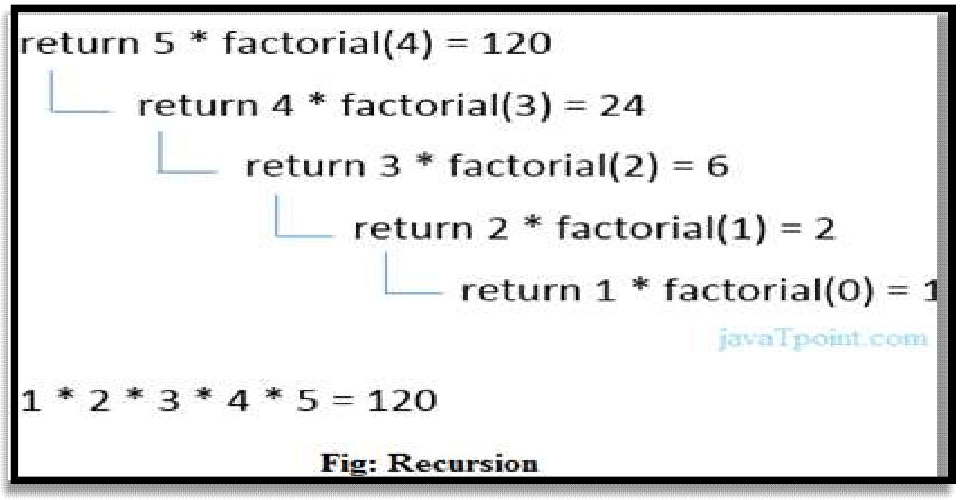
SR.NO.	QUESTION	ANSWER
1	In python , scope of variable can be_______ &_________.	Local & Global
2	The variable that declare inside function block and can be access within function is called _________.	Local scope
3	The variable that declare outside the function and can be access anywhere in the script is called _____________	Global scope

Q-7 What is Recursion ? Explain with example.

Detail :-

- Pythonprogramming supportrecursion as programmingconcept.
- Whenthefunctioncall itselfagainand again then it is calledrecursion.
- Recursionworklikealoop, you canconvert any loo to recursion.

- ✓ **Recursive Function :-**



```
return 5 * factorial(4) = 120
  └── return 4 * factorial(3) = 24
        └── return 3 * factorial(2) = 6
              └── return 2 * factorial(1) = 2
                    └── return 1 * factorial(0) = 1

1 * 2 * 3 * 4 * 5 = 120
```

Fig: Recursion

- ➢ Recursive function is called byexternalcode.
- ➢ If the base condition is available then the program do something meaningful otherwise exit.
- ➢ Functionhave to do somerequiredprocessingandthencall itself to continuerecursion.

- o **Example :-**

```
#Factorial using
recursion
def fact(n):

    if(n==0):
        return 1
  else:
return n*fact(n-1)
#calling function
    Print(fact(0))
    Print(fact(5))
```

- ➢ Nowtry to execute above function like afollowing:-
  - o **Example:-**

```
Print (fact(2000))
```

- Youwillgetfollowingerror at runtime:-
  - Runtimeerror:- Maximumrecursion depthexceeded in comparison.

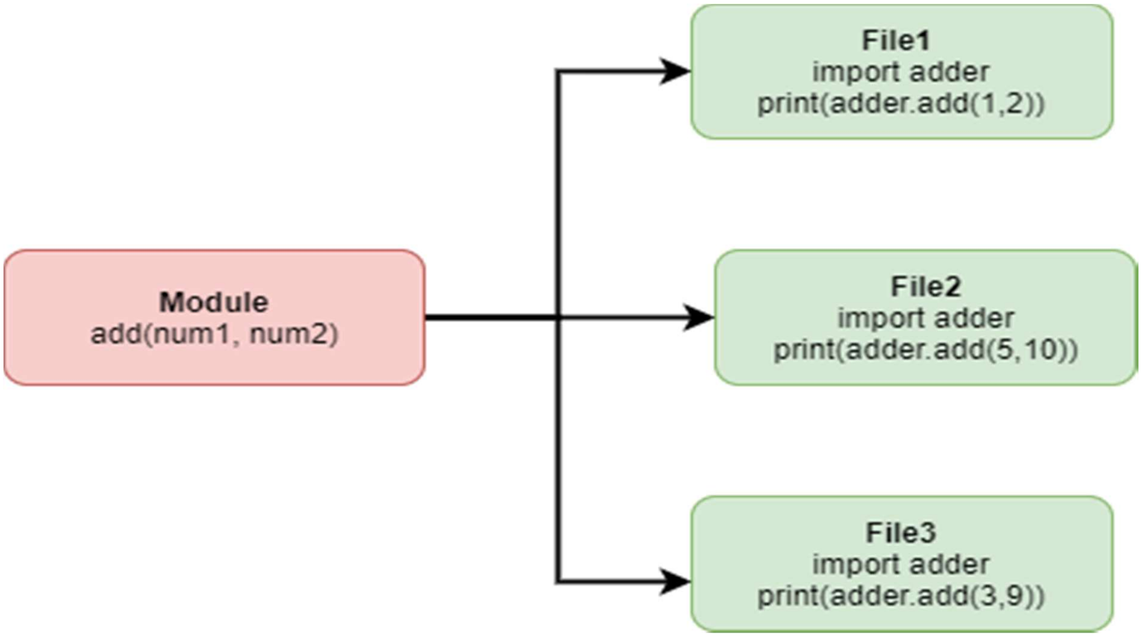
- The above error is available because python stop calling recursive function after 1000 calls by default.
- To change it you need to add following lines to starting of code.

```
Import Sys
Sys.setrecursionlimit(3000)
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	When the function call itself again and againthen it is called___________.	recursion
2	_____function is called by external	recursive

Q-8 What is Module ? Explain



Detail :-

- Moduleallow you to logically organizeyourpythoncode.
- To grouprelatedcodeintothemodule makes thecodeeasier to understand and us d.
- Simplymodule is a file having thepythoncode.
- Modulecan define function, variable and class.
- Modulecanalso incuderunnable code.
    - **Example:-**

[module – support.py]

```
Def  print_func(x):
          Print(“hello:”,x)
```

- Pythonprovidetwotypes of statements to load module:

□ **Import statement :-**

- You can use any pyton file as a module by executing import statement in other python file.
  - **Syntax :- import module 1 [,module 2 [,module N]]**
- When interpreter find out import statement it import the module if moduleis available.
  - **Example :- import module support**
    import support
    Support.print_func("zara")
- Module is loaded only once , but number of times it will be imported.
- The module search the path which is stored in system module as Sys.path.

✓ **From……import statement :-**

- Python's from statement lets you import specific attributes from a module into the current namespace.
- The from...import has the following syntax –
  - **Syntax :- from modname import name1[, name2[, ... nameN]]**
- For example, to import the function fibonacci from the module fib, use the following statement −
  - **Example :- from fib import Fibonacci**
- This statement does not import the entire module fib into the current namespace;
- it just introduces the item fibonacci from the module fib into the global symbol table of the importing module.

▪ **The from...import * Statement:-**

- It is also possible to import all names from a module into the current namespace by using the following import statement −

from modname import *

- This provides an easy way to import all the items from a module into the current namespace.

□

□ **Using the dir() Function**

□

- There is a built-in function to list all the function names (or variable names) in a module. The dir( ) function:
- module: import platform

□

- **Example :-**

```
x                =
dir(platform)
print(x)
```

□

- **Note: The dir() function can be used on all modules, also theones you create yourself.**

1 Word Question – Answer

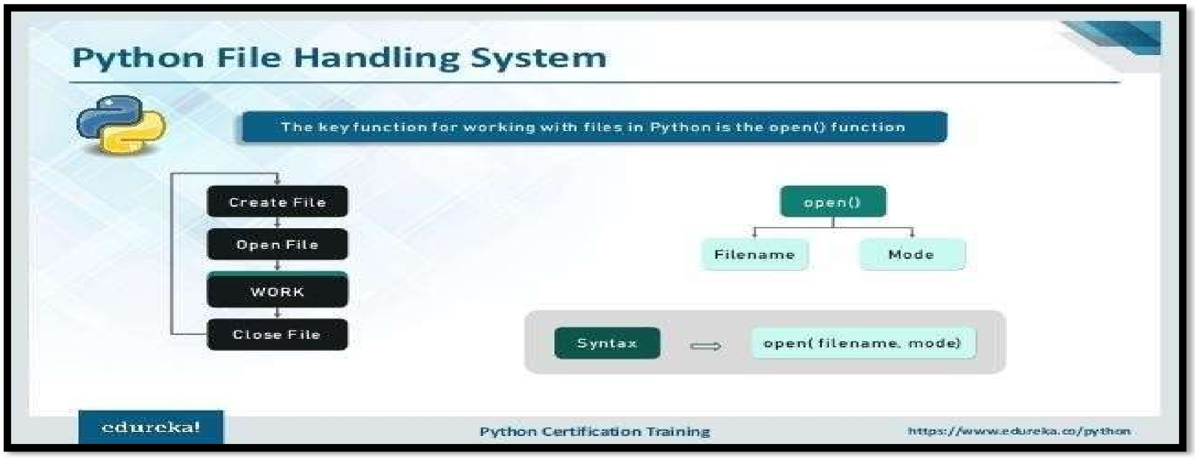
SR.NO.	QUESTION	ANSWER
1	Module is a________having python code.	file
2	____________statement can be used to import code from the given module.	import
3	To import all the files from the particular module we can use ________	Import *

Q-9What is file? Explain How to open , read , write and close the file –[file handling]

Detail :-

- File is used to store related information permanently.
- When we want to read from or write to a file , we need to open it first.
- There are following file operations available:
  - Open a file
  - Read a file
  - Write a file
  - Close a file

□ **Open a**


Python File Handling System

The key function for working with files in Python is the open() function

Syntax ⟹ open(filename, mode)

- Python provide open() to open particular file.
- We have to provide file mode which can be read – ‘r’ , write – ‘r’ or append – ‘a’.
- The default mode is reading mode.
- **File Modes :-**
  (1) ‘R’ :- This file mode is used to open any file for reading purpose.
  (2) ‘w :- This file mode is used to open any file for writing purpose.
  (3) ‘a’ :- This file mode is used to open any file for appending data.

Example :- f = open (“text1.txt”)
f = open (“text2.txt”,”w”)

□ **Write data to the file :-**

- To write data from the file , we need to open it into write mode.
- To write data into file , write() can be used.

Example :- f = open (“text1.txt”, “w”)
```
f.write(“hello”)
print(“written  successfully”)
f.close()
```

□ **Read data from the file :-**

- To read data from the file , we need too pen it into read mode.
- To read data from the file , read() is used.

Example :- f = open (“text1.txt”, “r”)
```
print (f.read())
f.close()
```

- **How to close file :-**
  - When we complete all the related operations the new need to close it properly.
  - To close any file , close() is used.

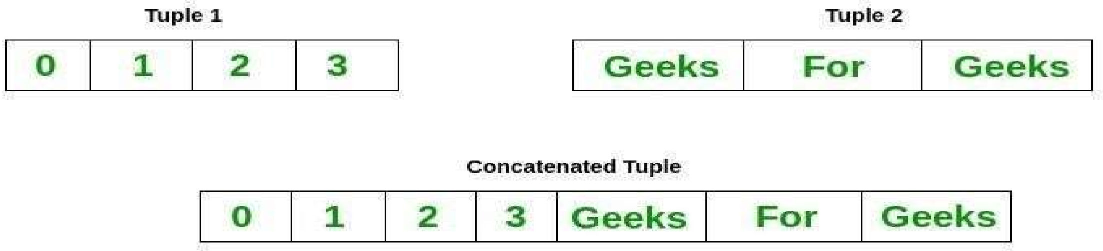
Example :-

```
 f = open (“text1.txt”, “r”)
print (f.read())
 f.close()
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	________is used to store related imformations permanently.	file
2	_________can be used to open particular file	Open()
3	_________can be used to read data from particular file.	Read()
4	__________can be used to write data to the particular file.	Write()
5	___________ can be used to close particular file.	Close()

Q-10 Write note on tuple .



Tuple 1

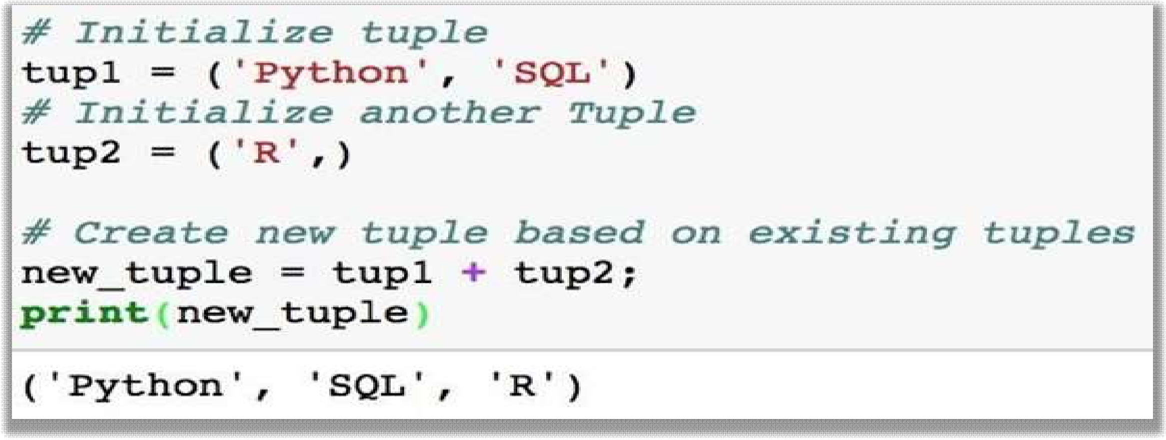
0	1	2	3

Tuple 2

Geeks	For	Geeks

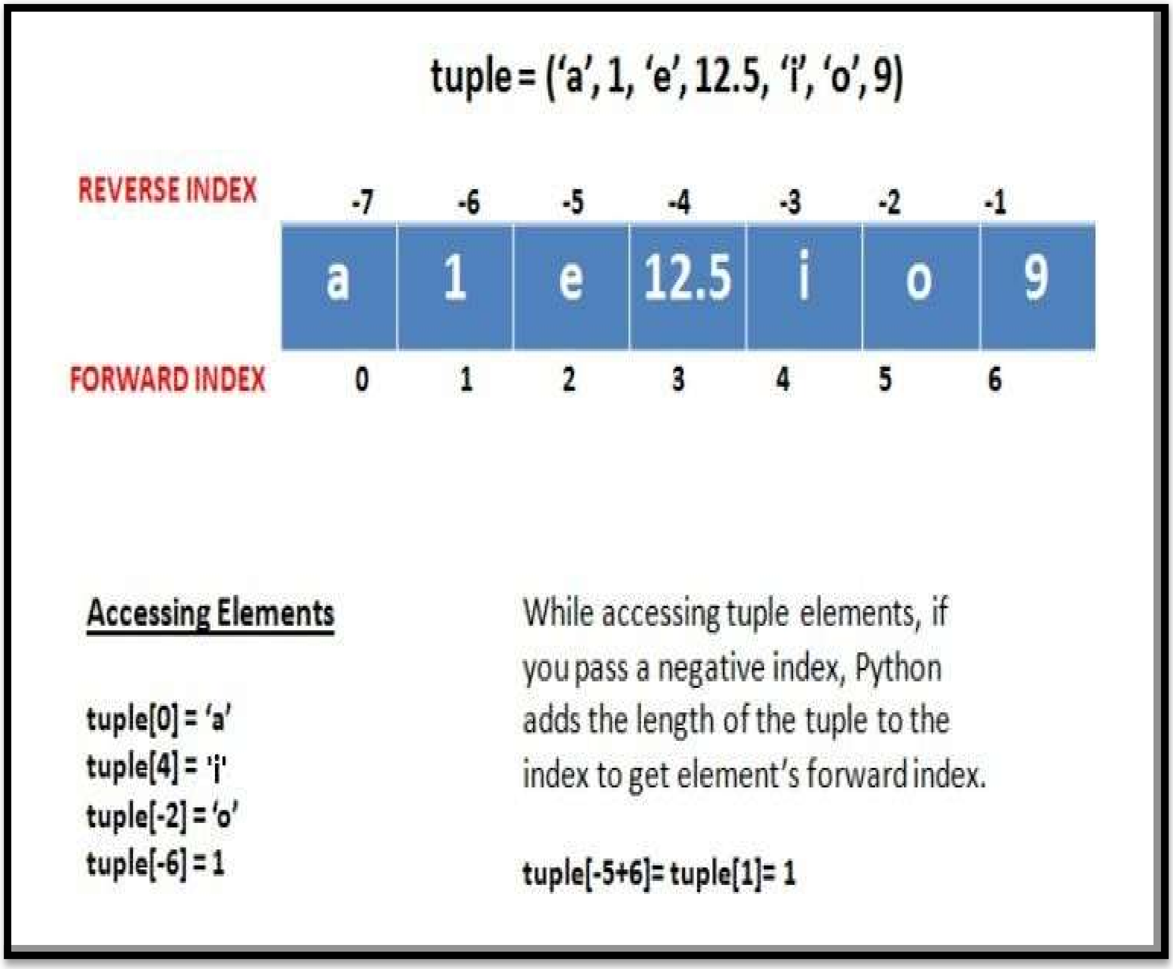
Concatenated Tuple

0	1	2	3	Geeks	For	Geeks



```
# Initialize tuple
tup1 = ('Python', 'SQL')
# Initialize another Tuple
tup2 = ('R',)

# Create new tuple based on existing tuples
new_tuple = tup1 + tup2;
print(new_tuple)
```

```
('Python', 'SQL', 'R')
```



tuple = (‘a’, 1, ‘e’, 12.5, ‘i’, ‘o’, 9)

REVERSE INDEX	-7	-6	-5	-4	-3	-2	-1
	a	1	e	12.5	i	o	9
FORWARD INDEX	0	1	2	3	4	5	6

Accessing Elements

tuple[0] = ‘a’
tuple[4] = ‘i’
tuple[-2] = ‘o’
tuple[-6] = 1

While accessing tuple elements, if you pass a negative index, Python adds the length of the tuple to the index to get element’s forward index.

tuple[-5+6]= tuple[1]= 1

Detail :-

- Tuple is a sequence of immutable python object.
- Tuple is very similar to the list but there may be following difference between list and tuple.
  - Tuple can not be change or update during execution.
  - We have to use round brackets( ) to represent tuple.
  - User can create empty tuple by representing empty brackets ( ).
  - Creating tuple is as simple as representing different values supported by comma (,).

- **Example :-**

```
T1=(‘a’,’b’,’c’,’d’,’e’)
T2=(1,2,3,4,5)
T3=(17,25,”hi”,”how”,4)
```

✓ **How to access value from the tuple :-**

- To access value from the tuple use square bracket [ ] with particular index number or range.
- **Example :-**

```
T1=(‘a’,’b’,’c’,’d’,’e’)
 Print (“t1[2]:”  ,  t1[2])
  o/p:- c
```

✓ **How to update value in the tuple :-**

- In tuple it is not possible to change the value during execution.
- But it is possible to merge more than one tuples at the same time.
- **Example :-**

```
T1=(‘a’,’b’,’c’,’d’,’
e’)T2=(1,2,3,4,5)
T3= T1 + T2
Print (T3)  o/p :- a,b,c,d,e,1,2,3,4,5
```

✓ **How to delete tuple :-**

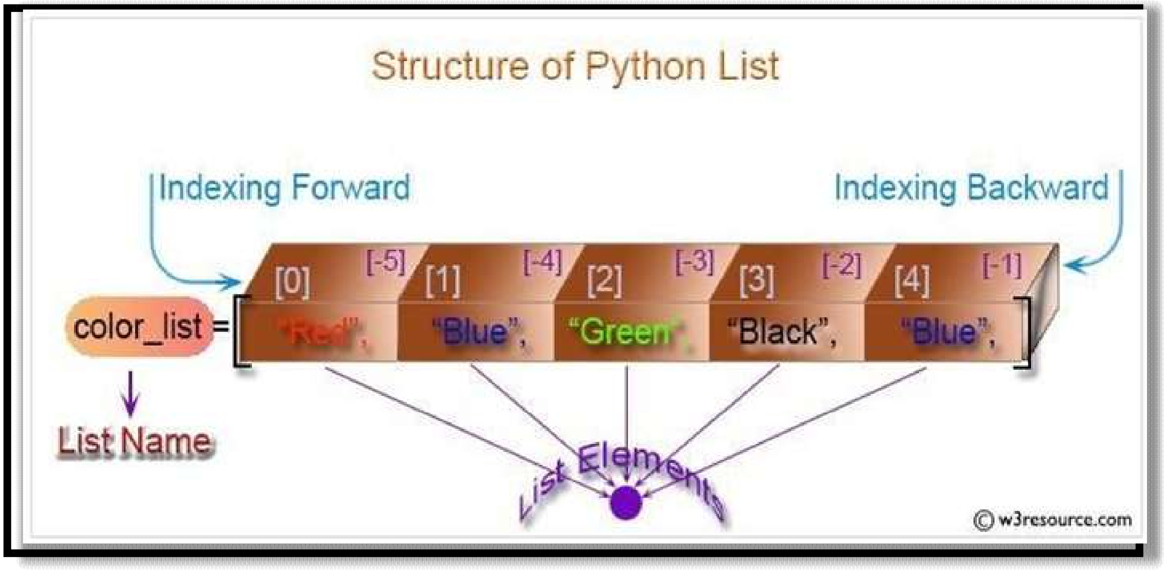
- To remove particular value of the tuple is not possible.
- So that we have to delete or remove entire tuple using **del** statement.
- **Example :-**

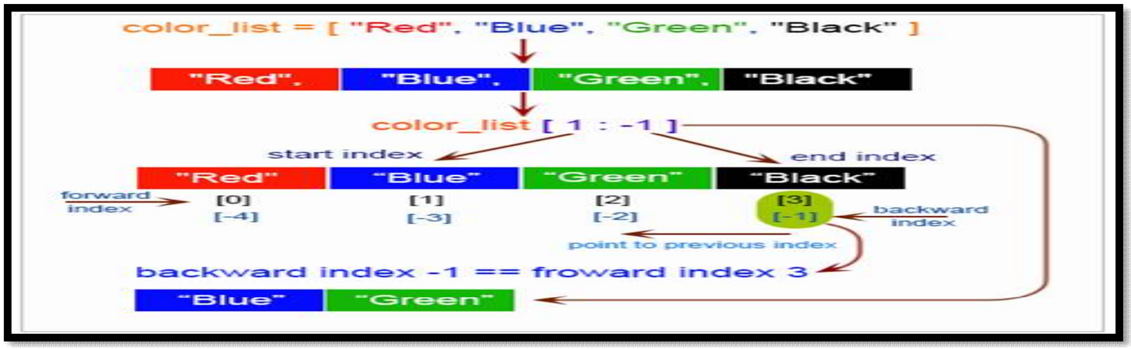
```
T1=(‘a’,’b’,’c’,’d’,’e’)
del    (T1)
Print (T1)
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	Tuple is___________object in python.	immutable
2	Tuple can be represented by________brackets.	( ) (round)
3	_____statement can be used to remove entire tuple.	del

Q-11 Write note on List.





Detail:-

- Thelist is mostversatile datatype in python.
- List can be represented by squarebracket[] , separated by comma(,).
- Inpython youcan createsimplelistlike following:

- **Example :-**

L1 = [“abc” , “xyz” , 2000 ,1999]
L2 = [1,2,3,4,5,6,7]

✓ **How to access value from the List :-**

- To access value from the List , use square bracket [ ] with particular indexnumber or range.

- **Example :-**

```
L1 = [“abc” , “xyz” , 2000 ,1999]
L2 = [1,2,3,4,5,6,7]
Print (“list1[0]:” , L1[0])                    o/p :- abc
Print(“list2[1:5]:” , L2[1:5])                 o/p :-  2,3,4,5
```

✓ **How to update value in the List :-**

- You can update single or multiple elements of list by assigning new valuefrom right to left.

Example :- **L1 = [“abc” , “xyz” , 2000 ,1999]**

```
Print (L1[2])               o/p :- 2000
L1[2] = 2005
Print (L1[2])               o/p :- 2005
```

✓ **How to delete value in the List :-**

- To remove list element , you can use **del** statement , if you know which element you are going to delete.
- You can also use remove() , if you do not know which element you are going to delete.

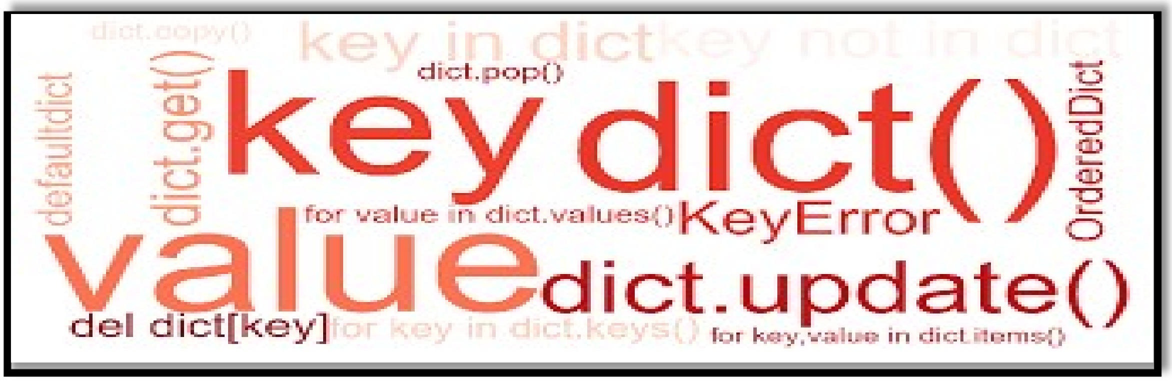
- **Example :-**

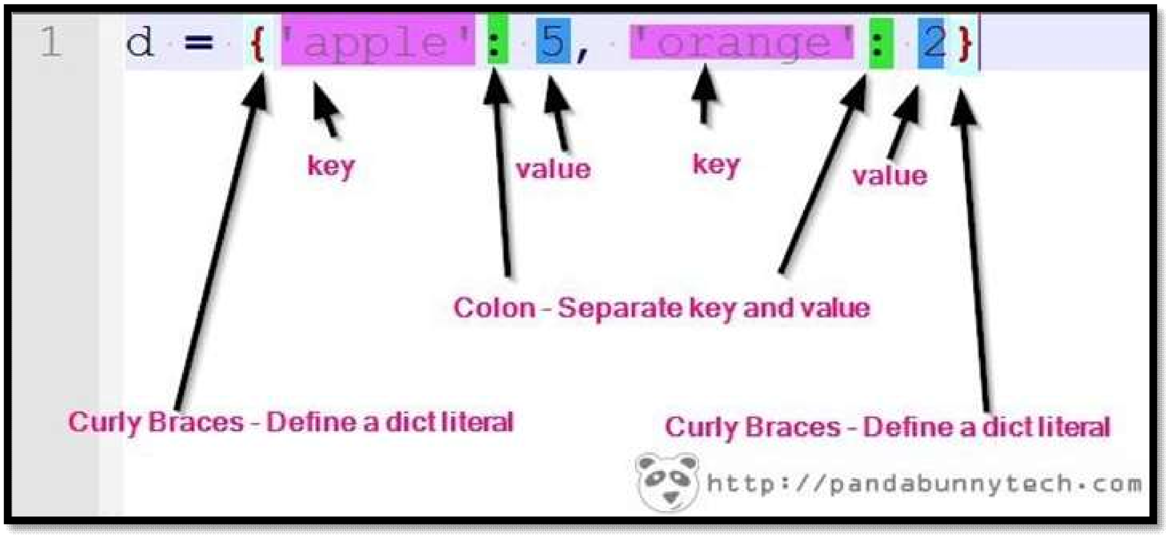
```
L1 = [“abc” , “xyz” , 2000 ,1999]
Print (L1[2])
del (L1[2])
Print (L1)
Del    (L1)
Print(L1)
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	List is_______object in python.	mutable
2	List can be represented by __________ brackets.	[ ] (square)
3	To remove particular element from the list ____statement can be used.	del

Q-12 Write note on Dictionary.





```
d = {'apple': 5, 'orange': 2}
```

key
value
key
value
Colon - Separate key and value
Curly Braces - Define a dict literal
Curly Braces - Define a dict literal
http://pandabunnytech.com

Detail:-

- In dictionary we have to manageourdata by key and valuepair.
- In dictionaryeach key is separatedfromitsvalue usingcolon(:).
- In dictionarythe key and value pairs are separated by comma(,).
- Thekeys are always unique but valuescannot be.
- We can createourdictionaryusingcurlybrackets { }.

- **Example :-**

```
D1 = {‘name’ : ‘zara’ , ‘city’ : ‘rajkot’ , ‘age’ : 5)
```

✓ **How to access value from dictionary :-**

- To access value from the dictionary , we have to use square bracket with particular key.

- **Example :-**

```
D1 = {‘name’ : ‘zara’ , ‘city’ : ‘rajkot’ , ‘age’ : 5)
Print (“D1[‘name’]:” , D1[‘name’])                o/p :- zara
```

✓ **How to update value in dictionary :-**

- You can update dictionary by adding new entry or by modifying existing entry.

- **Example :-**

```
D1 = {‘name’ : ‘zara’ , ‘city’ : ‘rajkot’ , ‘age’ : 5)

 D1[‘school’] = ‘G.T. Girls’        # adding new entry
 Print (D1)
 D1[‘age’] = 1                      #modifying  existingentry
 Print(D1)
```

✓ **How to delete value in the dictionary :-**

- You can remove particular element from dictionary as well as you can clear entire dictionary.
- To remove all the entries from dictionary , clear() is used.

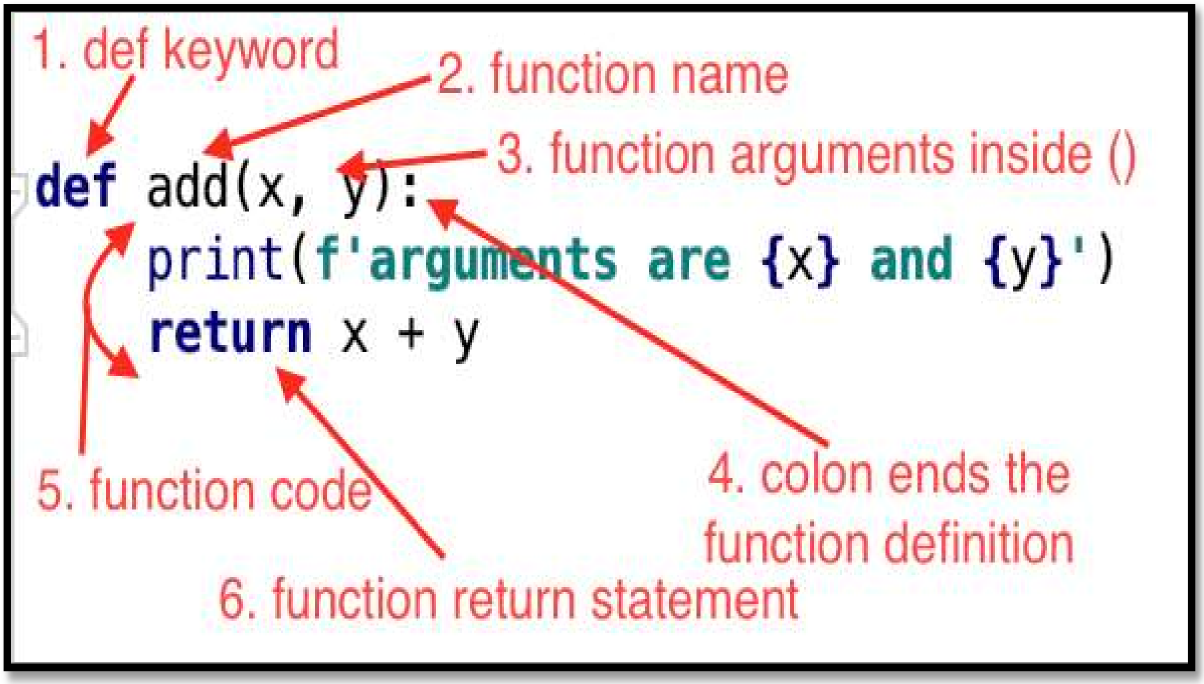
- **Example :-**
- **D1 = {‘name’ : ‘zara’ , ‘city’ : ‘rajkot’ , ‘age’ : 5)**

```
del  (D1  [‘name’])            #  removing  single  entry
print(D1)
del (D1)               # deleting entire dictionary
print(D1)
D1.clear()          # remove all entries in dictionaryPrint(D1)
```

1 Word Question – Answer

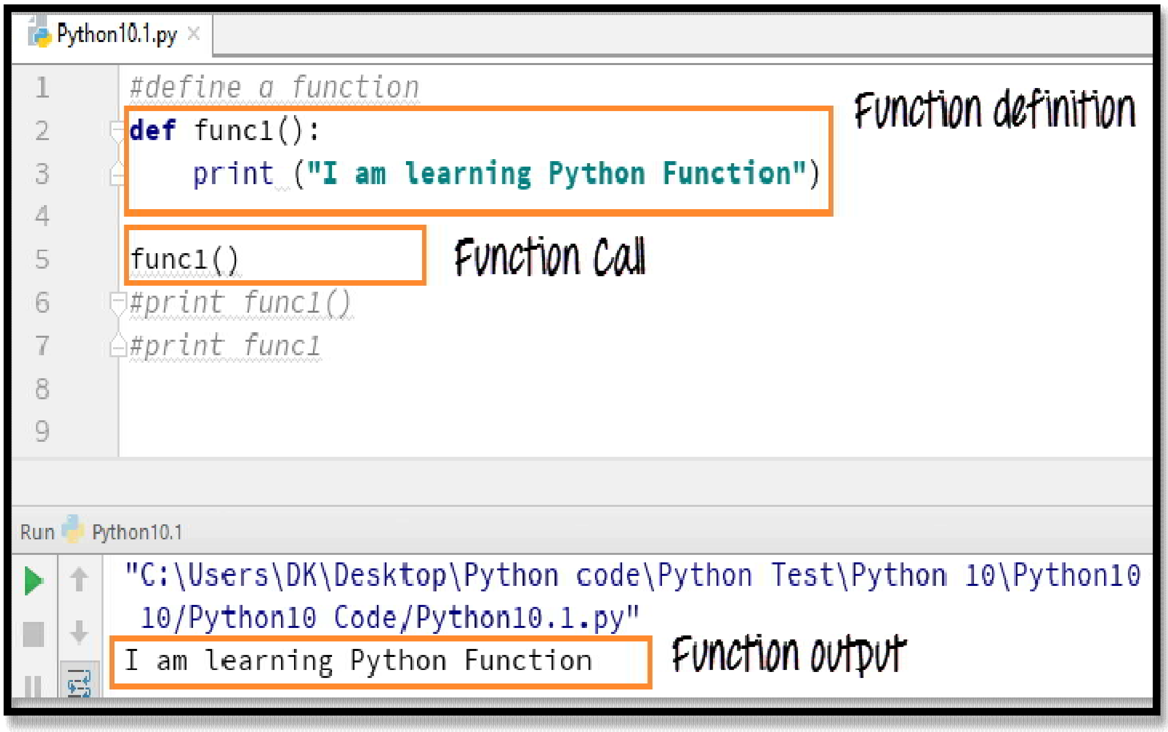
SR.NO.	QUESTION	ANSWER
1	Dictionay can be represented by__________ brackets.	{ } (curly)
2	In dictionary ,each key is separated from its values by_____.	: (colon)
3	Keys are always__________within dictionary.	Unique
4	To remove an entire dictionary _______ statement can be used.	del

Q-13 How to define function ? Explain.



```
def add(x, y):
    print(f'arguments are {x} and {y}')
    return x + y
```

1. def keyword
2. function name
3. function arguments inside ()
4. colon ends the function definition
5. function code
6. function return statement



Python10.1.py

```
#define a function
def func1():
    print ("I am learning Python Function")

func1()
#print func1()
#print func1
```

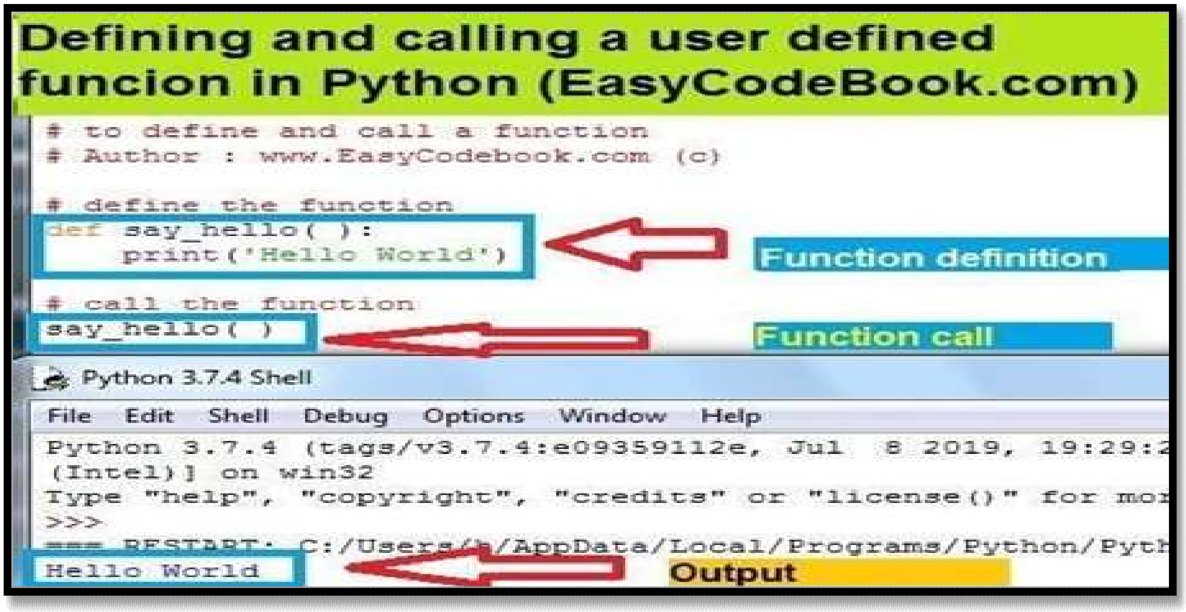
Function definition

Function Call

Run Python10.1

```
"C:\Users\DK\Desktop\Python code\Python Test\Python 10\Python10
 10/Python10 Code/Python10.1.py"
I am learning Python Function
```

Function output



Defining and calling a user defined funcion in Python (EasyCodeBook.com)

```
# to define and call a function
# Author : www.EasyCodebook.com (c)

# define the function
def say_hello( ):
    print('Hello World')

# call the function
say_hello( )
```

Function definition

Function call

Python 3.7.4 Shell

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:2
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for mor
>>>
=== RESTART: C:/Users/a/AppData/Local/Programs/Python/Pyth
Hello World
```

Output

Detail:-

✓ Defining a Function :-

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).
- You can place any number of arguments inside the brackets().
- The code block of every function must start with colon(:).
- The statement of the block must be exit with 'return' keyword.

o **Syntax :-**

```
def <function name> (parameters):
<block of code>
          Return
```

Example :-

```
def sp():
          Print("hello     sp")
return
```

✓ Calling a Function :-

o If you create your own function , then you can execute it by calling the function with its name.

o You have to take care about name of the function and argument of thefunction.

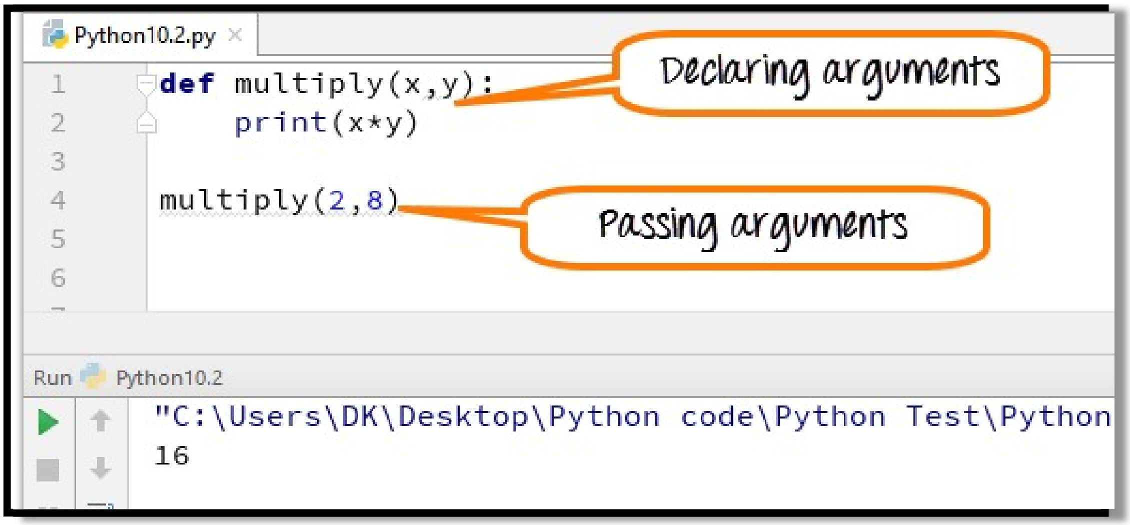
o **Syntax :-**

```
<function name> (parameters):
```

o **Example :-**

```
Sp()
```

Passing parameters to function :-



Python10.2.py

```
def multiply(x,y):
    print(x*y)

multiply(2,8)
```

```
"C:\Users\DK\Desktop\Python code\Python Test\Python
16
```

- You can pass multiple arguments by separating it with comma (,) .
- while passing multiple arguments always take care about number of arguments and types of arguments.

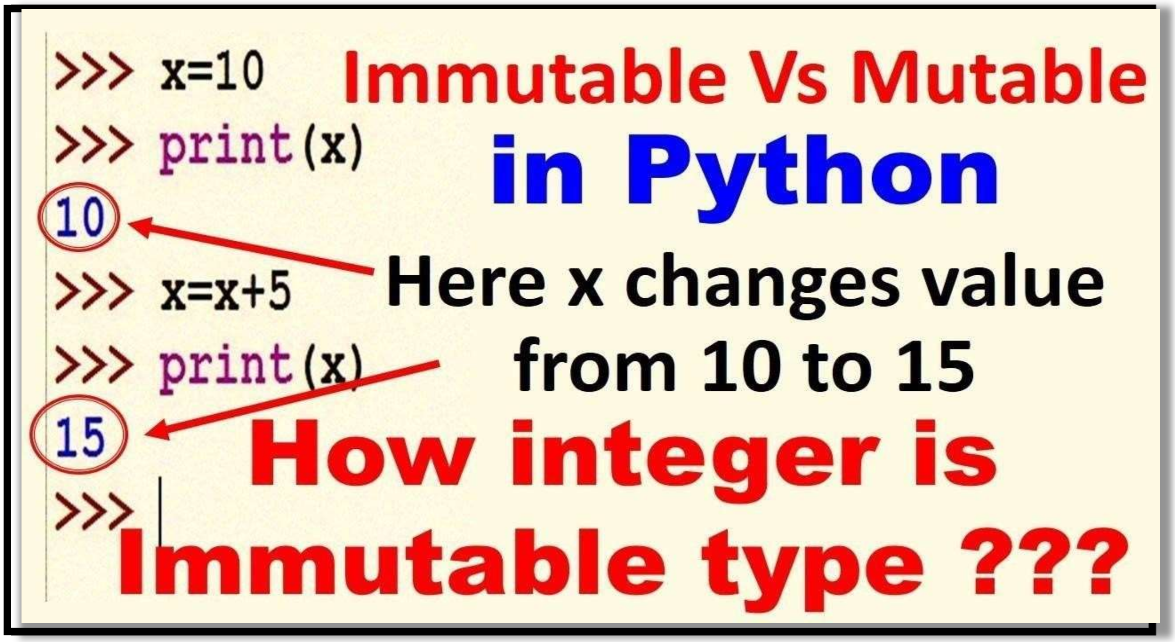
- **Syntax :-**

```
def <function name> (p1,p2,p3,…..,pn):
        <block of code>
Print Return
   Example :-
   def sp(str1 ,str2):
         ("hello         sp")
         Print(str1)
         Print(str2)

         Return
```

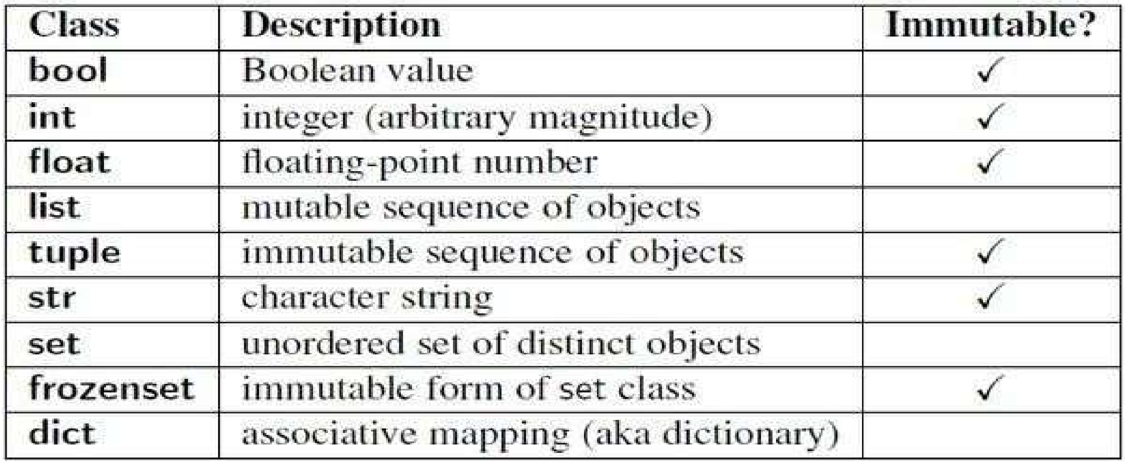
1 Word Question – Answer

1	__________keyword can be used to define function.	def
2	The code block within every function starts with____________& followed by__________.	Function name & colon(:)
3	Function can be called by just providing _____________.	Function name
4	User can pass some values with function which known as_____________ _	Parameters
5	To return some values by function________ _ statement can be used.	return

Q-14 Explain Mutability in brief.



```
>>> x=10
>>> print(x)
10
>>> x=x+5
>>> print(x)
15
>>>
```



Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	associative mapping (aka dictionary)	

Detail :-

- Everything in python is an object.
- Python represent all it’s data as object.
- The mutability of object decided by its type.
- Some of the object like list and dictionary are mutable.
- Mutable means you can change the content without changing their identity.
- Some other objects like tuple and string are immutable means that can not be change.
- Variable in a python also support mutability , means if you call same method with same variable can be muted anytime by other method.
- List object support mutability like following :

Example :-

```
My_list= [10,20,30]
Print (my_list)
My_list[0]=40
Print(my_list)
```

- Unlike tuple , the list is mutable it means we can change the value by assigning new value directly.

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	________is mutable object in python.	List
2	________is immutable object in python	Tuple
3	________means you can change the content without changing their identity.	Mutability

SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.
(AFFILIATED TO SAURASHTRA UNIVERSITY)

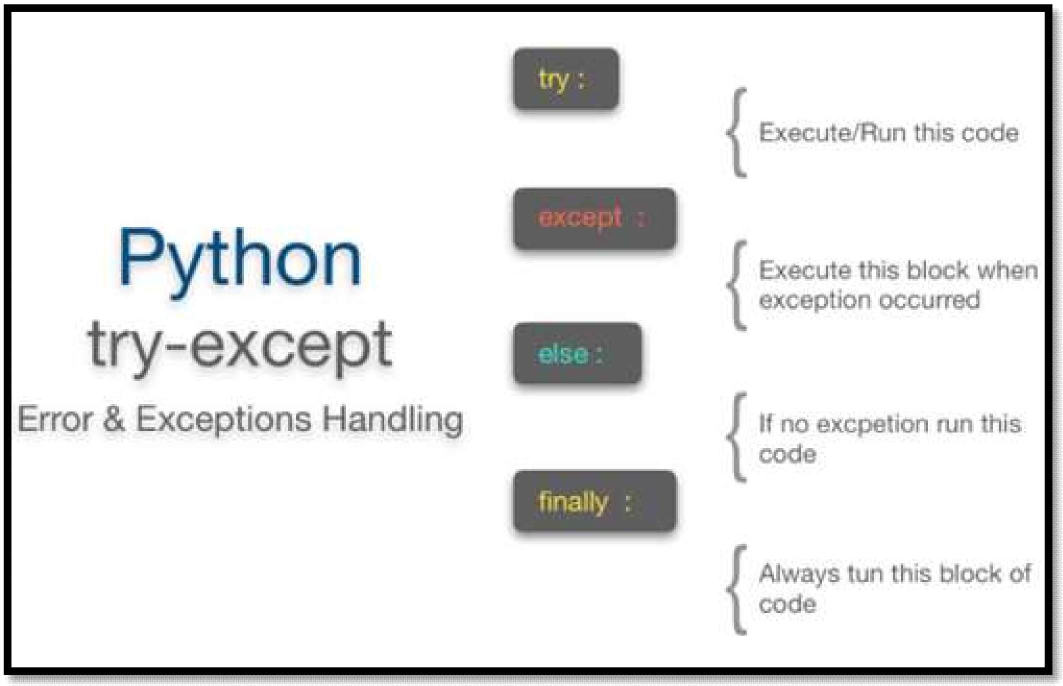
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001
Ph.No–(0281)2440478,2472590

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001
Ph.No–(0281)2471645

CHAPTER-2
OOP Using Python

- **Handling Exception**
- **Exception as Control Flow**
- **Assertion**
- **Abstract Data Type**
- **Class**
- **Inheritance**
- **Encapsulation**
- **Information hiding**
- **Search Algorithm**
- **Sorting Algorithm**
- **Hash table**

Q-1 What is Exception ? How to handle Exception?



Python
try-except
Error & Exceptions Handling

try : — Execute/Run this code

except : — Execute this block when exception occurred

else : — If no excpetion run this code

finally : — Always tun this block of code

Exception Handling in Python

01 What are exceptions?

Catching Exceptions in Python

03 Try-except statement

04 Adding the final clause

Raising Exceptions

06 User-defined exceptions

Detail :-

- An exception is event that available during execution of a program.
- python have many built-in exception.
- Python provide two very important features to handle an unexpected error in your program.
    1. Exception handling
    2. Assertion

✓ **Exception handling**

- If you have some doubtful code that create exception at that time you need to handle the exception.
- Python provide try ,except else and finally keyword to store and solve the error.

Try and exception:

- Try block support the code that you want to execute.
- Single try statement can have multiple except statement. Except statement can support handling of any exception.
- Try block contains the statement that must be thrown different type of exception.
- You can use except clause after try statement which can be multiple.
- At last you can include else clause , the code in else statement will be execute if the code in try block do not raised any error.

o **Syntax :-**

```
Try :
          Do your operations here
           Except  Exception1:
               If  there  is  exception1  ,  than  execute  this  block
           Except Exception2:
               If there is exception2 , then execute this block
```

Else:

If there is no exception , than execute this block

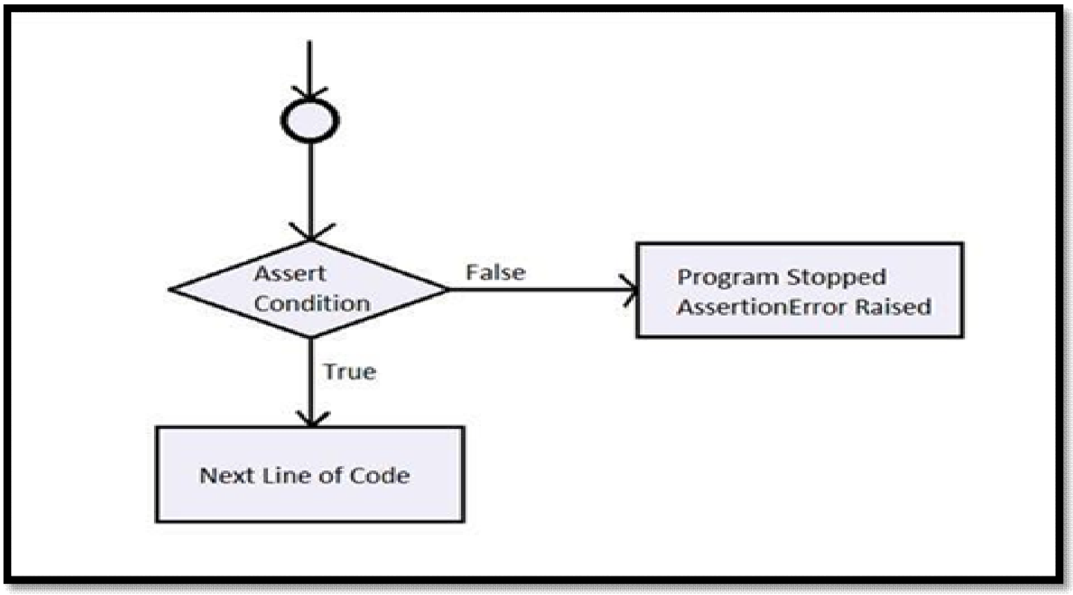
o **Example:-**

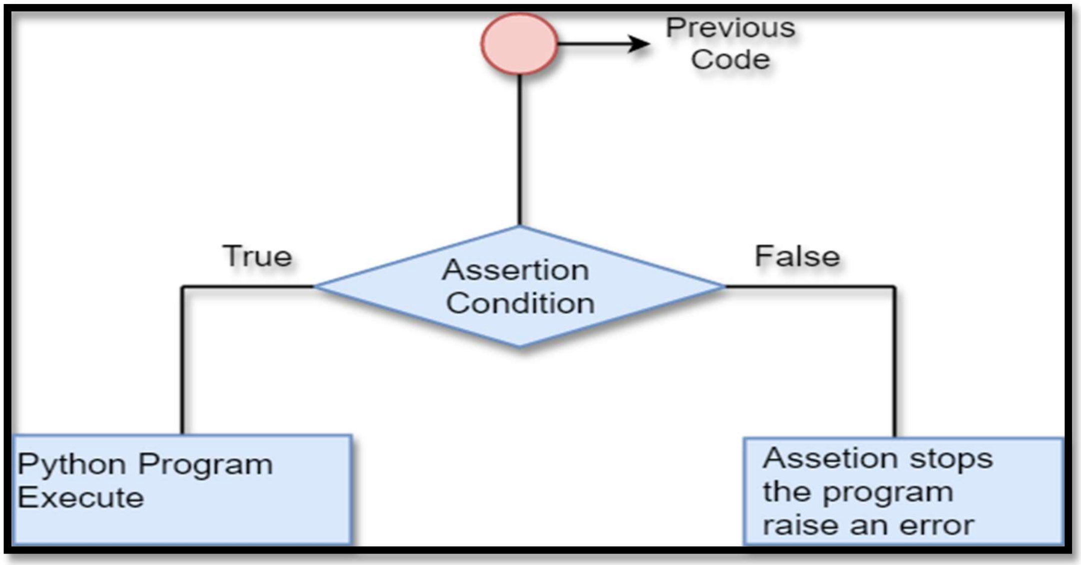
```
Try :
                F=open("testfile","w")
                f.write("hello friends")
Except:
            print ("error , can not find file")
Else:
            print("written successfully")
            f.close()
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	____________block support the code that you want to be execute.	try
2	_______statement can support handling of any execption.	except
3	if the code in try block do not raised any error,than the statement following ____________will be execute.	else

Q -2 Explain Assertion in brief.


Assert
Condition
False
Program Stopped
AssertionError Raised
True
Next Line of Code


Previous
Code
True
Assertion
Condition
False
Python Program
Execute
Assetion stops
the program
raise an error

Detail :-

- An Assertion is rule based checking that you can turn on and turn off when you done with your testing of program.
- Assertions are carried out by assert statement.
- You have to place assert statement at the starting of your optionto check for valid input.

□ **Assert Statement:**

- When interpreter encounter asserts statement , python execute the given expression which is true.
- But if the expression is false then python raised AssertionError Exception.
  - **Syntax :-**

```
Assert Expression [ , arguments ]
```

- If assertion fail , python use ArgumentExpression as argument for AssertionError.
- AssertionError exception can be handle like any other exception using try , catch , except statement.
- If the exceptions are not handle than terminate the program and produce the Trackback.
- Consider following example that indicate how AssertionException raised , if expression is false.
  - **Example :-**

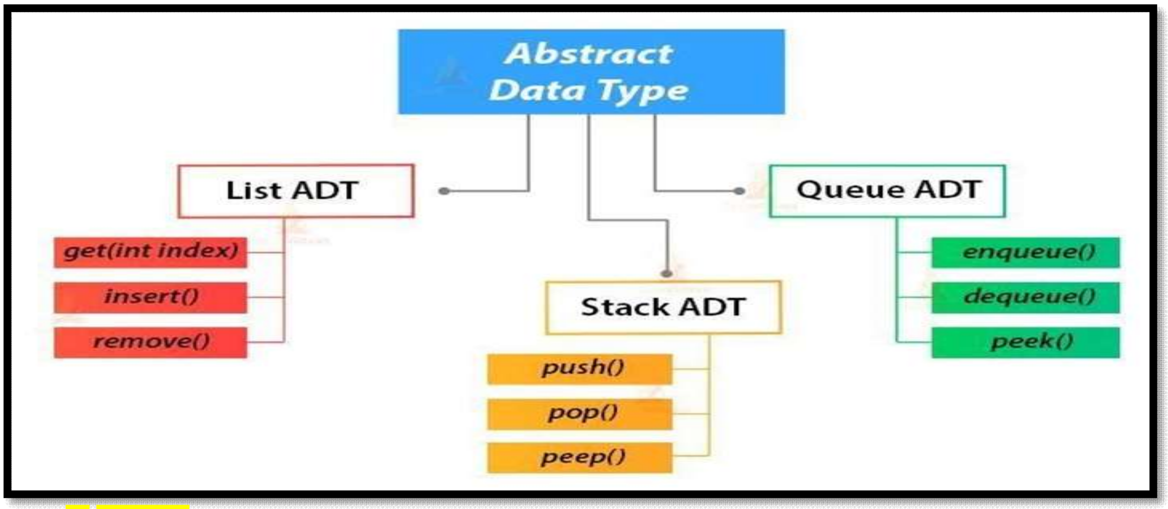
```
def no(i)
      Assert  (i>=0)  ,  “no  is  less  than
      zero”Return (i)
```

```
#calling function
      Print (no(5))
      Print (no(-5))
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	Assertions are carried out by__________ statement.	assert
2	If the exceptions are not handle than terminate the program and produce__________.	Trackback

Q -3 Explain Abstract Data type with class.



Detail :-

- Abstract data type is a type or class for the object whose behavior is define d by set of values.

- Abstract data type indicate “what’s operations are to be perform but not how the operations will be implemented.
- Abstractions is most powerful idea in python.
- Abstract data type provide very important feature called “modularity”.
- Classes are python representation for abstract data type.
- Abstract data type include both data and operation at the same time.
- There are following abstract data type available in python :
  - **Stack ADT**
  - **Queue ADT**
  - **List ADT**

✓ Stack ADT

- A stack contains elements of same type arranged in sequential order. Stack all the operatio ns are performed at top of the stack.

- Stack is performed op eration by LIFO[last in first out]. Stack supportfollowing operation or method.

1. Stack():-

- Stack create a new sta ck that is empty.
- it needs no parameter and return an empty stack.

2. Push(item):-

- Add new item to the t op of the stack it needs the item and retu rn nothing.

3. Pop ()

- Remove the top item from the stack. Return the item The stack is modified.

- **Example:-**

```
class Stack:
    def__init__(self):

        self.items = []
        def isEmpty(self):
                    return  self.items  ==  []
    def push(self, item):
         self.items.insert(0,item)
    def peek(self):
                return   self.items[0]
    def pop(self):
                    return  self.items.pop(0)
    def front(self):
                    return    self.item[(self.items)]
    def size(self):
                    return len(self.item s)


            s       =       Stack()
            s.push('hello')
            s.push('true')
            s.push('bca6b')
            print(s.items)
            print(s.size())
            print(s.peek())
            print(s.pop())
            print(s.items)
```

✓ Queue ADT:-

- Queue contains eleme nt of some type arranged in sequential o rder.
- Operation can be take place at both ends insertion is done at th e end and deletions is done at front.

1. **Queue()**
   - Create a new queue that is empty.
   - It need no parameters and return an empty queue.

2. **Enqueue(items)**
   - Add a new item to the rear of the queue.
   - It need the item and returns nothing.

3. **Dequeue()**
   - Remove the item from the front of the queue.
   - The queue is modified.

Example:-

```
class Queue:
def__init__(self):

self.items = []
def isEmpty(self):
          return   self.items   ==   []
def enqueue(self, it em):
   self.items.insert(0,item)
   return self.item[len(self.items)-1]

def size(self):
          return len(self.it ems)
```

```
q=Queue()
q.enqueue(4)
q.enqueue('dog')
q.enqueue(True)
print(q.size())
```

✓ List ADT:-

- List contains elements if same type arranged in sequential ord er. List contain following operation.

1.get()

It return eleme nt from the list at given position.

2.Insert()

Insert a new el ement of any position of list.

3. Remove ()

It remove first element from non empty list.

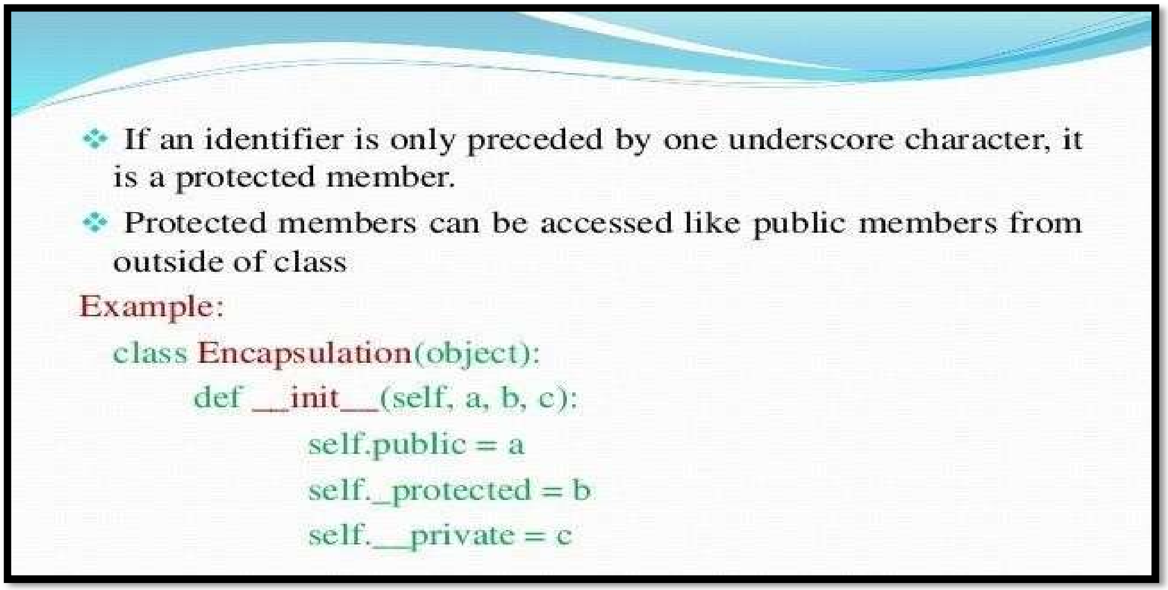
4. Replace()

Replace element of any position by other element.

1 Word Question -Answer

SR.NO.	QUESTION	ANSWER
1	ADT stands for_____________.	Abstarct Data Type
2	Abstract data type provide very important feature called___________.	Modularity
3	__________contains elements of some type arranged in sequential order.	Queue
4	_________Remove the top item from the stack.	Pop
5	_________ _is used to remove first element from non empty list.	remove

Q -4 Explain Encapsulation with information hiding.



- If an identifier is only preceded by one underscore character, it is a protected member.
- Protected members can be accessed like public members from outside of class

Example:

```
class Encapsulation(object):
    def __init__(self, a, b, c):
        self.public = a
        self._protected = b
        self.__private = c
```

Detail :-

- The object variable should not always access directly.
- The object variable sometimes changed with object method which known as private members.

- Python does not have private keyword unlike oop language but encapsulation can be take place.
- A class variable that should not directly access must be prefixed with __(Double underscore).

- Using encapsulation we can restrict access to methods and variable , which prevent data from direct modification.
- In python we can represent private attributes using____(Double Underscore)

  - **Example :-**

```
Class
abc(object): Def________________init_(self):
        Self.a   =   123
        Self.b=123
        Self__c=123
Return

 Obj    =    abc()
 Print(obj.a)
 Print(obj.b)
 Print(obj.__c)
```

- **When above code is execute the following output should be display:-123**

123
Trackback
Attribute Error : 'abc' object has no attribute '___c'.

- **Encapsulation prevent accessing of data accidently not intencially.**
- **In Encapsulation :-**

-Public Method – Accessible from anywhere.

-Private Method – Accessible only with its own class that start with __(double underscore).

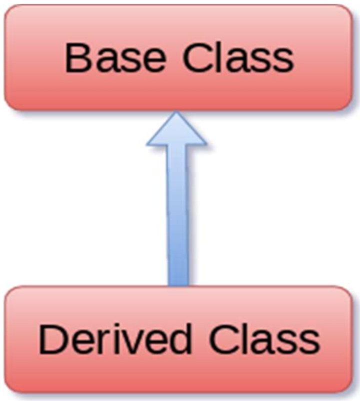
-Public Variable – Accessible from anywhere.

-Private Variable – Accessible only with its own class that starts with ___(double underscore).

1 Word Question – Answer

SR.NO .	QUESTION	ANSWER
1	Using___________we can restrict access to methods and variable.	encapsulation
2	In python we can represent private attributes using_________________.	__(double Underscore

Q-5 Explain Inheritance with example



Detail :-

- Inheritance is a feature of object oriented programming.
- It is use to specify that one class will get all the features of other class.
- It is a powerful feature that provide facility to create new class with few modification to existing class.
- The Main class from which child class inherit the property is called parent class or base class.
- The class that get all the features of parent class or base class is called Child class or derived class.
- The main purpose of inheritance is re-usability.

- **Syntax :-**

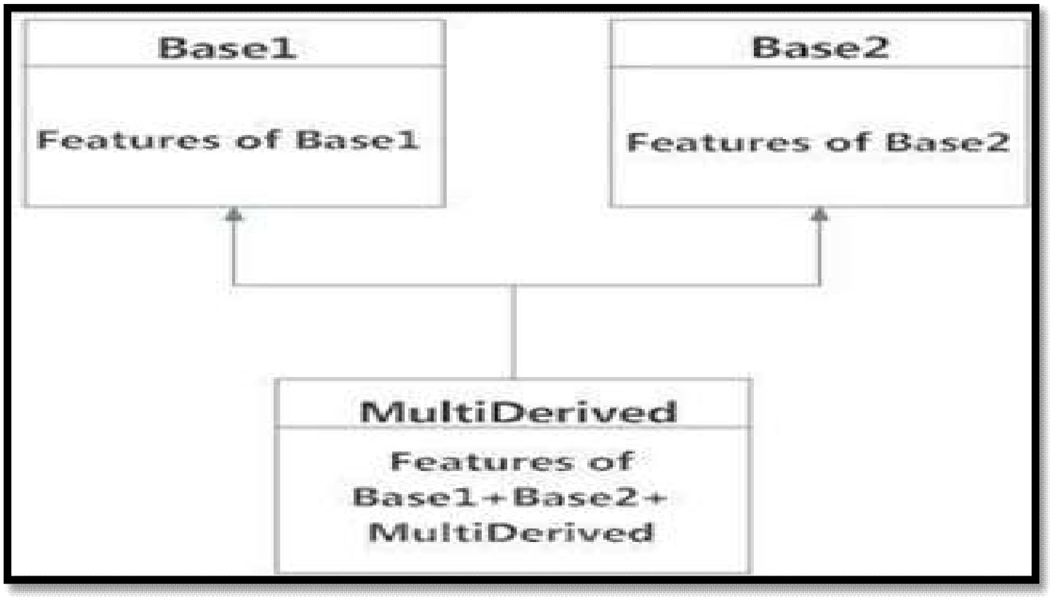
```
Class derivedclass (baseclass name):

   <statement 1>

   <statement 2>
          ……
          ……

   <statement N>
```

- **Example :-**

```
Class animal :
        Def eat(self):
                Print    ('eating……')
Class dog(animal):
        Def break(self):
                Print('barking……')
#calling
        d.dog()
        d.eat()
        d.bark()
```

✓ Multiple inheritance:-

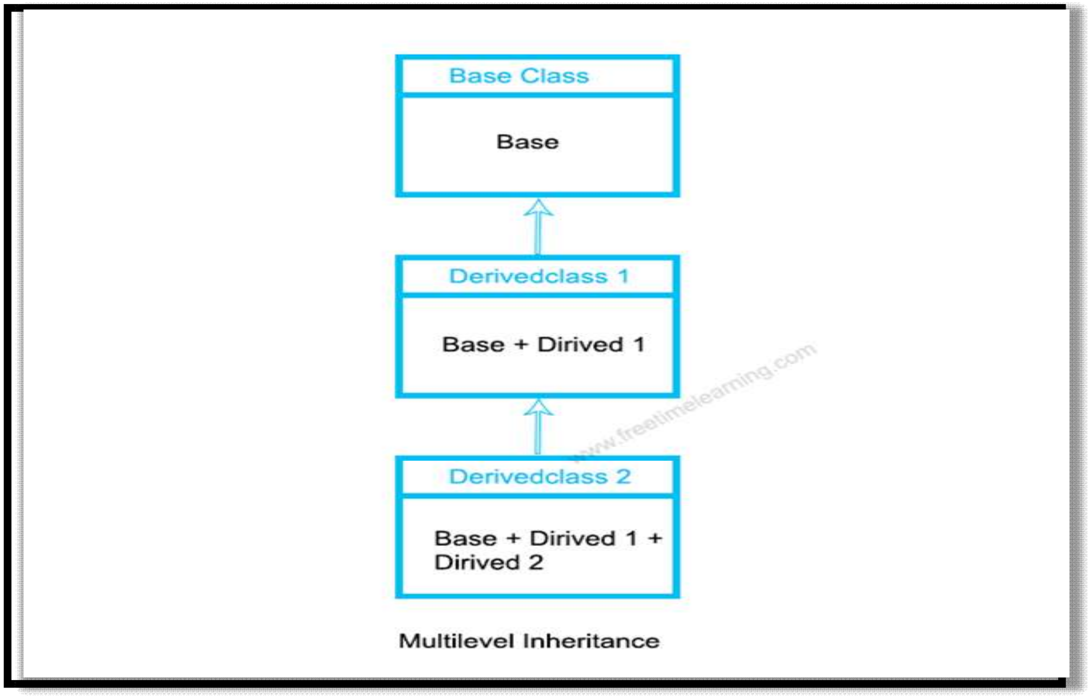
- Python support multiple inheritance which allow us to inherit multiple parent classes.
- We can derive child class from more than one parent or base classes.
- Python provides us the flexibility to inherit multiple base classes in the child class.



✓ Multilevel inheritance:-

- In multilevel inheritance , we can inherit derived class from another derived class.

- Multi-level inheritance is archived when a derived class inherits another derived class.
- There is no limit on the number of levels up to which, the multi-level inheritance is archived in python.



Multilevel Inheritance

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	The main purpose of inheritance is ______________.	Re-usability
2	In_____________inheritance ,We can derive child class from more than one parent or base classes.	__(double Underscore)

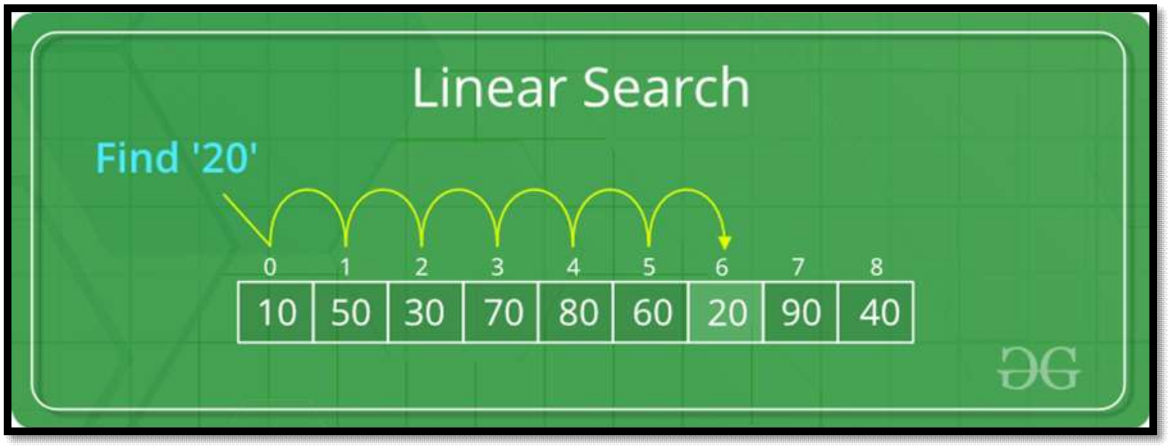
Q-5 Explain Searching Algorithm.

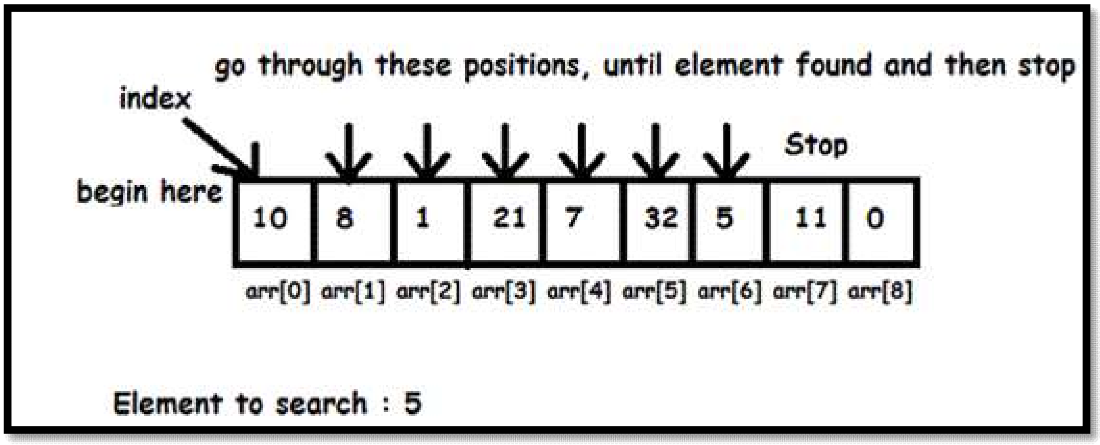
Detail :-

- Searching is very necessary when you store the data in different data structure.

- The main purpose of searching is to go for every elements in the data structure and match it with the value , you are searching for.
- There are two types of searching algorithm available in python:-
    - Linear search
    - Interpolation search

✓ **Linear search:-**




go through these positions, until element found and then stop
index
Stop
begin here
10 8 1 21 7 32 5 11 0
arr[0] arr[1] arr[2] arr[3] arr[4] arr[5] arr[6] arr[7] arr[8]
Element to search : 5

- In this type of search , the sequential order must be follow by all the elements.
- Every list item is checked , if match is found then the particular list item willbe return.

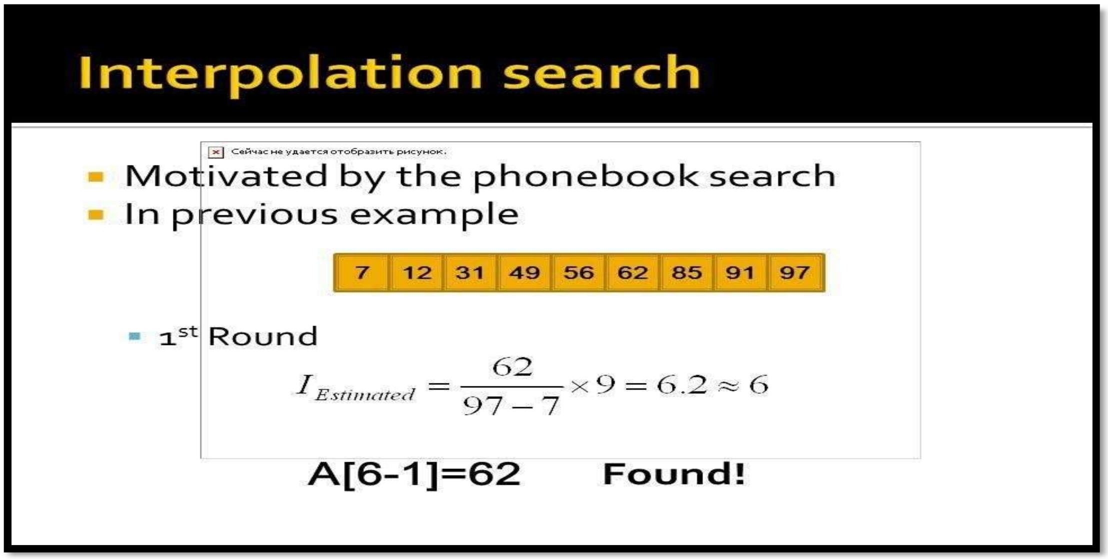
- Otherwise the search continue till the end of data structure.
  - **<u>Example :-</u>**

```
Def linear_search(values , search_for):

        Search_at      =0
        Search_Res=false
#match value with element
        While (search_at < len(values) and search_res is false):
        If (value[search_at]==search_for ):
                        search_res = true
                else:
                        search_at=search_at  +  1
                        return search_res
        l = [64,34,25, 12,22,10,90]
                print(linear_search(l,12))
                print(linear_search(l,91))
```

  - **Output :-**

```
True
False
```

✓ **<u>Interpolationsearch</u>**



Interpolation search

Сейчас не удается отобразить рисунок.

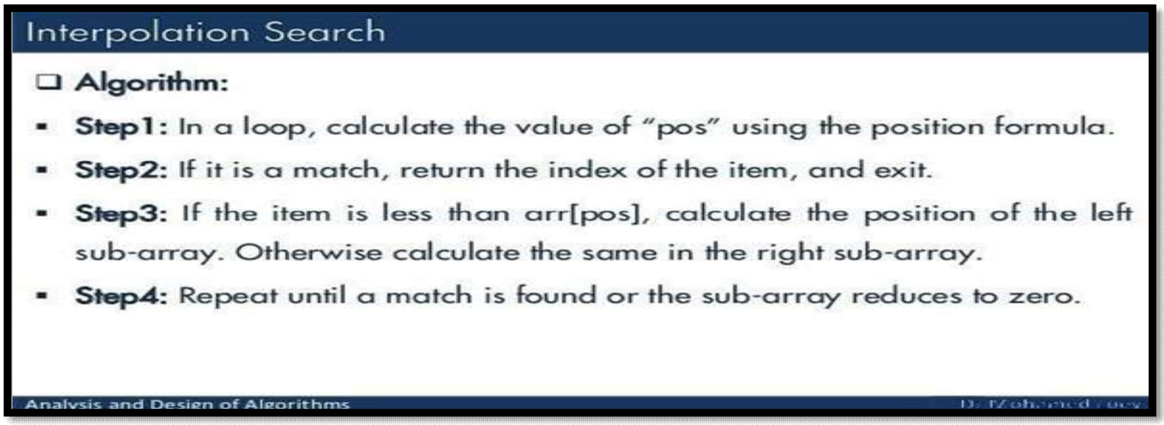
- Motivated by the phonebook search
- In previous example

7	12	31	49	56	62	85	91	97

- 1st Round

$$I_{Estimated} = \frac{62}{97-7} \times 9 = 6.2 \approx 6$$

A[6-1]=62 Found!



Interpolation Search

❑ **Algorithm:**

- **Step1:** In a loop, calculate the value of "pos" using the position formula.
- **Step2:** If it is a match, return the index of the item, and exit.
- **Step3:** If the item is less than arr[pos], calculate the position of the left sub-array. Otherwise calculate the same in the right sub-array.
- **Step4:** Repeat until a match is found or the sub-array reduces to zero.

Analysis and Design of Algorithms

- This search technique works on particular position of needed value.
- For this searching data collection must be in sorted order.
- In this search there may be probe position , it is the position of middle most list item of the collection.
- If middle item is > greater than search item then probe position is again calculated.

o **Example :-**

```
Def  inter_search(values  ,  x):
        idx =0
        idxn = (len(values)-1)

While  (idx  <  =  idxn  and  x>=values[idx]  and  x<=value[idxn]):
#find mid point
        Mid   =   idx
#compare value
                If (values[mid]==x):
                        Return "found" + str(x) + "at index" + str(mid)
                        If values [mid] < x:
                        idx = mid+1
                        return  "search  element  not  in  the
        list"l = [2,6,11,19,27,31,45,121]
                print(inter_search(l,2))
```

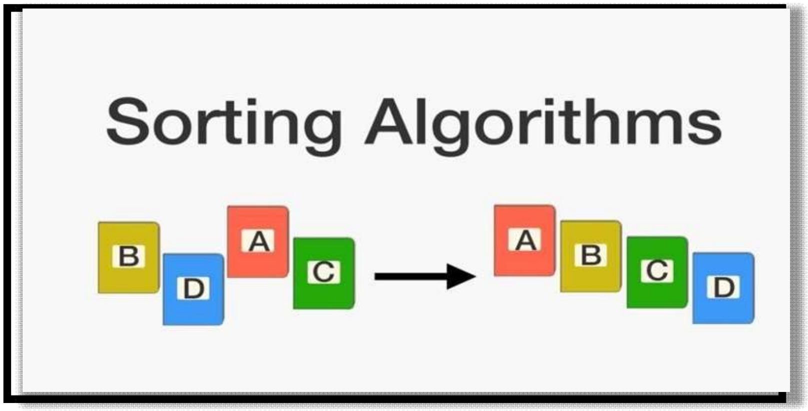
- **Output :-**

  Found 2 at index 0

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	__________ _ &___________are the types of searching algorithm in python.	Linear Interpolation
2	In_____________type of search , the sequential order must be follow by all the elements.	Linear
3	In____________ search there may be probe position , which return middle most list item of the collection.	interpolation

Q -7 Explain Sorting Algorithm.

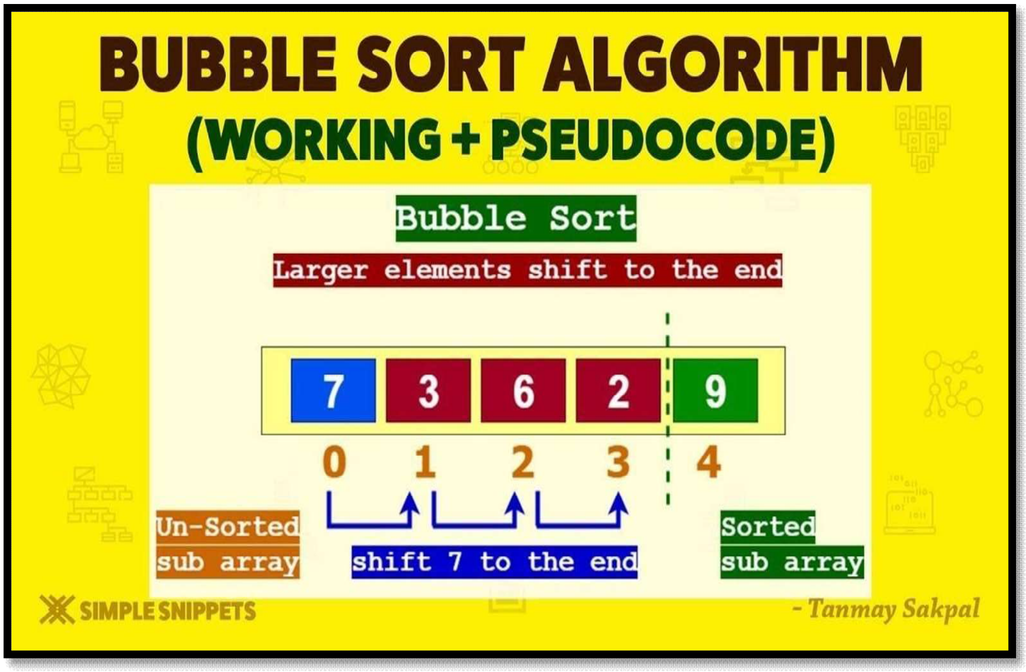


Detail :-

- Sorting is used to arrange data in particular format.
- Sorting algorithm specify a way to arrange data in particular order.
- Sorting support following implementation in python.
  - Bubble Sort
  - Merge Sort

- Selection Sort
- Shell Sort
- Intertion Sort

✓ **Bubble Sort:-**



➢ It is comparison based algorithm in which each pair of elements will be compare and the elements are swapped if they are not in the order.

- **Example :-**

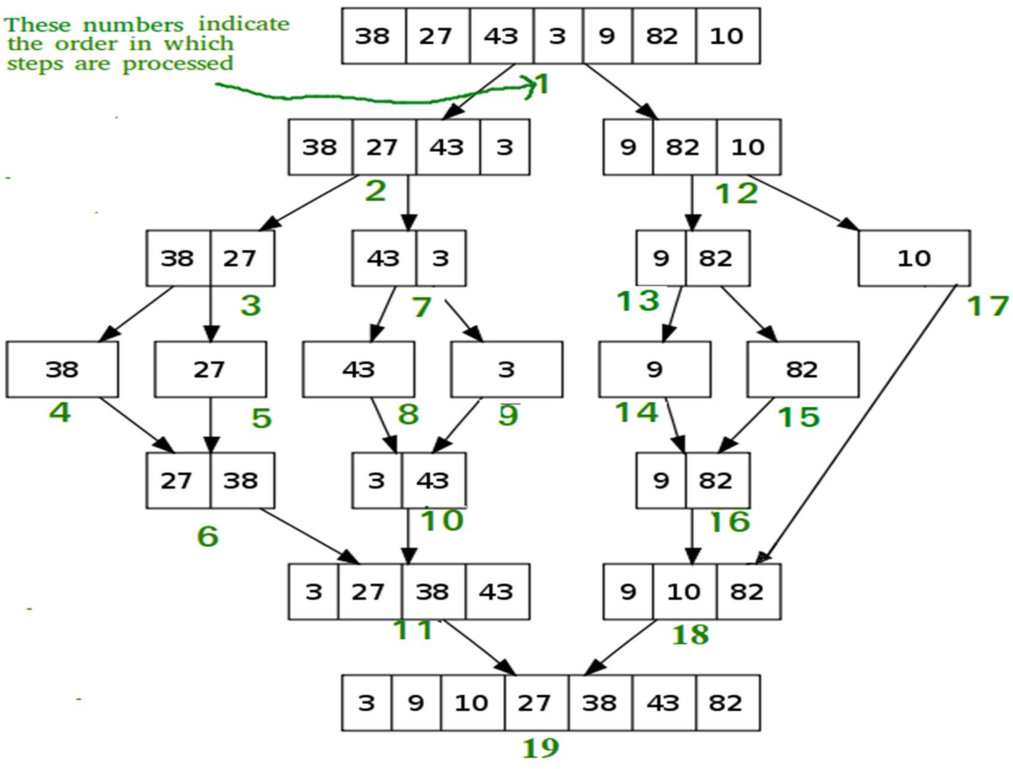
```
Def bubblesort(list):
        For item_num in range(len(list1)-1,0,-1):
        If list[idx] > list[idx+1]:
                Temp = list[idx]

                List[idx]=   list[idx+1]
                List[idx+1] = temp
```

```
List=[19,2,31,45,6,11,121,27]
Bubblesort(list)
Print(list)
```

o/p :- 2,6,11,19,27,31,45,121

✓ **Merge Sort:-**

- Merge sort first devide the array into equal parts then combine them into sort manner.



- Merge Sort is divide and conquer algorithm.
- It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves.

o **Example :-**

```
def          mergeSort(nlist):
  print("Splitting ",nlist)
if len(nlist)>1:
    mid = len(nlist)//2
    lefthalf = nlist[:mid]
    righthalf = nlist[mid:]

    mergeSort(lefthalf)
    mergeSort(righthalf)
    i=j=k=0

    while i < len(lefthalf) and j < len(righthalf):
    if lefthalf[i] < righthalf[j]:
        nlist[k]=lefthalf[i]
        i=i+1
      else:
        nlist[k]=righthalf[j]
        j=j+1
      k=k+1

    while   i   <   len(lefthalf):
      nlist[k]=lefthalf[i]
      i=i+1
      k=k+1


    while    j    <    len(righthalf):
      nlist[k]=righthalf[j]
     j=j+1
     k=k+1
  print("Merging  ",nlist)
```

✓ **Selection Sort:-**

- In selection sort we can start sorting of elements by finding minimum value.
- Then we can move it to the sortest list.
- We can repeat the process for each of the remaining element from unsorted
  List.

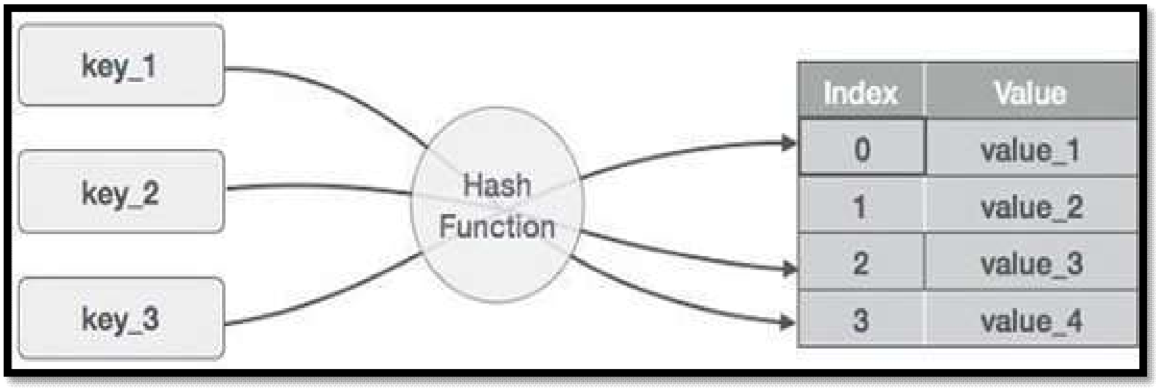
o **Example :-**

```
def   selectisn_sort(input_list):
        for idx in range (len(input_list)):
        min_idx = idx
        for j in range(idx+1 , len(input_list)):
                if          input_list[min_idx]<input_list[1]:
        input_list[min_idx] = input_list[idx]
         l=[19,2,31,45,30,11,121,27]
        selection_sort(l)
        print(l)
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	__________algorithm specify a way to arrange data in particular order.	sorting
2	___________ is divide and conquer algorithm.	Merge sort
3	In_______________we can start sorting of elements by finding minimum value.	Selection sort

Q-8 Explain Hash Table with Example.



Index	Value
0	value_1
1	value_2
2	value_3
3	value_4

D etail :-

- Hash tables are type of data structure in which index value of data isgenerated from hash table.
- It makes data accessing faster because index value behave as a key.
- In other words hash table store key-value pair but key will be generate dfrom hash function.
- In hashtable key-value pair become the index of array which store thevalue.
- In python dictionary data type represent implementation of hash table.
- The key of dictionary are known as hash table , which generates uniqu e result for each unique value supplied to the hash function.
- The order of data elements in a dictionary is not fixed.

✓ Performing Operations on Hash tables using Dictionaries:

- There are a number of operations that can be performed on has tables in Python through dictionaries such as:
  - Accessing Values
  - Updating Values
  - Deleting Element

Using key values:

✓ Accessing Values:

- The values of a dictionary can be accessed in many ways such as:

- Using key values
- Using Functions

- Dictionary values can be accessed using the key values as follows:

```
my_dict={'Dave':  '001',  'Ava':  '002',  'Joe':  '003'}my_dict['Dave']
```

- **EXAMPLE:**

OUTPUT: ‘001′

✓ Using functions:

- There are a number of built-in functions that can be used such as get(), keys(), values(), etc.

EXAMPLE:

```
my_dict={'Dave' : '001' , 'Ava': '002' , 'Joe':'003'}
print(my_dict.keys())print(my_dict.values())
print(my_dict.get('Dave'))
```

OUTPUT:

```
dict_keys([‘Dave’, ‘Ava’, ‘Joe’])
```

✓ Updating Values:

- Dictionaries are mutable data type and therefore, you can update them as and when required.
- For example, if I want to change the ID of the employee named Dave from ‘001’ to ‘004’ and if I want to add another key-value pair to my dictionary, Ican do as follows:

o **EXAMPLE:**

```
my_dict={'Dave' : '001' , 'Ava': '002' , 'Joe':'003'}
my_dict['Dave'] = '004'                  #Updating the value ofDave
my_dict['Chris'] = '005' #adding a key-valuepair
print(my_dict)
```

OUTPUT: {'Dave': '004', 'Ava': '002', 'Joe': '003', 'Chris': '005'}

✓ Deleting items from a dictionary:

➢ There a number of functions that allow you to delete items from a dictionary such as *del(), pop(), popitem(), clear(),* etc. For example:

o **EXAMPLE:**

```
my_dict={'Dave': '004', 'Ava': '002', 'Joe':
'003', 'Chris': '005'}
del my_dict['Dave'] #removes key-value pair of'Dave'
my_dict.pop('Ava')            #removes   the   value   of   'Ava'
my_dict.popitem()             #removes the last inserted item
print(my_dict)
```

OUTPUT: {'Joe': '003'}

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	__________are type of data structure in which index value of data is generated from hash table.	Hash tables
2	In hash table , the index value always behave as a_________.	key
3	In hash table ,key will be generated from __________.	hash()

SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.
(AFFILIATED TO SAURASHTRA UNIVERSITY)

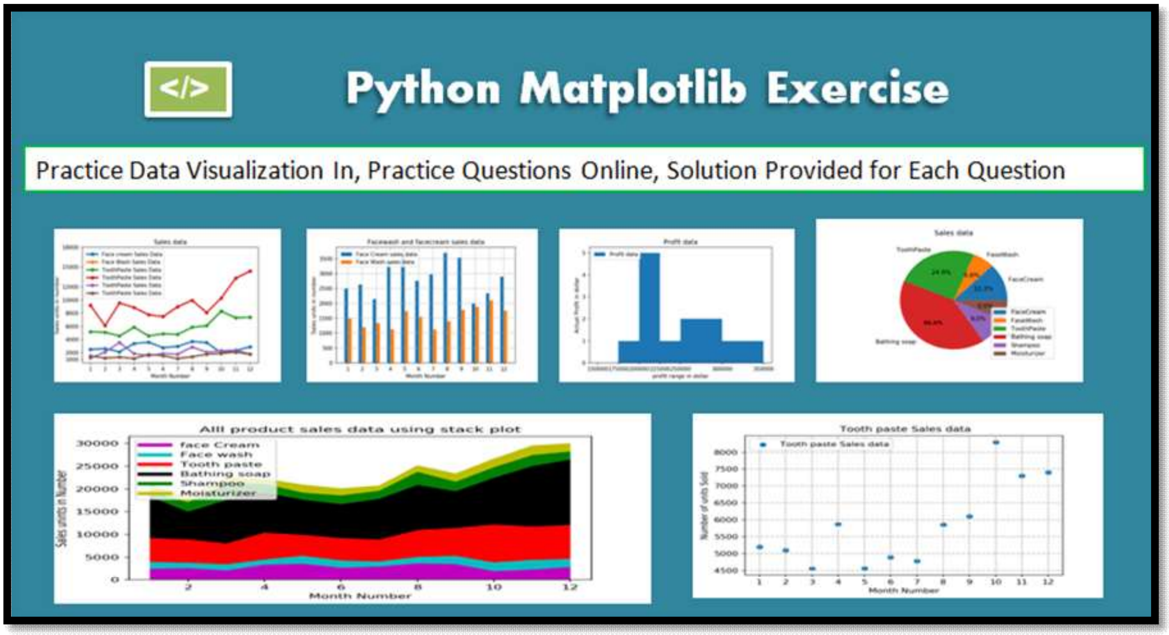
2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001

CHAPTER-3
Plotting Using PyLab

- **Plotting using PyLab**
- **Plotting Mortgages**
- **Extended Example**
- **Fibonacci Sequence Revisited**
- **Dynamic Programming**
- **0/1 Knapsack Algorithm**
- **Dynamic Programming with Divide and Conquer**

Q-1 Explain How to plotting using PyLab.



</>

Python Matplotlib Exercise

Practice Data Visualization In, Practice Questions Online, Solution Provided for Each Question

Detail :-

- □ PyLab is a module inside MATPLOTLIB library.
- □ MATPLOTLIB was developed by John . D. Hunter in 2003.
- □ MATPLOTLIB has it’s roots in MATLAB which need to decide PyLab.
- □ MATLAB support many built – in function for users to develop the code.
- □ It become easy for MATLAB user who don’t want to use import statement.
- □ PyLab having number of functions and classes for generating the drawings.
- □ MATPLOTLIB in python having 1000 lines of code to create quality graphics.

✓ MATPLOTLIB :-

MATPLOTLIB is free python library for generating plots in 1D ,
2D Graphics.
MATPLOTLIB is easy to use interface for PyLab modules.

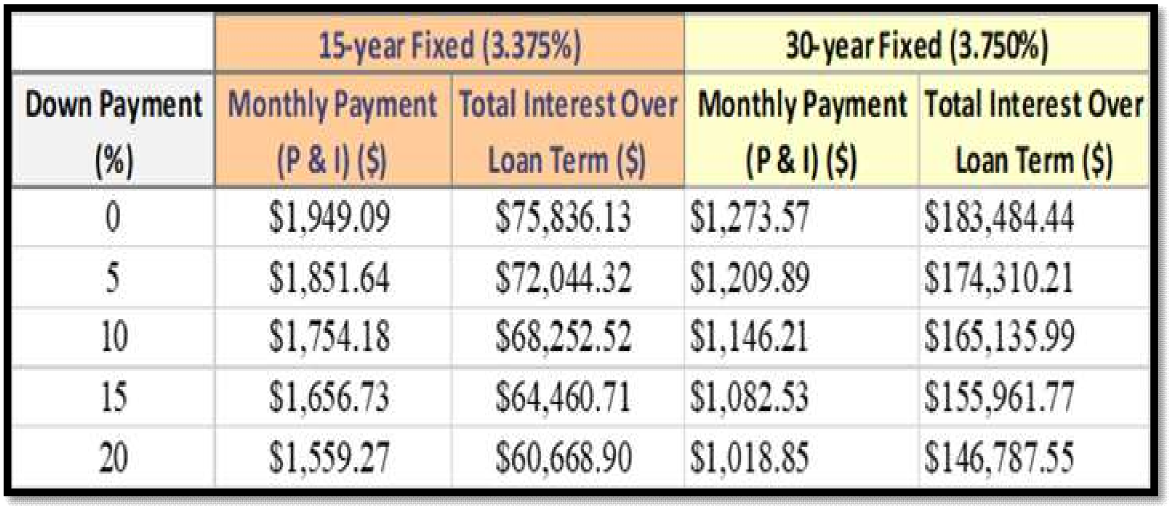
Example :-

```
Import matplotlib.pyplot as plt
Import numpy as np
        X = np.linspace(0,10,100)
         Plt.plot(x   ,   x.Label   =   “linear”)
         Plt.legend()
         Plt.show()
```

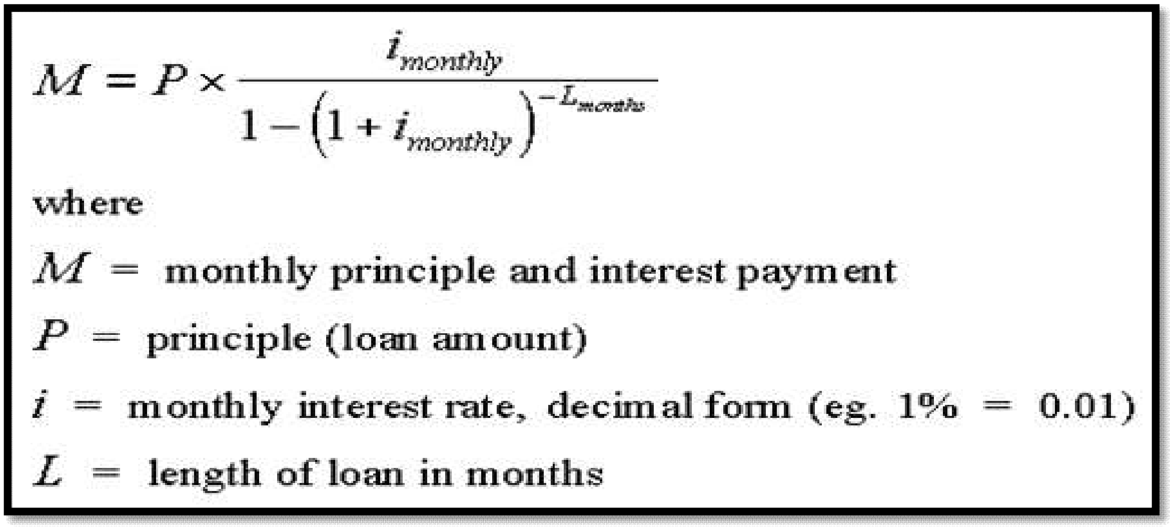
1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	MATPLOTLIB was developed by ______________in 2003.	John . D. Hunter
2	_____________is a module inside MATPLOTLIB library.	PyLab
3	__________ is free python library for generating plots in 1D , 2D Graphics.	MATPLOTLIB
4	______________ support many built – in function for users to develop the code.	MATLAB

Q-2 Explain Plotting Mortgage in brief.



	15-year Fixed (3.375%)		30-year Fixed (3.750%)	
Down Payment (%)	Monthly Payment (P & I) ($)	Total Interest Over Loan Term ($)	Monthly Payment (P & I) ($)	Total Interest Over Loan Term ($)
0	$1,949.09	$75,836.13	$1,273.57	$183,484.44
5	$1,851.64	$72,044.32	$1,209.89	$174,310.21
10	$1,754.18	$68,252.52	$1,146.21	$165,135.99
15	$1,656.73	$64,460.71	$1,082.53	$155,961.77
20	$1,559.27	$60,668.90	$1,018.85	$146,787.55



$$M = P \times \frac{i_{monthly}}{1-\left(1+i_{monthly}\right)^{-L_{months}}}$$

where

M = monthly principle and interest payment

P = principle (loan amount)

i = monthly interest rate, decimal form (eg. 1% = 0.01)

L = length of loan in months

Detail :-

- Mortgage is a simple calculation to find out or understand true cost of any loan or interest.
- To install mortgage you have to write following command at CMD prompt.
  - **Pip install mortgage**
- The above package can provide easy way to compare different mortgages.
- It you are consider in mortgage loan , you should understand all the dtails about how principal and interest will be calculated.
- It will be also easy to find out monthly payment system for mortgage loan which include following :
  - HOA :- [Home Owner Association fee]
  - PMI :- [Private Mortgage Insurance fee]
  - Home Owner Insurance
  - Taxes
- Here are some common tricks to find out mortgage loan calculation :-
  1. **Provide sales price of house.**
  2. **Enter down payment.**

3.Calculate loan amount.
4.Enter loan terms in years.
5.Enter interest rate.
6.Calculate monthly payment.
7.Calculate final balance and monthly interest.

Example :-

Calculate simple mortgage

```
From mortgage import loan
l = Loan(p=2,00,000 , i=0.06 , term  = 30)
l.summarize
>>>original balance : 2,00,000
>>>interest rate : 0.06%
>>>terms : 30 years
>>>monthly payment :
```

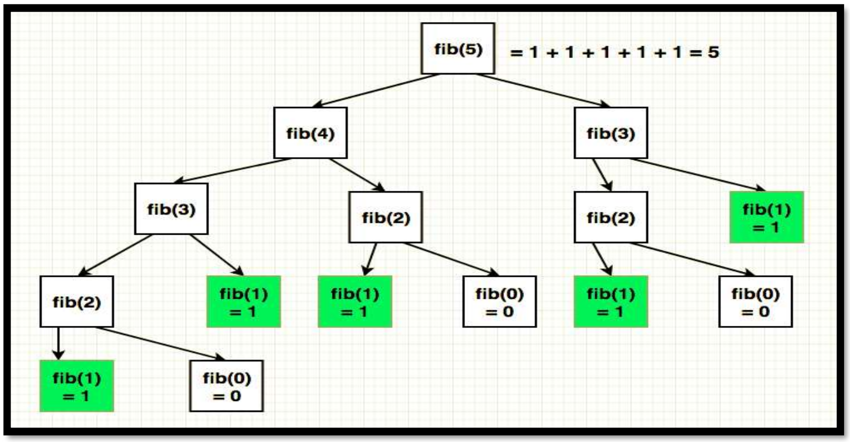
1 Word Question – Answer

sr. no		
.	____________is a simple cal1culation to find out or understand true cost of any loan or interest.	Mortgage
	To install mortgage you have to write ____________command.	Pip install mortgage

Q-3 Explain Fibonacci sequence revisited with example.

Detail :-

- One of the most common type of math based technical challenges are ones that deal with Fibonacci sequence.
- Each new term in the Fibonacci sequence is generated by adding the previous two terms.
- For example , starting with 1 and 2 ,the first 10 numbers in the sequence would be :
    - **1,2,3,5,8,13,21,34,55,89**
- One of the favourite challenge that deals with Fibonacci sequence is one that asks for index value of some high number in the sequence.
- It might be good idea to record the value returned by the first call , and then look it up rather than compute it each time it is needed.This is called “memorization”.
- “Memorization” is key idea behind any dynamic programming.
- Normally , easy way to go about doing something like would be to put all the numbers in array and then cycle them with for loop.
- First it requires two different functions , one function to generate Fibonacci sequence and second function to cycle through all the numbers we have generated.
- Let’s see ,implementation of Fibonacci sequence by following figure.



As above figure , look at tree of calls associated with the invocation fib(6). Notice that we are computing same values over & over again.

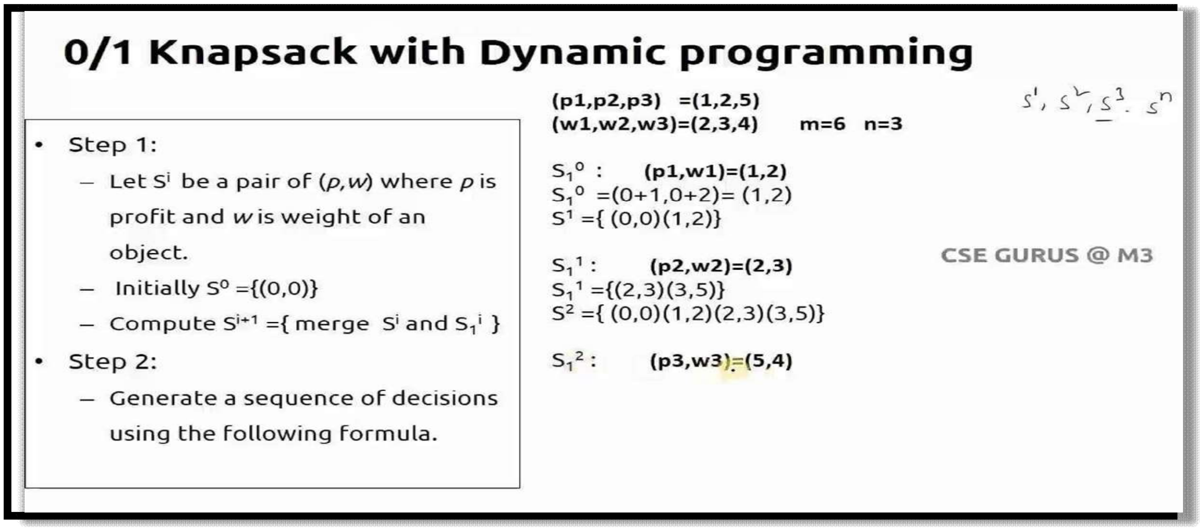
o**Example :-**

```
    Def
    fib_seq(n)
    A=0
    B=1
    If n==1:
       Print(a)
    Elif n==2:
       Print(a,b)
    Else:
       Print(a,b,end=”      “)
    For I in range(n-2):
       c=a+b
       a=b
       b=c
       print(c , end=” “)
#calling
    fib_seq(10)
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	Each new term in the Fibonacci sequence is generated by adding the previous____ terms.	two
2	__________is key idea behind any dynamic programming.	Memorization

Q-4 Explain 0/1 knapsack algorithm.



0/1 Knapsack with Dynamic programming

- Step 1:
  - Let S^i be a pair of (p,w) where p is profit and w is weight of an object.
  - Initially $S^0 = \{(0,0)\}$
  - Compute $S^{i+1} = \{$ merge S^i and S_1^i $\}$
- Step 2:
  - Generate a sequence of decisions using the following formula.

$(p1,p2,p3) = (1,2,5)$
$(w1,w2,w3) = (2,3,4)$ $m=6$ $n=3$

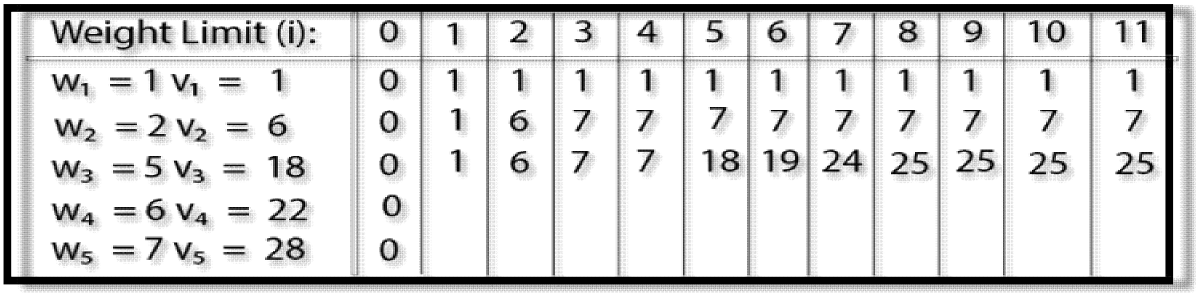
S_1^0 : $(p1,w1)=(1,2)$
$S_1^0 = (0+1,0+2) = (1,2)$
$S^1 = \{ (0,0)(1,2)\}$

S_1^1 : $(p2,w2)=(2,3)$
$S_1^1 = \{(2,3)(3,5)\}$
$S^2 = \{ (0,0)(1,2)(2,3)(3,5)\}$

S_1^2 : $(p3,w3)=(5,4)$

$s^1, s^2, s^3 . s^n$

CSE GURUS @ M3



Weight Limit (i):	0	1	2	3	4	5	6	7	8	9	10	11
$w_1 = 1$ $v_1 = 1$	0	1	1	1	1	1	1	1	1	1	1	1
$w_2 = 2$ $v_2 = 6$	0	1	6	7	7	7	7	7	7	7	7	7
$w_3 = 5$ $v_3 = 18$	0	1	6	7	7	18	19	24	25	25	25	25
$w_4 = 6$ $v_4 = 22$	0											
$w_5 = 7$ $v_5 = 28$	0											

Detail :-

- In 0-1 knapsack , items can not be broken.
- It means if 1 than cover all the elements and if 0 than leave all the elements.
- This is the main reason behind calling it as 0-1 knapsack.
- In case of 0-1 knapsack , the value of x_i can be either 0 or 1 where other elements remains same.
- Let's consider the capacity of the knapsack is w=25 and the items as shown in the following table.

ITEM	A	B	C	D
PROFIT	24	18	18	10
WEIGHT	24	10	10	7

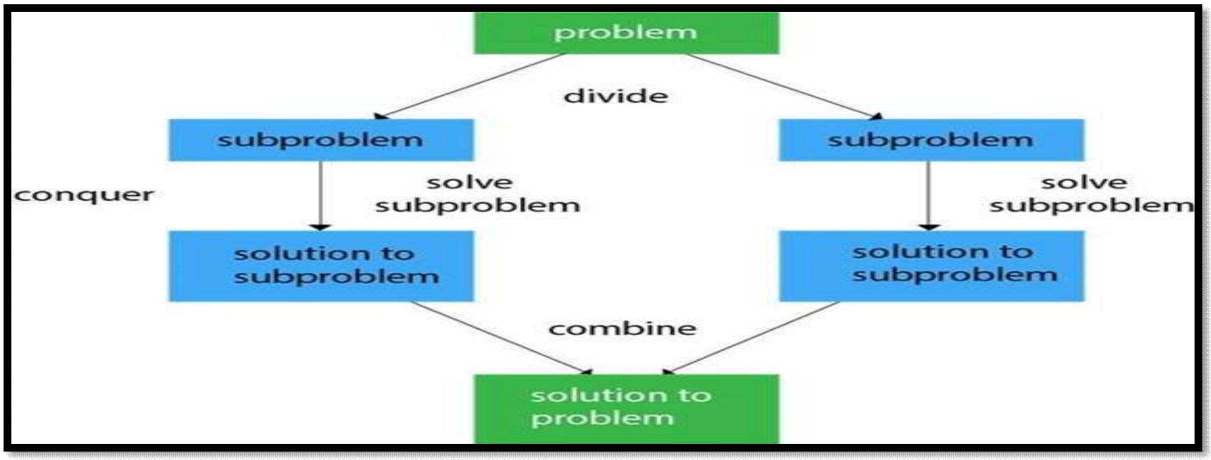
- Here , profit per unit weight = **(p_i/w_i).**
- First item A will be selected as it will contribute maximum profit among allthe elements.
- After selecting item A , no more item will be selected.
- Here , for this given set of items total profit is 24.
- The optimal solution can be achieved by selecting items , B and C , where total profit is 18+18=36.
- In this example , the items are selected based on ratio **(p_i/w_i).**
- Let's us consider capacity of knapsack is w=60 and the items are as shown in following table.

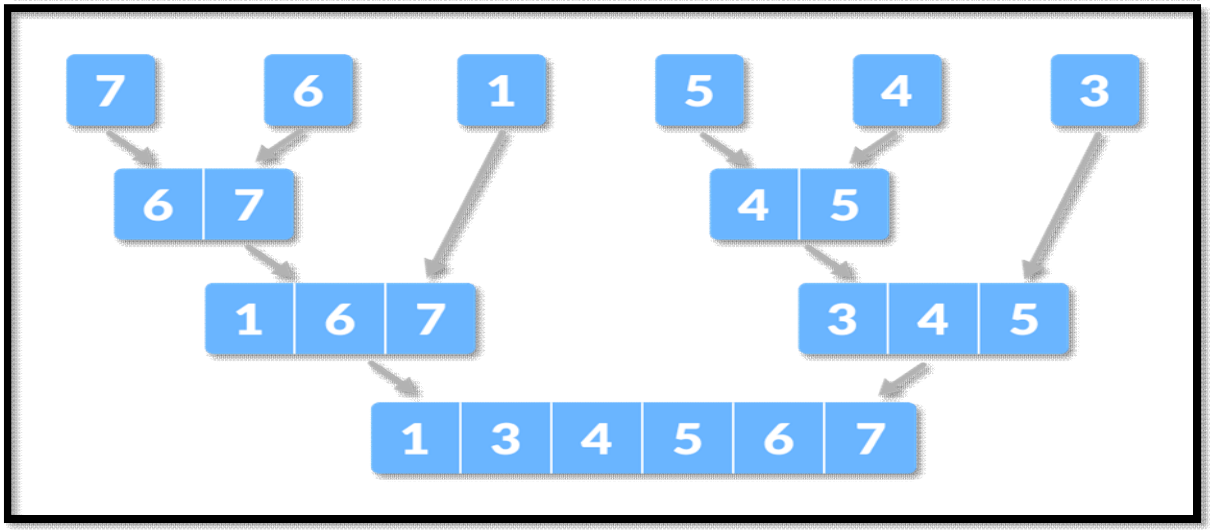
ITEM	A	B	C
PRICE	100	280	120
WEIGHT	10	40	20
RATIO	10	7	8

- First item A is selected , then next item B is select.
- Here , total profit is 100 + 280 =380.
- The optimal solution of this instance can be achieved by selecting items B and C , where total profit is 280 + 120 = 400.
- 0/1 knapsack alogorithm takes following inputs :
  - The maximum weight = W.
  - The number of items = n.
  - The two sequences ….
    - Value V = <v1,v2,……,vn>
    - Weight W = <w1,w2,…..,wn>

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	In__________, items can not be broken.	0-1 knapsack
2	In 0-1 knapsake algorithm,_______means cover all the elements and_________means leave all the elements.	1 0





Q-5 Explain dynamic programming with divide & conquer algorithm.

DIVIDE & CONQUER ALGORITHM

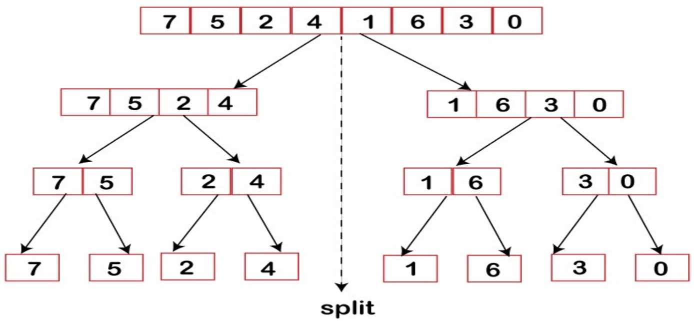
Detail :-

- Devide & conquer is the process of breaking down problem into smaller parts.
- Break a problem into subprograms that are similar with original problems.
- Recursively , solves the sub problems , and finally combines the solutions to the sub programs to solve the original problem.
- Divide & conquer is an alogorithm paradigm.
- A typical divide & conquer algorithm solve a problem using following 3steps:
  - **DIVIDE (Break) :-**
    - It breaks the given problem into sub – problems of same type.
    - Ths step involves breaking the problem into smaller sub – problems.
    - At this stage , sub- problem should represent a part of original problem.
  - **CONQUER(Solve) :-**
    - It include process of recursively solve the sub – problems.
    - This step receives a lot of smaller sub – problem to be solved.
    - Generally , at this level , the problems are considered as “solved”.
  - **COMBINE (Merge):-**
    - It combines the appropriate answers as well as results.
    - When smaller sub – problems are solved , this stage recursively combines them until they found solution of original problem.

- **Example :-**

```
def     bsearch(list     ,val):
        list_size = len(list) -1
        idx0 = 0
        idxn = list_size #find middle most value
        While idx0 < = idxn :
                Midval = (idx0 _ idxn)/2
                If list[midval] == val:
           Return midval
#compare value – middle most value
        If val > list[midval]:
                Idx0 = midval +1
        Else:
                 Idxn = midval – 1
                 If idx0 > idxn:
             Return none
#calling
        l1 = [55,44,2,4,9,8]
           Print (bsearch(l1,4)
```

- A classic example of divide & conquer is merge sort which demonstrated below.
- In merge sort , we divide array into two halves & sort the two halves recursively & then finally merge the sored halves.



SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.
(AFFILIATED TO SAURASHTRA UNIVERSITY)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot - 360001

CHAPTER-4

Network Programming and GUI using Python

- Network Programming:
- Protocol, Sockets,
- Knowing IP Address,
- URL, Reading the Source Code of a Web Page,
- Downloading a Web Page from Internet,
- Downloading an Image from Internet,
- A TCP/IP Server, A TCP/IP Client,
- A UDP Server, A UDP Client,
- File Server, File Client,
- Two-Way Communication between Server and Client,
- Sending a Simple Mail.
- GUI Programming:
- Event-driven programming paradigm;
- creating simple GUI;
- buttons, labels, entry fields, dialogs;
  widget attributes - sizes, fonts, colors ,layouts,nested frames

Q-1 What is Socket in network programming?

Detail :-

- Python provides two levels of access to network services.
- At a low level, you can access the basic socket support in the underlying operating system, which allows you to implement clients and servers for both connection-oriented and connectionless protocols.
- Python also has libraries that provide higher-level access to specific application-level network protocols, such as FTP, HTTP, and so on.

What is Sockets?

- Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.
- Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The socket library provides specific classes for handling the common transports as well as a generic interface for handling threst.

The *socket* Module

- To create a socket, you must use the socket.socket() function available in socket module, which has the general syntax −
- **s = socket.socket (socket_family, socket_type,**

protocol=0) Server Socket Method

Sr.No.	Method & Description	
1	**s.bind()**	This method binds address (hostname, port number pair) to socket.
2	**s.listen()**	This method sets up and start TCP listener.
3	**s.accept()**	This passively accept TCP client connection, waiting until connection arrives (blocking).

Client Socket Methods

Sr.No.	Method & Description
1	**s.connect()** This method actively initiates TCP server connection.

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	To create Socket , you must use _________module	Socket
2	_____________Method bind address to the socket.	Bind()
3	__________ method will accept the client connection.	Accept()

Q-2 IP Address in python.

- **IP (Internet Protocol)** -Address is the basic fundamental concept of computer networks which provides the address assigning capabilities to a network.
- Python provides *ipaddress* module which is used to validate and categorize the IP address according to their types*(IPv4 or IPv6)*.
- This module is also used for performing wide range of operation like arithmetic, comparison, etc to manipulate the IP addresses.

Detail :

□ **Validating IP Addresses**

- For validating IP addresses python uses *ip_address()* function provided by the *ipaddress* module which raises error if values of IP exceeds the range of the type of the IP address.

□ **IPv4 :** It is a 32-bit number typically written in decimal digits formatted as four 8-bit numbers separated by dots, is used to identify the network interface of a machine.

□ The *ip_address()* function throws an error if the range value exceeds from 0 to 255.

```
# Import module
import ipaddress

# Example of valid IPv4 address
print (ipaddress.ip_address(u'175.198.42.211'))

# Invalid IPv4 address raises error
print (ipaddress.ip_address(u'175.198.42.270'))
```

Output :
175.198.42.211

➢ **IPv6 :** It is represented by eight groups of four hexadecimal digits separated by colons, where each group represents 16 bits, two octets also known as hextet.

➢ The *ip_address()* function throws an error if the range value exceeds from 0 to FFFF.

```
#    Import    module
import ipaddress

# Example of valid IPv6 address
print
(ipaddress.ip_address(u'2001:0db8:85a3:2bfe:070d:8a2e:0370:7334')
```

➢

```
# Invalid IPv6 address raises
errorprint
(ipaddress.ip_address(u'2001:0db8:85a3:0ff0:00000:8a2e:0370:7334'))
```

Output :

```
2001:db8:85a3:2bfe:70d:8a2e:370:7334
```

- **Operations on IP Address-**

Various operations like arithmetic, comparison, type, etc can be performed on the IP addresses with the help of *ipaddress* module. Some operations are listed below:

- **Type Check operation:** The *type()* method takes various formats of IP addresses as input and recognizes whether it is IPv4 or IPv6 address, indicating the category of the IP address.

```
# Import module
import ipaddress

# IPv4 address
print(type(ipaddress.ip_address(u'175.198.42.211')))
print(type(ipaddress.ip_address(u'5.0.0.1')))

# IPv6 address
print(type(ipaddress.ip_address(u'2001:0db8:85a3:2bfe:070d:8a2e:0
370:7334')))
print(type(ipaddress.ip_address(u'0000:f0f0::7b8a:ffff')))
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	____________ module support various operations performed on IP address.	ipaddress
2	______________is represented by eight groups of four hexadecimal digits	IPv6
3	IP stands for __ __	Internet Protocol

Q-3Explain Protocol in detail.[2-3 marks]

Detail :-

- The Internet Protocol is designed to implement a uniform system of addresses on all of the Internet-connected computers everywhere and to make it possible for packets to travel from one end of the Internet to the other.

- A program like the web browser should be able to connect to a host anywhere without ever knowing which maze of network devices each packet is traversing on its journey.
- There are various categories of internet protocols.
- These protocols are created to serve the needs of different types of data communication between different computers in the internet.
- Python has several modules to handle each of these communication scenarios.
- The methods and functions in these modules can do the simplest job of just validating a URL or also the complex job of handling the cookies and sessions.
- In this chapter we will look at the most prominent python modules used for internet protocols.

Protocol	Python Module Name	Descriptio n
TTP	urllib.request	Opening the HTTP URL
HTTP	urllib.response	Create a reponse object for a url request
HTTP	urllib.parse	To break Uniform Resource Locator (URL) strings up in components like (addressing scheme, network location, path etc.),
HTTP	urllib.robotparser	It finds out whether or not a particular user agent can fetch a URL on the Web site that published the robots.txt file.

FTP	ftplib	implements the client side of the FTP protocol. You can use this to write Python programs that perform a variety of automated FTP jobs, such as mirroring other FTP servers.
POP	poplib	This module defines a class, POP3, which encapsulates a connection to a POP3 server to read messages from a email server
IMAP	imaplib	This module defines three classes, IMAP4, IMAP4_SSL and IMAP4_stream, which encapsulate a connection to an IMAP4 server to read emails.
SMTP	smtplib	The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP listner deamon.
Telnet	telnet	This module provides a Telnet class that implements the Telnet protocol to access a server thorugh teleent.

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER

1	__________ is designed to implement a uniform system of addresses	Internet protocol
2	_____________Protocol can be used to open HTTP url.	HTTP
3	__________ protocol helps to support different file operations	FTP

Q-4What is URL ? How to read source code of webpage.

Detail :

- URL stands for Uniform resource locator.
- The requests module can help us build the URLS and manipulate the URLvalue dynamically.
- Any sub-directory of the URL can be fetched programmatically and then some part of it can be substituted with new values to build new URLs.

What is urllib?

- urllib is a Python module that can be used for opening URLs. It defines functions and classes to help in URL actions.
- With Python you can also access and retrieve data from the internet like XML, HTML, JSON, etc. You can also use Python to work with this data directly.
- In this tutorial we are going to see how we can retrieve data from the web. For example, here we used a guru99 video URL, and we are going to access this video URL using Python as well as print HTML file of this URL.

Steps :-

- Import urllib
- Define your main function
- Declare the variable webUrl
- Then call the urlopen function on the URL lib library
- The URL we are opening is guru99 tutorial on youtube
- Next, we going to print the result code

- Result code is retrieved by calling the getcode function on the webUrlvariable we have created
- We going to convert that to a string, so that it can be concatenated with our string "result code"
- This will be a regular HTTP code "200", indicating http request is processedsuccessfully

"how to get source code of website in python" :-

```
    import requests
    url = input('Webpage to grab source from: ')
    html_output_name = input('Name for html file: ')
req = requests.get(url, 'html.parser')

with open(html_output_name, 'w') as f:
f.write(req.text)
f.close()
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	__________ is a Python module that can be used for opening URLs .	urllib
2	URL stands for ___ __ ____	Uniform Resource Locator
3	_ module can help us build the URLS and manipulate the URL value dynamically.	request

Q-5Explain TCP/IP server and TCP/IP client.

Detail :-

- □ Python is one of the most popular object-oriented scripting languages with a programmer-friendly syntax and a vast developer community.
- □ Here, we'll showcase how to write a TCP server and client in Python and

implement them using classes.

TCP/IP Client and Server

- Sockets can be configured to act as a *server* and listen for incoming messages, or connect to other applications as a *client*. After both ends of a TCP/IP socket are connected, communication is bi-directional.
- Hence, we'll keep our focus only on the workflow and example code of the Python TCP server and client.

- The sample contains the source code for a TCP server and client. For practice, you can extend it to build a small chat system or a local attendance tracking system.

Synchronous socket entities

- **TCPServer class –** It follows the (Internet) TCP protocol that allows continuous streams of data between the server and client.
- The **socketserver** module has more classes to handle sockets.
- Now let's see the example of the Python TCP Server.

Python-TCP-Server.py

```
import socketserver

    class
      Handler_TCPServer(socketserver.BaseRequestHandler):
      """
      The TCP Server class for demonstration.

      Note: We need to implement the Handle method to exchange data
      with TCP client.
      """
      def handle(self):
        # self.request - TCP socket connected to the client
        self.data = self.request.recv(1024).strip()
```

```
 print("{}
sent:".format(self.client_address[0]))
print(self.data)
# just send back ACK for data arrival confirmation
self.request.sendall("ACK from TCP Server".encode())

if __name__ == "_main__":
  HOST, PORT = "localhost", 9999

  # Init the TCP server object, bind it to the localhost on 9999 port
  tcp_server = socketserver.TCPServer((HOST, PORT),
  Handler_TCPServer)

  # Activate the TCP server.
  # To abort the TCP server, press
  Ctrl-C.tcp_server.serve_forever()
```

In the next example code, you’ll see the Python TCP client module code to communicate with the TCP server.

Python-TCP-Client.py

```
      import socket

       host_ip, server_port = "127.0.0.1", 9999
       data = " Hello how are you?\n"

       # Initialize a TCP client socket using SOCK_STREAM
  tcp_client = socket.socket(socket.AF_INET,
  socket.SOCK_STREAM)

       try:
         # Establish connection to TCP server and exchange data
         tcp_client.connect((host_ip, server_port))
         tcp_client.sendall(data.encode())

         # Read data from the TCP server and close the connection
         received = tcp_client.recv(1024)
```

```
finally:
   tcp_client.close()

print ("Bytes Sent:   {}".format(data))
print ("Bytes Received: {}".format(received.decode()))
```

Execution of Python TCP Server and Client modules

- You can run both the server and client in separate Python instances. We recommend that you use Python version 3 for executing the above modules.

- Next, you would first run the server module followed by the client. See below the output of both the client and the server.

- Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25) [MSC v.1900 64 bit (AMD64)] on win32

- Type "copyright", "credits" or "license()" for more information.

- RESTART:
- C:\Users\Techbeamers\AppData\Local\Programs\Python\Python35\Python-TCP- Server.py

- 127.0.0.1 sent: b'Hello how are you?'

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	_________class allows continuous streams of data between the server and client.	TCPserver
2	________________module has more classes to handle sockets.	socketserver
3	____________method can be implement to exchange data with TCP client.	handle

Q-6Explain UDP server and UDP client.

Detail :-

- UDP or user datagram protocol is an alternative protocol to its more common counterpart TCP.
- UDP like TCP is a protocol for packet transfer from 1 host to another, but has some important differences.
- UDP is a connection-less and non-stream oriented protocol.
- It means a UDP server just catches incoming packets from any and many hosts without establishing a reliable pipe kind of connection.

Create UDP sockets

A udp socket is created like this

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

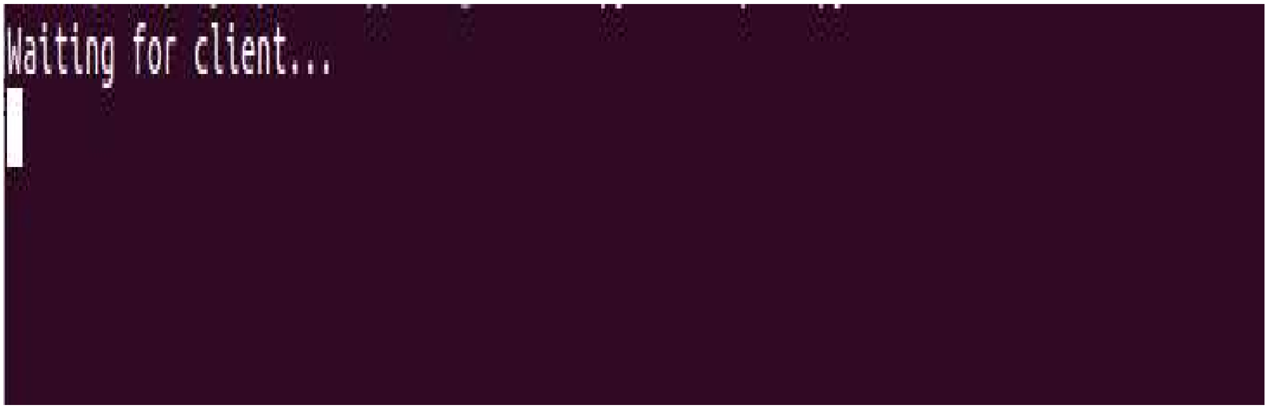
User Datagram Client and Server

- The user datagram protocol (UDP) works differently from TCP/IP.
- Where TCP is a *stream oriented* protocol, ensuring that all of the data is transmitted in the right order, UDP is a *message oriented* protocol.
- UDP does not require a long-lived connection, so setting up a UDP socket is a little simpler.
- On the other hand, UDP messages must fit within a single packet

Simple UDP Server program

```
import socket

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      # For UDP

udp_host = socket.gethostname()                          # Host IP
udp_port = 12345                                     # specified port to connect

#print type(sock) ============> 'type' can be used to see type
                                        # of any variable ('sock' here)

sock.bind((udp_host,udp_port))

while True:
        print "Waiting for client..."
        data,addr = sock.recvfrom(1024)              #receive data from client

        print "Received Messages:",data," from",addr
```

Output of the above script is as follows. Keep it running and than fire up the client.py module.



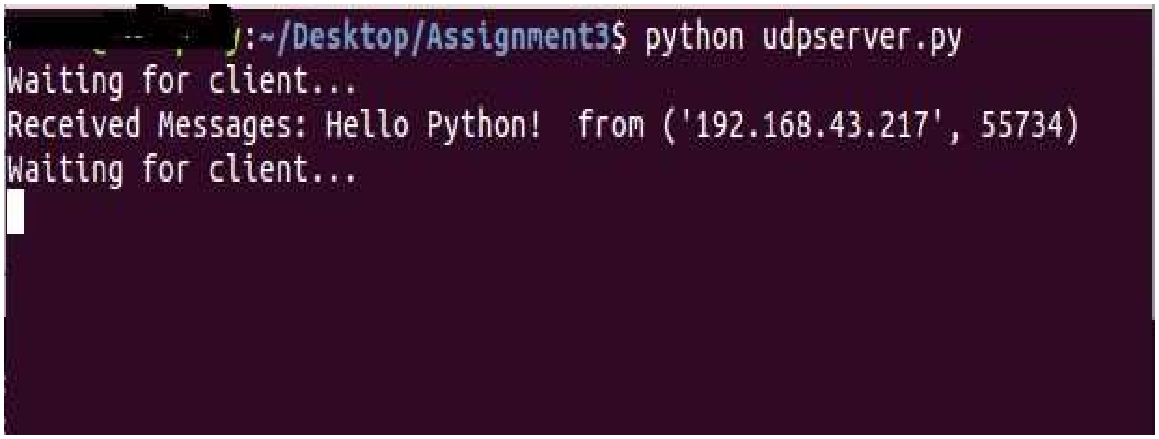
```
Waiting for client...
```

Simple UDP Client program

This is the udpclient.py script:

```
import socket

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)      # For UDP

udp_host = socket.gethostname()                         # Host IP
udp_port = 12345                                    # specified port to connect


msg = "Hello Python!"
print "UDP target IP:", udp_host
print "UDP target Port:", udp_port
```

Our udpserver.py is up and running, so now we try to runthe udpclient.py script, And here is what happened to our server after the client sends the request:



```
[redacted]:~/Desktop/Assignment3$ python udpserver.py
Waiting for client...
Received Messages: Hello Python!  from ('192.168.43.217', 55734)
Waiting for client...
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	UDP stands for________________.	User Datagram Protocol
2	__________is a connection-less and non-stream oriented protocol.	UDP
3	__________does not require a long-lived connection	UDP
4	UDP messages must fit within a	Single packet

Q-7 Explain sending E-mail using SMTP.

- Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending e-mail and routing e-mail between mail servers.
  - Python provides **smtplib** module, which defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.
  - Here is a simple syntax to create one SMTP object, which can later be used to send an e-mail −

```
import smtplib
 smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

- Here is the detail of the parameters −
- **host** − This is the host running your SMTP server. You can specify IP address of the host or a domain name like tutorialspoint.com. This is optional argument.
- **port** − If you are providing *host* argument, then you need to specify a port, where SMTP server is listening. Usually this port would be 25.
- **local_hostname** − If your SMTP server is running on your local machine, then you can specify just *localhost* as of this option.
- An SMTP object has an instance method called **sendmail**,

Example

Here is a simple way to send one e-mail using Python script. Try it once −

```
import smtplib

sender = 'from@fromdomain.com'
receivers = ['to@todomain.com']

message = """From: From Person <from@fromdomain.com>
To: To Person <to@todomain.com>
Subject: SMTP e-mail test

This is a test e-mail message.
"""
try:
  smtpObj = smtplib.SMTP('localhost')
  smtpObj.sendmail(sender, receivers, message)
  print "Successfully sent email"
except SMTPException:
  print "Error: unable to send email"
```

- Here, you have placed a basic e-mail in message, using a triple quote, takingcare to format the headers correctly.

- An e-mail requires a **From**, **To**, and **Subject** header, separated from the body of the e-mail with a blank line.

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1 2	SMTP stands for________________.	Simple Mail sendmail
	SMTP object has an instance method called ____________	
3	Python provides __________module to work with mail.	smtplib
4	Sendmail() have___________,____________ and __________ parameters.	Sender Receivers, message

Q-8 Explain File server and File client

- A simple file transfer server written in Python 3, that allows the user to download files located on the server.

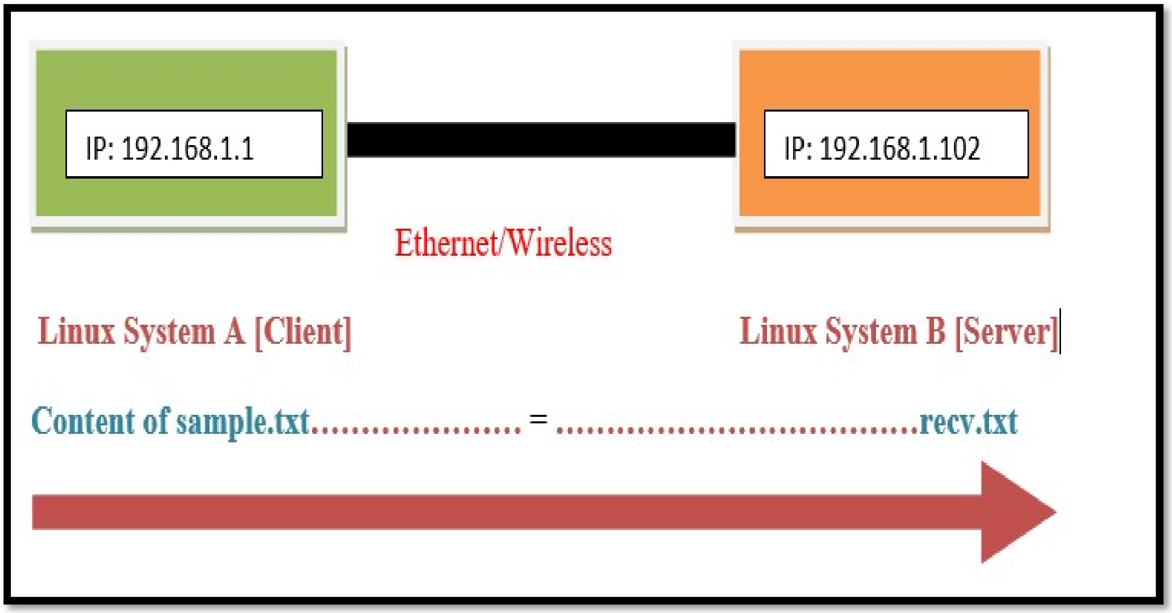
- The server can handle and serve multiple clients at the same time and send files in the same/child directories.

Usage

- Run the server.py file, entering the port you wish for the server to run on.

- Now, users can run the client.py file and connect to the server by entering the IP address and port displayed on your server.
- A user can enter the name of a file on the server (in the same folder as the server.py) and download that file - the file will appear in the same directory as the client.py file, named from_server+filename.
- file transfer is based on server client model to use socket programmingin python3+.

- **Basic Set up Diagram:**



- Here is the basic set up diagram to run this program.

File requirements:

➢ We need **server.py** and this file should be present at server system. In our case server.py should be at B_server system.

➢ Another two files **client.py** and **sample.txt** should be present at client system. In our case those two files should be present at A_client system.

- File transfer is the process of copying or moving a file from a computer to another over a network or Internet connection.

- In this tutorial, we'll go step by step on how you can write client/server Python scripts that handles that.

- The basic idea is to create a server that listens on a particular port, this server will be responsible for receiving files (you can make the server sends files as well).

- On the other hand, the client will try to connect to the server and send a file of any type.

- Here is the code to send a file from a local server to a local client.

- **<u># server.py</u>**

```
    import socket       # Import socket module

    port = 60000        # Reserve a port for your service.
    s = socket.socket() # Create a socket object
    host = socket.gethostname()# Get local machine names
    .bind((host, port))  # Bind to the port
    s.listen(5)        # Now wait for client connection.

print 'Server listening ...'
while True:

  conn, addr = s.accept()     # Establish connection with client.
```

```
     print 'Got connection from', addr
   data = conn.recv(1024)


   print('Server received', repr(data))
   filename='mytext.txt'
   f = open(filename,'rb')
   l = f.read(1024)
   while (l):
     conn.send(l)
     print('Sent ',repr(l))
     l = f.read(1024)
   f.close()

   print('Done sending')
   conn.send('Thank you for connecting')
   conn.close()
```

# client.py

```
import socket               # Import socket module

s = socket.socket()           # Create a socket object
host = socket.gethostname()      # Get local machine name
port = 60000                # Reserve a port for your service.

s.connect((host, port))
```

```
s.send("Hello server!")
with open('received_file', 'wb') as f:
  print 'file opened'
   while                    True:
      print('receiving  data...')
      data    =    s.recv(1024)
      print('data=%s',  (data))
      if not data:
         break
      # write data to a file
      f.write(data)
f.close()
print('Successfully get the file')
s.close()
print('connection closed')
```

```
Server listening....
Got connection from ('192.168.56.10', 62854)
('Server received', "'Hello server!'")
('Sent ', "'1 1234567890\\n)

('Sent ', "'4567890\\n105)
('Sent ', "'300 1234567890\\n'")
Done sending
```

Output on a local client:

```
file opened
receiving data...
data=1 1234567890
2 1234567890
...
103 1234567890
104 123
receiving data...
data=4567890
105 1234567890
106 1234567890
...
299 1234567890
receiving data...

data=300 1234567890
Thank you for connecting

receiving data...
data=Successfully get the file
connection closed
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	___________is the process of copying or moving the file across internet	File transfer
2	FTP stands for _______________ _.	File Transfer Protocol
3	_________ method can be used to close the currently opened file	close()

Q-9 Explain two way communication between client and server.

- Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on different continents.

- So we will design both server and client model so that each can communicate with them. The steps can be considered like this.

1. Python socket server program executes at first and wait for any request
2. Python socket client program will initiate the conversation at first.
3. Then server program will response accordingly to client requests.
4. Client program will terminate if user enters “bye” message.

- Server program will also terminate when client program terminates, this is optional and we can keep server program running indefinitely or terminate with some specific command in client request.

A Simple Server

A socket object is then usedto call other functions to setup a socket server.

- Now call **bind(hostname, port)** function to specify a *port* for your service on the given host.
- Next, call the *accept* method of the returned object. This method waits until a client connects to the port you specified, and then returns a *connection* object that represents the connection to that client.

A Simple Client

- Let us write a very simple client program which opens a connection to a given port 12345 and given host.
- This is very simple to create a socket client using Python's *socket* module function.
- The **socket.connect(hosname, port)** opens a TCP connection to *hostname* on the *port*. Once you have a socket open, you can read from it like any IO object.
- When done, remember to close it, as you would close a file.
- The following code is a very simple client that connects to a given host and port, reads any available data from the socket, and then exits
- Now run this server.py in background and then run above client.py to see the result.

```
# Following would start a server in background.
$ python server.py &

# Once server is started run client as follows:
$ python client.py
```

This would produce following result −

```
Got connection from ('127.0.0.1', 48437)
Thank you for connecting
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	are the endpoints of a bidirectional communications channel	sockets
2	__________ mehod can be used to specify a *port* for your service on the given host.	bind()
3	___________method waits until a client connects to the port you specified	accept

Q-10 Explain how to download webpage from internet.

- Python provides several ways to do just that in its **standard library** .

- The urllib.request method is a Python module for fetching URLs.
- It offers a very simple interface, in the form of the urlopen function.
- This is capable of fetching URLs using a variety of protocols.

urllib.request.urlopen() , open the URL url,
which can be eithera string or a Request object.

```
from urllib.request import urlopen
html = urlopen("http://net-informations.com/about.htm")

print(html.read())
```

```
downloadFile(URL_from_file)
```

If you want to obtain the contents of a web page into a variable, just read the response of urllib.request.urlopen:

```
import urllib.request
...
url = 'http://example.com/'
response = urllib.request.urlopen(url)
data = response.read()     # a `bytes` object
text = data.decode('utf-8') # a `str`; this step can't be used if data is binary
```

The easiest way to download and save a file is to usethe **urllib.request.urlretrieve** function:

```
import urllib.request

...
# Download the file from `url` and save it locally under
`file_name`:
urllib.request.urlretrieve(url, file_name)
import urllib.request
...
# Download the file from `url`, save it in a temporary directory and get
the# path to it (e.g. '/tmp/tmpb48zma.txt') in the `file_name` variable:

file_name, headers = urllib.request.urlretrieve(url)
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER

1	_____________method is a Python module for fetching URLs.	urllib.request
2	___________module provide a way to download a file is over HTTP	urllib
3	__________ method can open the URL url, which can be either a string or a Request object.	urlopen()

Q-11Explain how to download image from internet.

- The **standard Python library** for accessing websites via your program is **urllib**. It is also used by the requests module.
- Through urllib, we can do a variety of things: **access websites**, **download data**, **parse data**, **send GET and, POST requests**.
- We can download our image using just a few lines of code:

```
import urllib.request

 # setting filename and image
 URL          filename          =
 'sunshine_dog.jpg' image_url =
       "https://cdn.pixabay.com/photo/2020/02/06/09/39/summer-
       4823612_960_720.jpg"

 #calling   urlretrieve   function   to   get   resource
 urllib.request.urlretrieve(image_url, filename)
```

- We used the **urlretrieve** method to copy the required web resource toa local file.

- It is important to note that on some systems and a lot of websites, theabove code will result in an error: **HTTPError: HTTP Error 403: Forbidden**.
- This is because a lot of websites don't appreciate random programs accessing their data.
- Some programs can attack the server by sending a large number of requests. This prevents the server from functioning.

PYTHON GUI PROGRAMMING :-

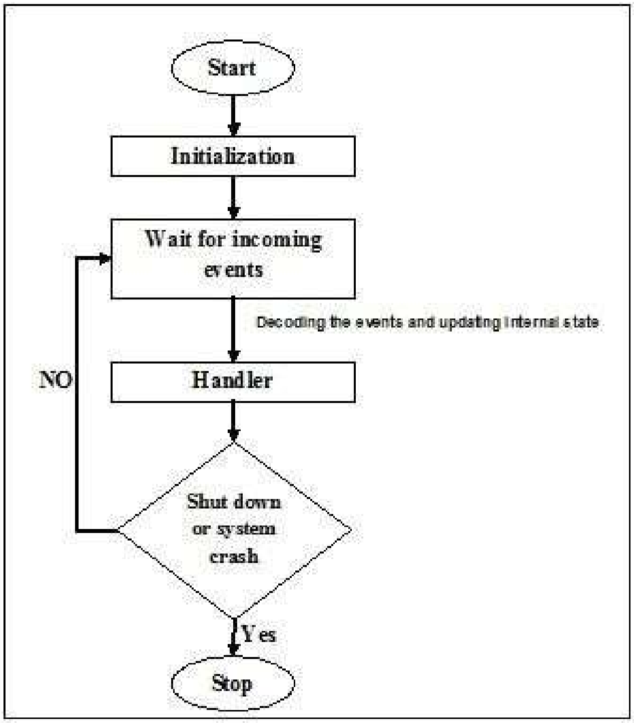
Q-1 Explain Event Driven Programming.

- Anything that happens in a user interface is an *event*.
- We say that an event is *fired* whenever the user does something – for example, clicks on a button or types a keyboard shortcut.
- Some events could also be triggered by occurrences which are not controlled by the user – for example, a background task might complete, or a network connection might be established or lost.
- Our application needs to monitor, or *listen* for, all the events that we find interesting, and respond to them in some way if they occur.
- To do this, we usually associate certain functions with particular events. We call a function which performs an action in response to an event an *event handler* – we *bind* handlers to events.
- Event-driven programming focuses on events. Eventually, the flow of program depends upon events.

- Until now, we were dealing with either sequential or parallel execution model but the model having the concept of event-driven programming is called asynchronous model.

- Event-driven programming depends upon an event loop that is always listening for the new incoming events.
- The working of event-driven programming is dependent upon events.
- Once an event loops, then events decide what to execute and in what order.

- Following flowchart will help you understand how this works −


Start
Initialization
Wait for incoming events
Decoding the events and updating internal state
Handler
NO
Shut down or system crash
Yes
Stop

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	Anything that happens in a user interface is an_________.	event
2	Event-driven programming depends upon _________________	eventloop
3	The model having concept of event-driven programming is called______________.	asynchronous model

Q-2Explain How to create simple GUI.

- **Gui stands for Graphical User Interface.**
- tkinter provides us with a variety of common GUI elements which we can use
- to build our interface – such as buttons, menus and various kinds of entry fields and display areas.
  - We call these elements *widgets*. We are going to construct a *tree* of widgets for our GUI – each widget will have a parent widget, all the way up to the *root window* of our application.
  - For example, a button or a text field needs to be *inside* some kind of containing window.
  - The widget classes provide us with a lot of default functionality.
  - They have methods for configuring the GUI's appearance – for example, arranging the elements according to some kind of *layout* – and for handling various kinds of user-driven events.
  - Once we have constructed the backbone of our GUI, we will need to customise it by integrating it with our internal application class.
  - **Our first GUI will be a window with a label and two buttons:**

```
from tkinter import Tk, Label, Button

class MyFirstGUI:
  def__init__(self, master):
    self.master = master
    master.title("A simple GUI")

    self.label = Label(master, text="This is our first GUI!")
    self.label.pack()

    self.greet_button = Button(master, text="Greet" command=self.greet)
    self.greet_button.pack()

    self.close_button = Button(master, text="Close",
   command=master.quit)
    self.close_button.pack()
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	GUI stands for___________________.	Graphical User Interface
2	__________module provides us with a variety of common GUI elements	tkinter

Q-3 Explain Python GUI – tkinter.

- Python offers multiple options for developing GUI (Graphical UserInterface).
- Out of all the GUI methods, tkinter is the most commonly used method.
- It is a standard Python interface to the Tk GUI toolkit shipped with Python.
- Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

- Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x it is ‘tkinter’.

```
import tkinter
```

- There are two main methods used which the user needs to remember while creating the Python application with GUI.

1. mainloop(): There is a method known by the name mainloop() is used when your application is ready to run.

mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

```
m.mainloop()
```

```
import  tkinter
m              =
tkinter.Tk()'''
widgets are added here
m.mainloop()
```

where m is the name of the main window object

1 Word Question – Answer

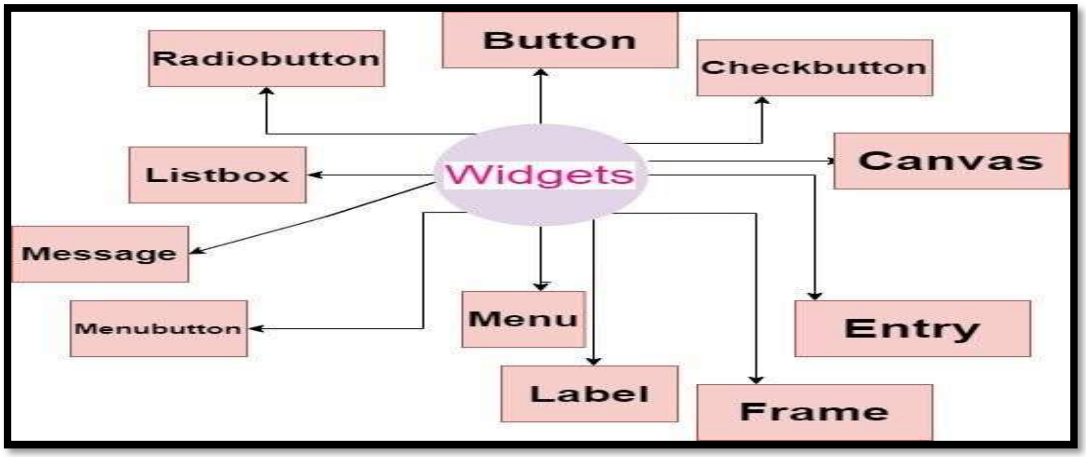
SR.NO	QUESTION	ANSWER
1	__________is the fastest and easiest way to create the GUI applications.	tkinter
2	_____________is an infinite loop used to run the application	mainloop()

Q-4 Explain Widgets in Python GUI.

- Widgets are the bread and butter of the Python GUI framework Tkinter.
- They are the elements through which users interact with your program.
- Each **widget** in Tkinter is defined by a class. Here are some of the widgetsavailable:

- There are various controls, such as **buttons**, **labels**, **scrollbars**, **radiobuttons**, and **text boxes** used in a GUI application.
- These **little components** or controls of **Graphical User Interface (GUI)** are known as **widgets** in Tkinter.

- These are **19 widgets** available in Python Tkinter module.



Widget Class	Description
Label	A widget used to display text on the screen
Button	A button that can contain text and can perform an action when Clicked
Entry	A text entry widget that allows only a single line of text
Text	A text entry widget that allows multiline text entry
Frame	A rectangular region used to group related widgets or provide Padding between widgets

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	Each_________in tkinter is defined by a class	widget
2	**little components** or controls of **Graphical User Interface (GUI)** are known as________in Tkinter.	widget

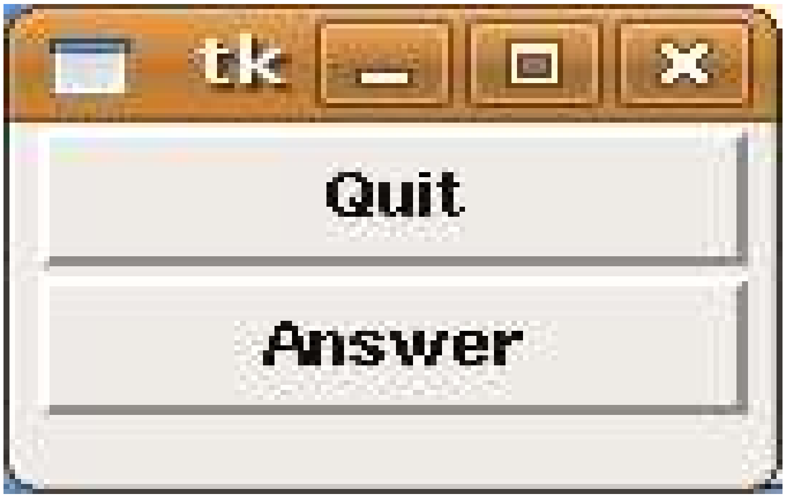
Q-5 Explain Dialogues and Message boxes.

- Tkinter (and TK of course) provides a set of dialogues (dialogs in American English spelling), which can be used to display message boxes, showing warning or errors, or widgets to select files and colours.

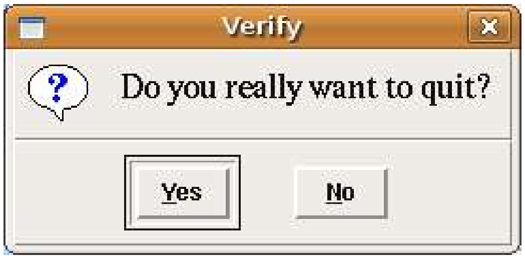
- There are also simple dialogues, asking the user to enter string, integers or float numbers.

- Let's look at a typical GUI Session with Dialogues and Message boxes.

- There might be a button starting the dialogue, like the "quit" button in the following window:



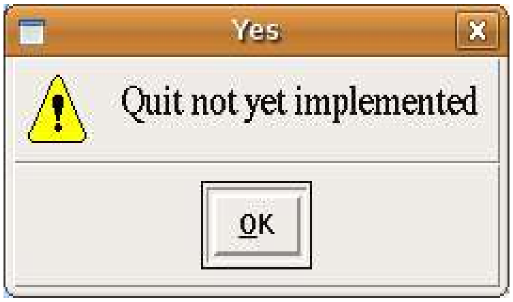
Pushing the "quit" button raises the Verify window:



Verify

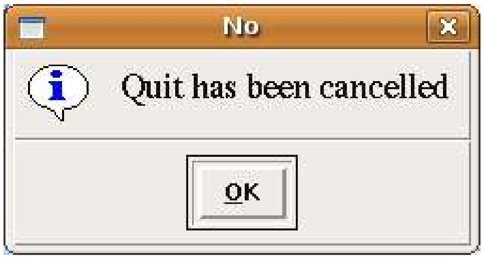
Do you really want to quit?

Let's assume that we want to warn users that the "quit" functionality is not yet implemented. In this case we can use the warning message to inform the user, if he or she pushes the "yes" button:



Yes

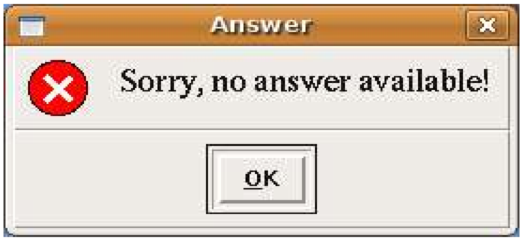
Quit not yet implemented



No

Quit has been cancelled

Let's go back to our first Dialogue with the "quit" and "answer" buttons. If the "Answer" functionality is not implemented, it might be useful to use the following error message box:



Answer

Sorry, no answer available!

- Python script, which implements the previous dialogue widgets:

```
import tkinter as tk
from tkinter import messagebox as mb

def answer():
  mb.showerror("Answer", "Sorry, no answer
available")def callback():
  if mb.askyesno('Verify', 'Really quit?'):
    mb.showwarning('Yes', 'Not yet
    implemented')
  else:
    mb.showinfo('No', 'Quit has been cancelled')


    tk.Button(text='Quit', command=callback).pack(fill=tk.X)
tk.Button(text='Answer', command=answer).pack(fill=tk.X)
tk.mainloop()
```

Message Boxes

- The message dialogues are provided by the 'messagebox' submodule oftkinter.
- 'messagebox' consists of the following functions, which correspond todialog windows:
- askokcancel(title=None, message=None, **options)
  Ask if operation should proceed; return true if the answer is ok
- askquestion(title=None, message=None, **options)Ask a question
- askretrycancel(title=None, message=None, **options)
  Ask if operation should be retried; return true if the answer is yes
- askyesno(title=None, message=None, **options) Ask a question; return true if the answer is yes
- askyesnocancel(title=None, message=None, **options)

Ask a question; return true if the answer is yes, None if cancelled.

- showerror(title=None, message=None, **options)Show an error message
- showinfo(title=None, message=None, **options)Show an info message
- showwarning(title=None, message=None, **options)Show a warning message

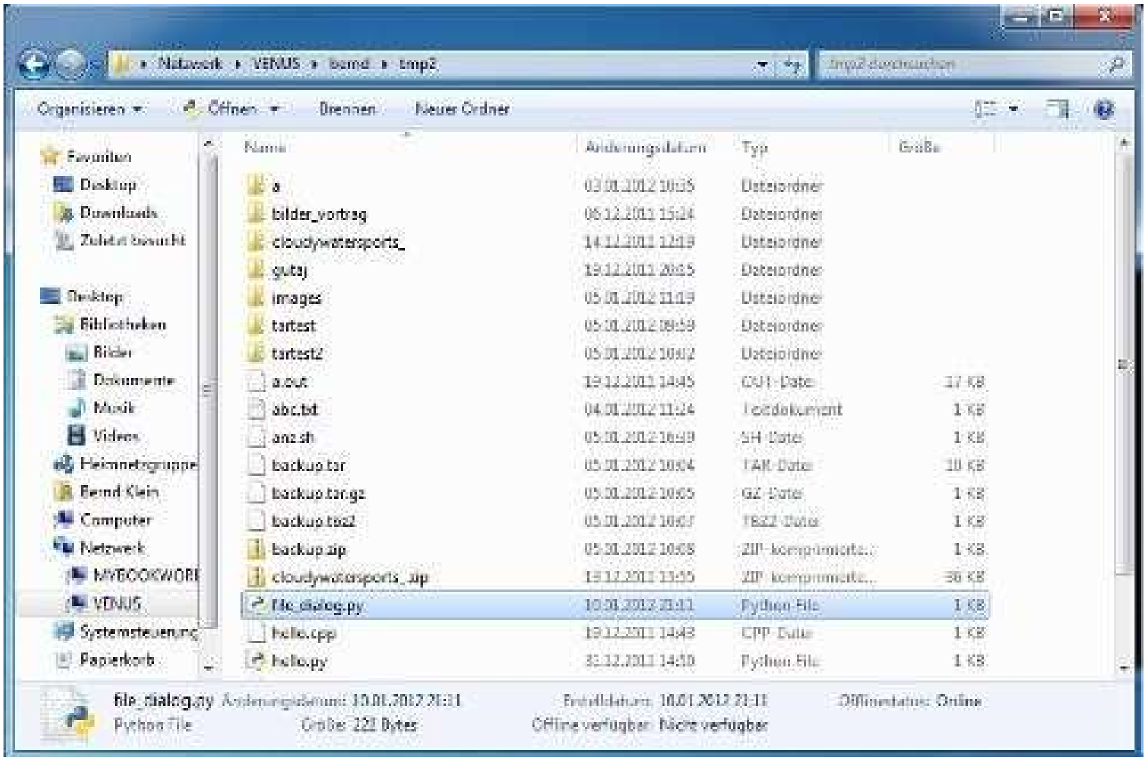
Open File Dialogue

- There is hardly any serious application, which doesn't need a way to read from a file or write to a file. Furthermore, such an application might have tochoose a directory.

- Tkinter provides the module tkFileDialog for these purposes.

```
import tkinter as tk
from tkinter import filedialog as fd
def callback():
   name= fd.askopenfilename()
   print(name)
errmsg = 'Error!'
tk.Button(text='File Open',
     command=callback).pack(fill=tk.X)
tk.mainloop()
```

- The code above creates a window with a single button with the text "FileOpen".
- If the button is pushed, the following window appears:



Name	Änderungsdatum	Typ	Größe
a	03.01.2012 10:35	Dateiordner	
bilder_vortrag	06.12.2011 15:24	Dateiordner	
cloudywatersports_	14.12.2011 12:19	Dateiordner	
gutaj	19.12.2011 20:35	Dateiordner	
images	05.01.2012 11:19	Dateiordner	
tartest	05.01.2012 09:59	Dateiordner	
tartest2	05.01.2012 10:02	Dateiordner	
a.out	19.12.2011 14:45	OUT-Datei	17 KB
abc.txt	04.01.2012 11:24	Textdokument	1 KB
anz.sh	05.01.2012 16:39	SH-Datei	1 KB
backup.tar	05.01.2012 10:04	TAR-Datei	10 KB
backup.tar.gz	05.01.2012 10:05	GZ-Datei	1 KB
backup.tbz2	05.01.2012 10:07	TBZ2-Datei	1 KB
backup.zip	05.01.2012 10:08	ZIP-komprimierte...	1 KB
cloudywatersports_.zip	19.12.2011 13:55	ZIP-komprimierte...	36 KB
file_dialog.py	10.01.2012 21:11	Python File	1 KB
hello.cpp	19.12.2011 14:43	CPP-Datei	1 KB
hello.py	31.12.2011 14:50	Python File	1 KB

file_dialog.py Änderungsdatum: 10.01.2012 21:11 Erstelldatum: 10.01.2012 21:11 Offlinestatus: Online
Python File Größe: 222 Bytes Offline verfügbar: Nicht verfügbar

- The above example was created using a gnome desktop under Linux. If westart the same program under Windows 7, it looks like this:

Choosing a Colour

- There are applications where the user should have the possibility to select acolour.
- Tkinter provides a pop-up menu to choose a colour. To this purpose wehave to import the 'tkinter.colorchooser' module and have to use the method askColor:

```
result = tkinter.colorchooser.askcolor ( color, option=value, ...)
```

- If the user clicks the OK button on the pop-up window, respectively, the return value of askcolor() is a tuple with two elements, both a representation of the chosen colour, e.g. ((106, 150, 98), '#6a9662') The first element return[0] is a tuple (R, G, B) with the RGB representationin decimal values (from 0 to 255).
- The second element return[1] is a hexadecimal representation of thechosen colour.

If the user clicks "Cancel" the method returns the tuple (None, None).

The optional keyword parameters are:

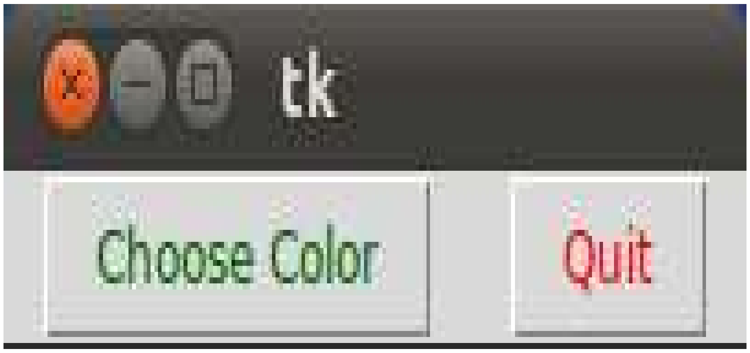
color	The variable color is used to set the default colour to be displayed. If color is not set, the initial colour will be grey.
title	The text assigned to the variable title will appear in the pop-up window's title area. The default title is "Color".
parent	Make the pop-up window appear over window W. The default behaviour is that it appears over the root window.

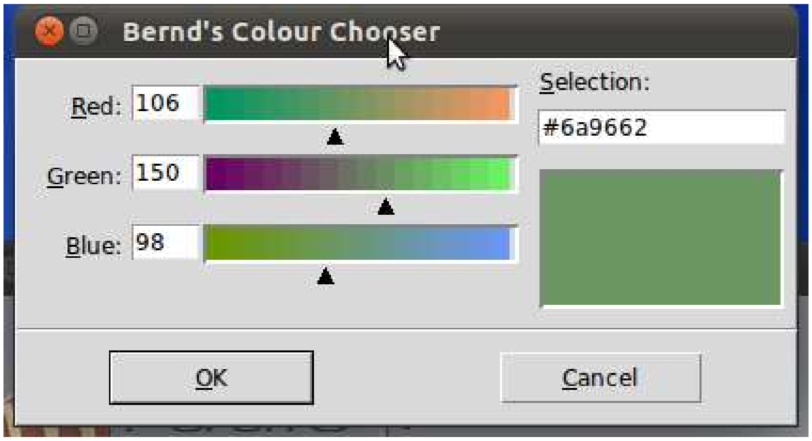
Let's have a look at an example:

```
import tkinter as tk

from tkinter.colorchooser import askcolor
```

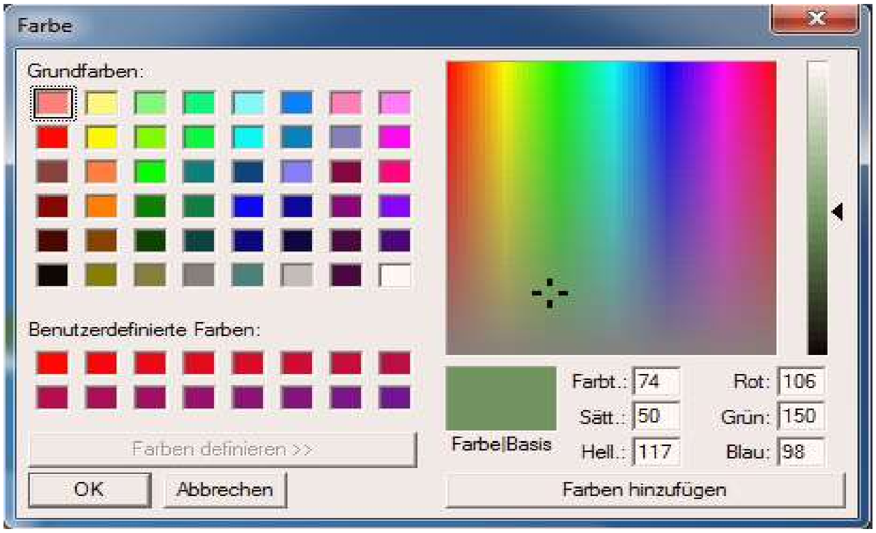
```
def callback():
    result = askcolor(color="#6A9662",
                      title = "Bernd's Colour Chooser")
    print(result)
root = tk.Tk()
tk.Button(root,
          text='Choose Color',
          fg="darkgreen",
          command=callback).pack(side=tk.LEFT, padx=10)
tk.Button(text='Quit',
          command=root.quit,
          fg="red").pack(side=tk.LEFT, padx=10)
tk.mainloop()
```

The look and feel depends on the operating system (e.g. Linux orWindows) and the chosen GUI (GNOME, KDE and so on). The followingwindows appear, if you use Gnome:




Bernd's Colour Chooser

Red: 106
Green: 150
Blue: 98
Selection: #6a9662


Farbe

Farbt.: 74
Sätt.: 50
Hell.: 117
Rot: 106
Grün: 150
Blau: 98

- Using the same script under Windows 7 gives us the following result.

Q-6 Explain Widget Attributes

- In this part ,we talk about standard widget attributes, including cursors,reliefs, colours, and fonts.
- The standard widget attributes are keywords used in widget constructors.

o Tkinter widget state

- The state attribute defines the state of the widget. It can have the followingvalues: NORMAL, ACTIVE, and DISABLED.

We demonstrate the state attribute on the Label widget.

```
label1 = Label(self, text=txt, state=NORMAL)
```

o Tkinter Background colours

- The background colours of widgets can be set with background attribute. Itcan be abbreviated to bg.
- Likewise, the foreground colours of widgets can be set with foreground attribute. It can be abbreviated to fg.

In the code example, we create label with background colour.

```
lbl1 = Label(frame, bg='SlateGray3', width=15, height=10)
lbl1.pack(side=LEFT, padx=3)
```

o Width and height

- The width and height attributes set the width and height of the widget. In the example, we control the size of buttons with width and height.

```
btn2 = Button(frame, text='Button', width=5 ,height =4 )
```

o Tkinter fonts

- Tkinter has a tkinter.font module for working with fonts. It has some built- in fonts such as TkTooltipFont, TkDefaultFont, or TkTextFont. The font is setwith the font attribute.

  Here we use a built-in font name.

```
label3 = Label(self, text=txt, font=('Times', '18', 'italic'))
```

o Tkinter cursors

- The cursor is a small icon that shows where the mouse pointer is located.The cursor in Tkinter is set with the cursor attribute.

  In the example, we have four labels. Different cursors are used for eachof the labels.

```
lbl4 = Label(frame, bg='DarkSeaGreen4', width=15, height=10,
  cursor='pencil')
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	A_________is a small icon that shows where the mouse pointer is located.	cursor
2	___________attribute defines the state of the widget	state
3	Tkinter has a___________module for working with fonts.	tkinter.font

Q-7Explain Layout Manager in tkinter.

- When we design the GUI of our application, we decide what widgets we willuse and how we will organize those widgets in the application.

- To organize our widgets, we use specialized non-visible objects calledlayout managers.

- There are two kinds of widgets: **containers and their children**.
- The containers group their children into suitable layouts.
- Tkinter has three built-in layout managers: the pack, grid,and place managers.

- The place geometry manager positions widgets using absolute positioning. The pack geometry manager organizes widgets in horizontal and vertical boxes. The grid geometry manager places widgets in a two dimensional grid.

Absolute positioning

- In most cases, programmers should use layout managers.
- There are a few situations where we could use absolute positioning.
- In absolute positioning, the programmer specifies the position and the sizeof each widget in pixels.
- The size and the position of a widget do not change if we resize a window. Applications look different on various platforms, and what looks OK on Linux, might not look OK on Mac OS.

- Changing fonts in our application might spoil the layout. If we translate ourapplication into another language, we must redo our layout.

```
absolute.py
#!/usr/bin/env
python3

"""
ZetCode Tkinter tutorial

In this script, we lay out
images using absolute
positioning.

from PIL import Image,
ImageTk    from    tkinter
import Tk, BOTH
from tkinter.ttk import Frame, Label, Style

class Example(Frame):

   def__init
      ___(self):
      super().__init__()

      self.initUI()

      def initUI(self):

      self.master.title("Absolute
      positioning")      self.pack(fill=BOTH,
      expand=1)

      Style().configure("TFrame", background="#333")

      bard = Image.open("bardejov.jpg")

bardejov=
ImageTk.PhotoImage(bard)
label1=Label(self, image=bardejov)
```

In this example, we place three images using absolute positioning. Weuse the place geometry manager.

```
from PIL import Image, ImageTk
```

- We use Image and ImageTk from the Python Imaging Library (PIL) module.

```
style = Style()
style.configure("TFrame", background="#333")
```

We configure our frame to have a dark gray background using styles.

```
bard = Image.open("bardejov.jpg")
bardejov = ImageTk.PhotoImage(bard)
```

We create an image object and a photo image object from an image in the current working directory.

```
label1 = Label(self, image=bardejov)
```

We create a Label with an image. Labels can contain text or images.

```
label1.image = bardejov
```

We must keep the reference to the image to prevent image from being garbage collected.

```
label1.place(x=20, y=20)
```

The label is placed on the frame at x=20 and y=20 coordinates.

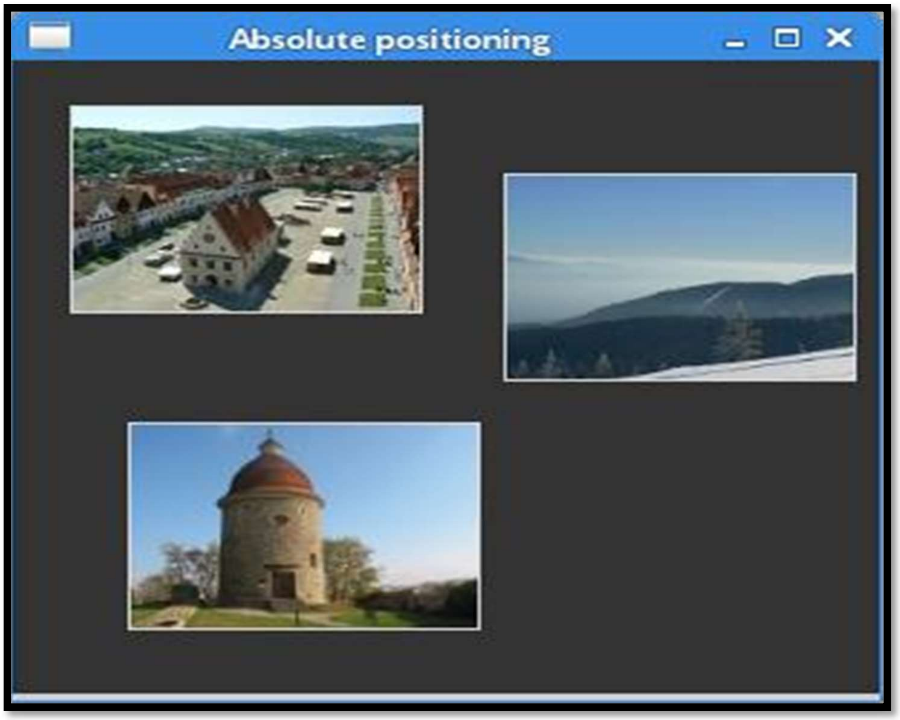


Figure: Absolute positioning

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	The containers group their children into___________.	suitable layout
2	the position and the size of each widget can be specified in _____________	pixels
3	Tkinter has three built-in layout managers: ____________,_______ &___________.	pack Grid place managers.

Q-8 Explain Button , Label ,Entryfield.

Button:To add a button in your application, this widget is used.The general syntax is:

w=Button(master, option=value)

- master is the parameter used to represent the parent window.
- There are number of options which are used to change the format of theButtons.
- Number of options can be passed as parameters separated by commas.Some of them are listed below.
  - **activebackground**: to set the background color when button isunder the cursor.
  - **activeforeground**: to set the foreground color when button is underthe cursor.
  - **bg**: to set he normal background color.
  - **command**: to call a function.
  - **font**: to set the font on the button label.
  - **image**: to set the image on the button.
  - **width**: to set the width of the button.
  - **height**: to set the height of the button.

CheckButton: To select any number of options by displaying a number of options toa user as toggle buttons.

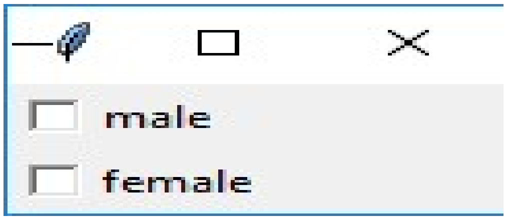
The general syntax is:

w = CheckButton(master, option=value)

- There are number of options which are used to change the format of thiswidget.
- Number of options can be passed as parameters separated by commas.Some of them are listed below.
  - **Title**: To set the title of the widget.
    activebackground: to set the background color when widget isunder the cursor.
    activeforeground: to set the foreground color when widget isunder the cursor.
  - **bg**: to set he normal backgrouSteganographyBreak.
  - **command**: to call a function.
  - **font**: to set the font on the button label.
  - **image**: to set the image on the widget.

```
from tkinter import *

 master = Tk()

 var1 = IntVar()
Checkbutton(master,  text='male',  variable=var1).grid(row=0,
 sticky=W)
var2 = IntVar() Checkbutton(master, text='female',
 variable=var2).grid(row=1, sticky=W)
  mainloop()
```

Output :-



□ male
□ female

Entry: It is used to input the single line text entry from the user..
For multi-line text input, Text widget is used.

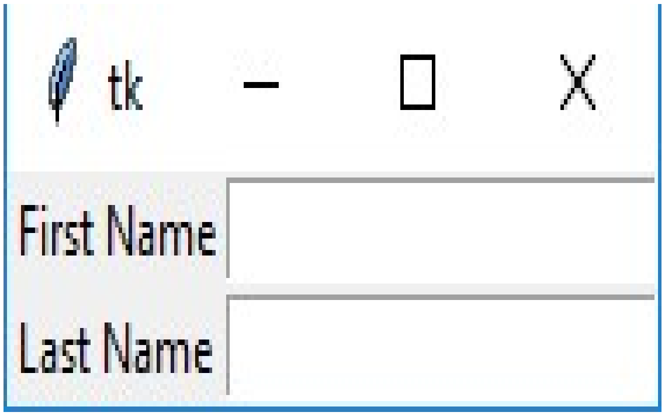
The general syntax is:

w=Entry(master, option=value)

- master is the parameter used to represent the parent window.
- There are number of options which are used to change theformat of the widget.
- Number of options can be passed as parameters separatedby commas. Some of them are listed below.
  - **bd**: to set the border width in pixels.
  - **bg**: to set the normal background color.
  - **cursor**: to set the cursor used.
  - **command**: to call a function.
  - **highlightcolor**: to set the color shown in the focus highlight.
  - **width**: to set the width of the button.
  - **height**: to set the height of the button.

```
from        tkinter
import * master =
Tk()
Label(master, text='First Name').grid(row=0)


Label(master,text='Last Name').grid(row=1)


e1 = Entry(master)

e2      =      Entry(master)

e1.grid(row=0, column=1)

e2.grid(row=1, column=1)

mainloop()
```

Output :-



First Name

Last Name

Frame: It acts as a container to hold the widgets. It is used for grouping and organizing the widgets.

The general syntax is:

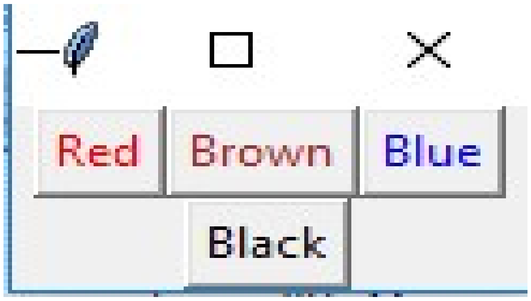
w = Frame(master, option=value)

- master is the parameter used to represent the parent window.
- There are number of options which are used to change the format of thewidget.
- Number of options can be passed as parameters separated by commas.Some of them are listed below.
  - **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
  - **bd**: to set the border width in pixels.
  - **bg**: to set the normal background color.
  - **cursor**: to set the cursor used.
  - **width**: to set the width of the widget.
  - **height**: to set the height of the widget.

```
from        tkinter

import  *  root  =

Tk()
```

```
frame = Frame(root)

frame.pack()

bottomframe = Frame(root)

bottomframe.pack( side = BOTTOM )

redbutton = Button(frame, text = 'Red', fg ='red')

redbutton.pack( side = LEFT)

greenbutton = Button(frame, text = 'Brown', fg='brown')

greenbutton.pack( side = LEFT )

bluebutton = Button(frame, text ='Blue', fg ='blue')

bluebutton.pack( side = LEFT )

blackbutton = Button(bottomframe, text ='Black', fg ='black')
blackbutton.pack( side = BOTTOM)

root.mainloop()
```

Output:-



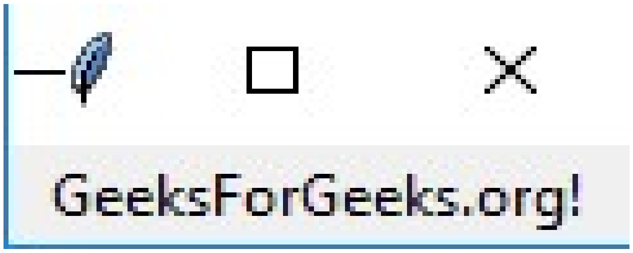
Label: It refers to the display box where you can put any text or image which canbe updated any time as per the code.
The general syntax is:

w=Label(master, option=value)

- master is the parameter used to represent the parent window.
- There are number of options which are used to change the format of thewidget.
- Number of options can be passed as parameters separated by commas. Some of them are listed below.
  - **bg**: to set he normal background color.
  - **bg** to set he normal background color.
  - **command**: to call a function.
  - **font**: to set the font on the button label.
  - **image**: to set the image on the button.
  - **width**: to set the width of the button.
  - **height**” to set the height of the button.

```
from        tkinter

import  *  root  =

Tk()

w  =  Label(root,  text='GeeksForGeeks.org!')

w.pack()

root.mainloop()
```

Output :-



GeeksForGeeks.org!

Listbox: It offers a list to the user from which the user can accept any number ofoptions.
The general syntax is:

w = Listbox(master, option=value)

- master is the parameter used to represent the parent window.
- There are number of options which are used to change the format of thewidget.
- Number of options can be passed as parameters separated by commas.Some of them are listed below.

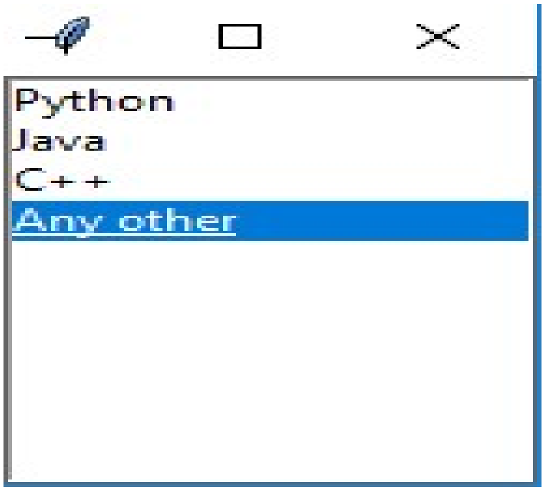
- **highlightcolor**: To set the color of the focus highlight whenwidget has to be focused.

- **bg**: to set he normal background color.
- **bd**: to set the border width in pixels.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from        tkinter
import  *  top   =
Tk()
Lb               =
Listbox(top)
Lb.insert(1,
'Python')
Lb.insert(2,
'Java')
```

```
Lb.insert(3,   'C++')
Lb.insert(4,      'Any
other')     Lb.pack()
top.mainloop()
```

Output :-



```
Python
Java
C++
Any other
```

SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.

(AFFILIATED TO SAURASHTRA UNIVERSITY)

2 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001

3 – Vaishalinagar
Nr. Amrapali Under Bridge
Raiya Road
Rajkot – 360001

CHAPTER – 5

CONNECTING WITH DATABASE

- **Verifying the MySQL dB Interface Installation,**
- **Working with MySQL Database,**
- **Using MySQL from Python,**
- **Retrieving All Rows from a Table,**
- **Inserting Rows into a Table,**
- **Deleting Rows from a Table,**
- **Updating Rows in a Table,**
- **Creating Database Tables through Python**

Q-1 What is Database and what is MySQL db?

- **A database** is basically a collection of structured data in such a way that it caneasily be retrieved, managed and accessed in various ways.
- One of the simplest forms of databases is a text database. Relational databases- the most popular database system which includes the following:
  - MySQL
  - Oracle Database
  - SQL server
  - Sybase
  - Informix
  - IBM db2
  - NO SQL
- Among all these databases, **MySQL** is one of the easiest databases to work with.
- Let me walk you through about this in detail.

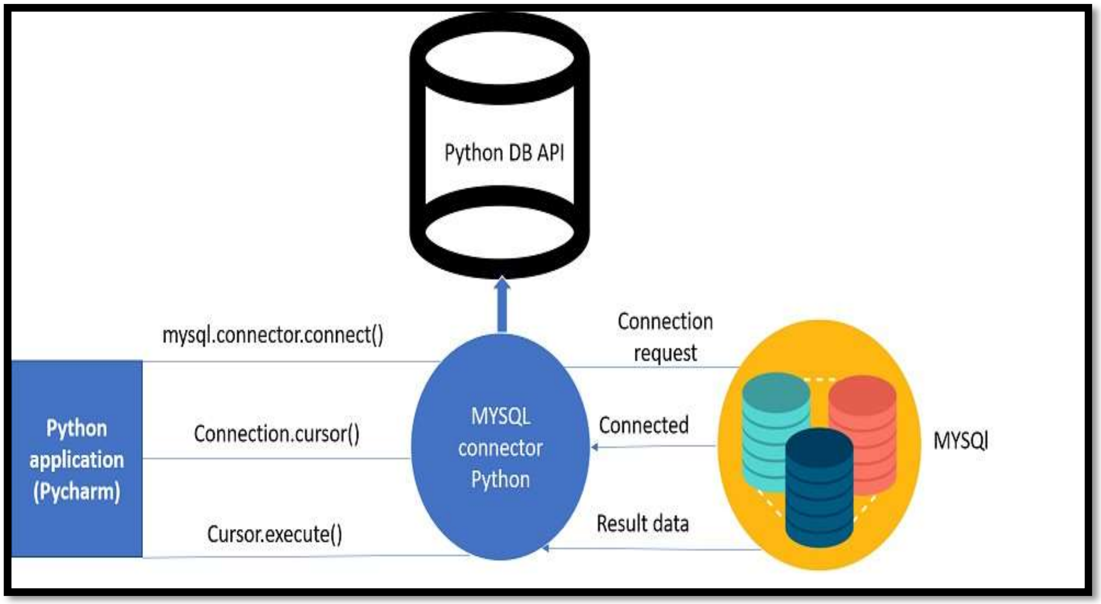
What is MySQLdb?

- MySQLdb is an open-source freely available relational database managementsystem that uses Structured Query Language.
- Now one of the most important question here is “What is SQL?”
- SQL (Structured Query Language) is a standard language for relational databases that allow users to do various operations on data like, Manipulating, Creating, Dropping, etc.
- In a nutshell, SQL allows you to do anything with the data.

Q-2 Explain how does python connect to the MySQL database?

Detail :-

- It is very simple to connect Python with the database.
- Refer the below image which illustrates a Python connection with the database where how a connection request is sent to MySQL connector Python, gets accepted from the database and cursor is executed with result data.



- Before connecting to the MySQL database, make sure you have MySQLinstaller installed on your computer.
- It provides a comprehensive set of tools which helps in installing MySQL withthe following components

  - MySQL server
  - All available connectors
  - MySQL Workbench
  - MySQL Notifier
  - Tools for Excel and Microsoft Visual Studio
  - MySQL Sample Databases
  - MySQL Documentation

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	Before connecting to the MySQL database, make sure you have__________installed on your computer.	MySQL installer

Q-3 How to verify installation of MySQL dB interface.

Detail :-

Install mysql.connector

- To connect the python application with the MySQL database, we must importthe mysql.connector module in the program.
- The mysql.connector is not a built-in module that comes with the pythoninstallation. We need to install it to get it working.
- Execute the following command to install it using pip installer.
- > python -m pip install mysql-connector

Or follow the following steps.

1. Click the link to download the source code :

https://files.pythonhosted.org/packages/8f/6d/fb8ebcbbaee68b172ce3dfd08c7b8660d09f91d8d5411298bcacbd309f96/mysql-connector-python-8.0.13.tar.gz

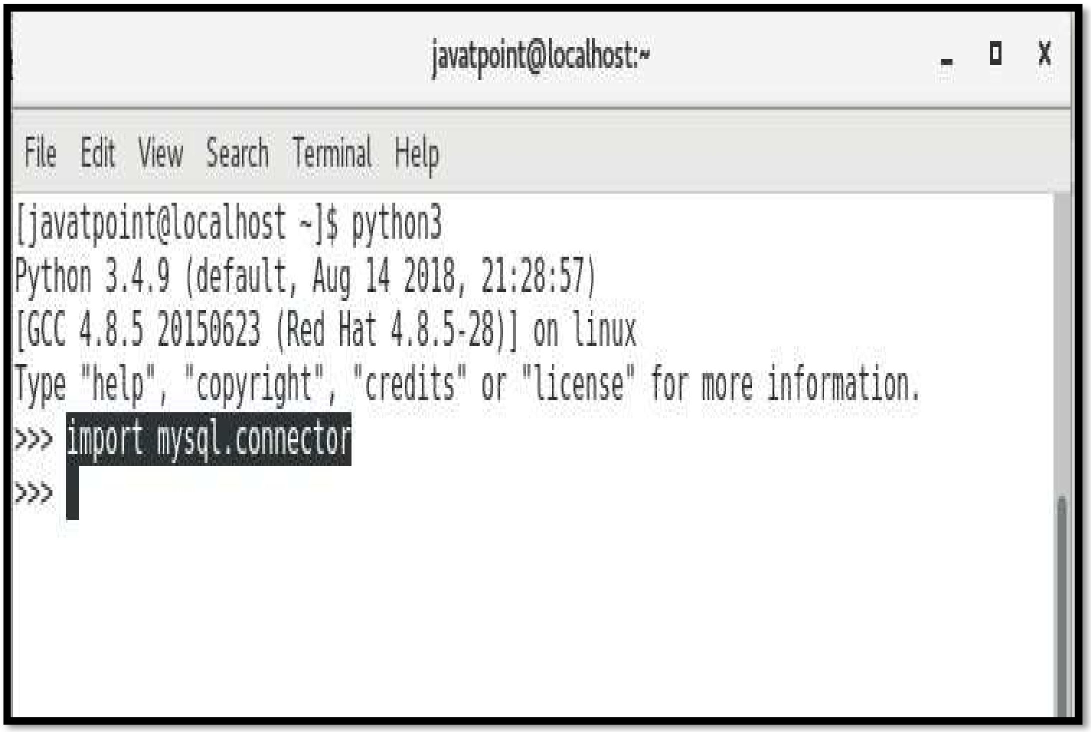
- Open the terminal (CMD for windows) and change the present workingdirectory to the source code directory.
- **$ cd mysql-connector-python-8.0.13/**

1. Run the file named setup.py with python (python3 in case you have also installedpython 2) with the parameter build.

- **$ python setup.py build**

Run the following command to install the mysql-connector

- **$ python setup.py install**
- This will take a bit of time to install mysql-connector for python. We can verify the installation once the process gets over by importing mysql-connector on the python shell.



```
[javatpoint@localhost ~]$ python3
Python 3.4.9 (default, Aug 14 2018, 21:28:57)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import mysql.connector
>>> 
```

- Hence, we have successfully installed mysql-connector for python on oursystem.

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	To connect the python application with the MySQL database, we must import the_______________module.	mysql.connector
2	_______________Command can be used to install mysql-connector	python setup.py install

Q-4 Write note on working with MySQL Database.

Detail :-

Database Connection

- In this section ,we will discuss the steps to connect the python application tothe database.
- There are the following steps to connect a python application to our database.

Import mysql.connector module

Create the connection object.

Create the cursor objectExecute the query

Creating the connection

- To create a connection between the MySQL database and the python application, the connect() method of mysql.connector module is used.
- Pass the database details like HostName, username, and the database password in the method call. The method returns the connection object.

- The syntax to use the connect() is given below.

Connection-Object= mysql.connector.connect(host = <host- name> , user = <username> , passwd = <password>)

- Consider the following example.

Example

1. **import mysql.connector**
2. **#Create the connection object**
3. **myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google")**

```
#printing the connection object
        print(myconn)
```

Output:

```
<mysql.connector.connection.MySQLConnection object at 0x7fb142edd780>
```

- Here, we must notice that we can specify the database name in the connect()method if we want to connect to a specific database.

Creating a cursor object

- The cursor object can be defined as an abstraction specified in the Python DB-API 2.0.
- It facilitates us to have multiple separate working environments through thesame connection to the database.
- We can create the cursor object by calling the 'cursor' function of theconnection object.
- The cursor object is an important aspect of executing queries to thedatabases.

The syntax to create the cursor object is given below

```
<my_cur> = conn.cursor()
```

Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google", database = "mydb")
#printing the connection object
print(myconn)6
creating the cursor object
cur = myconn.cursor()
print(cur)
```

Output :-

```
<mysql.connector.connection.MySQLConnection object at 0x7faa17a15748>
MySQLCursor: (Nothing executed yet)
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	To create a connection between the MySQL database and the python application, the______method can be used.	connect()
2	The_________object is an important aspect of executing queries to the databases.	cursor

Q-5Write note on creating new MySQL Database.

Detail :-

- In this section , we will create the new database PythonDB.

Getting the list of existing databases

- We can get the list of all the databases by using the following MySQL query.

```
> show database
```

Example :-

```
1.import
mysql.connector2.
3. #Create the connection object


4. myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
   = "google")

5. #creating the cursor object
6. cur = myconn.cursor()
```

```
try:
    dbs = cur.execute("show databases")


except:
    myconn.rollback()
for x in cur:
    print(x)
myconn.close()
```

```
('EmployeeDB',)
('Test',)
('TestDB',)
('information_schema',)
('javatpoint',)
('javatpoint1',)
('mydb',)
('mysql',)
('performance_schema',)
('testDB',)
```

Creating the new database

- The new database can be created by using the following SQL query.

```
>   create   database   <database-name>Example
```

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google")
#creating the cursor object
cur = myconn.cursor()

try:
    #creating a new database
```

```
8.    cur.execute("create database PythonDB2")
9.    #getting the list of all the databases which will now include the new
    database PythonDB
10.   dbs = cur.execute("show databases")

11.   except:
12.
      myconn.rollback
()13.for x in cur:
14.       print(x)
15.myconn.close()
```

Output :-

```
('EmployeeDB',)
('PythonDB',)
('Test',)
('TestDB',)
('anshika',)
('information_schem
a',) ('javatpoint',)
('javatpoint1',)
('mydb',)


('mydb1',)
('mysql',)
('performance_schema',)
('testDB',)
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	The syntax for creating new database is ___________________	create database <databasename>

2	___________Method can be used to execute particular query.	execute()
3	____________ method can be used to close the connection.	close()

Q-6 Write note on creating Database Table through Python.

Detail :-

Creating the table

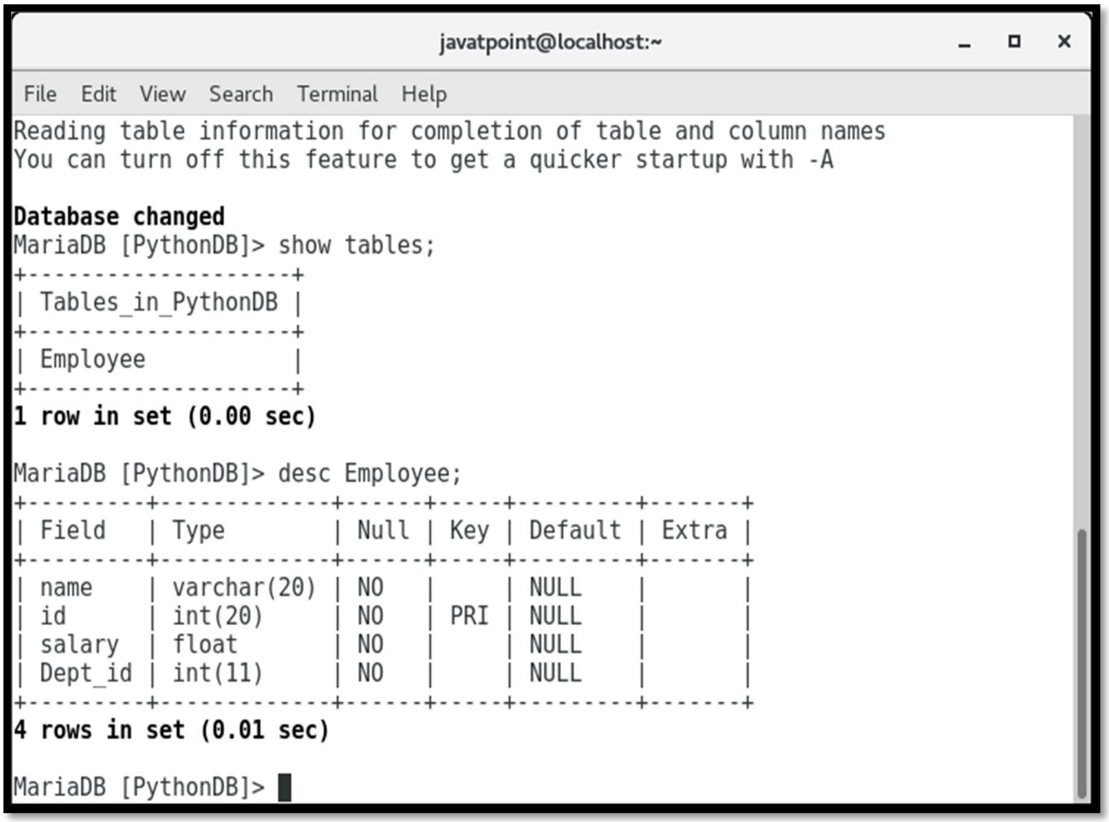
- In this section , we will create the new table Employee. We have to mentionthe database name while establishing the connection object.
- We can create the new table by using the CREATE TABLE statement of SQL. In our database PythonDB, the table Employee will have the four columns, i.e., name, id, salary, and department_id initially.
- The following query is used to create the new table Employee.

```
> create table Employee (name varchar(20) not null, id int primarykey, salary float not null, Dept_Id int not null)
```

Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object

cur = myconn.cursor()


try:
    #Creating a table with name Employee having four columns i.e., name, id, salary, and department id
    dbs = cur.execute("create table Employee(name varchar(20) not null, id int(20) not null primary key, salary float not null, Dept_id int not null)")
except:
```

10. **myconn.rollback()**
11. **Myconn.close()**



```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> show tables;
+--------------------+
| Tables_in_PythonDB |
+--------------------+
| Employee           |
+--------------------+
1 row in set (0.00 sec)

MariaDB [PythonDB]> desc Employee;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | NO   |     | NULL    |       |
| id      | int(20)     | NO   | PRI | NULL    |       |
| salary  | float       | NO   |     | NULL    |       |
| Dept_id | int(11)     | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

MariaDB [PythonDB]> 
```

- Now, we may check that the table Employee is present in the database.

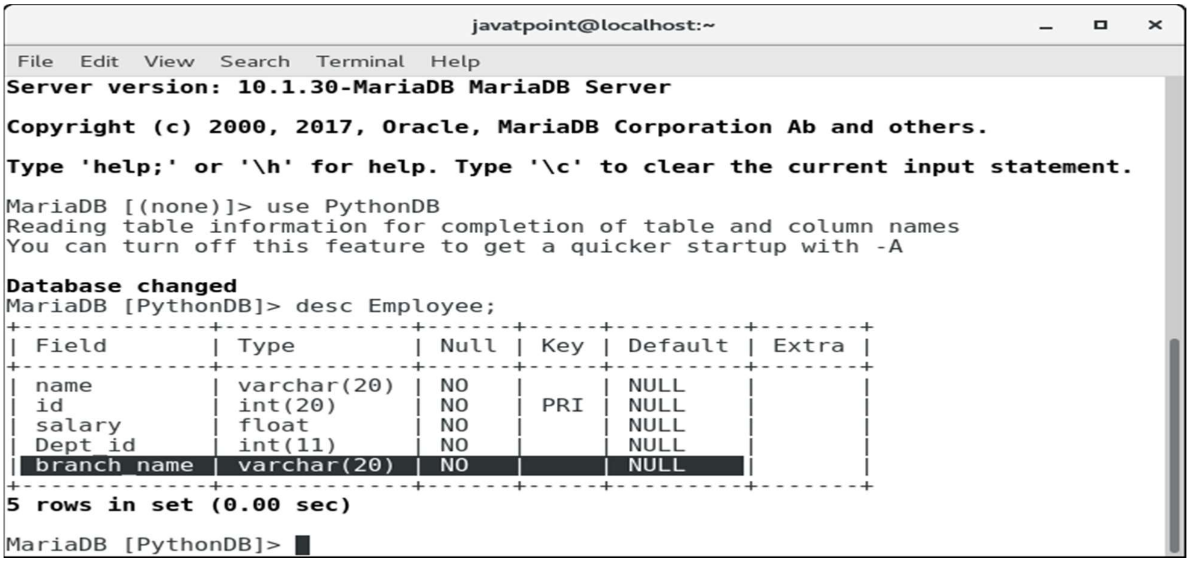
Alter Table

- Sometimes, we may forget to create some columns, or we may need to updatethe table schema.
- The alter statement used to alter the table schema if required.
- Here, we will add the column branch_name to the table Employee.
- The following SQL query is used for this purpose.

alter table Employee add branch_name varchar(20) not null

Consider the following example.

```
import mysql.connector

#Create the connection object

myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")

#creating the cursor object
cur = myconn.cursor()
try:
    #adding a column branch name to the table Employee
    cur.execute("alter table Employee add branch_name varchar(20) not null")
 except:
 myconn.rollback()
 Myconn.close()
```



```
Server version: 10.1.30-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use PythonDB
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> desc Employee;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| name        | varchar(20) | NO   |     | NULL    |       |
| id          | int(20)     | NO   | PRI | NULL    |       |
| salary      | float       | NO   |     | NULL    |       |
| Dept_id     | int(11)     | NO   |     | NULL    |       |
| branch_name | varchar(20) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

MariaDB [PythonDB]> 
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	Table can be created using ________________statement	Create Table

2	_____________module must be import to connect with mysql.	mysql.connector
3	___________method can be used to close the connection.	mysql.close()

Q-7 Write note on inserting rows into table.

Detail :-

Insert Operation - Adding a record to the table

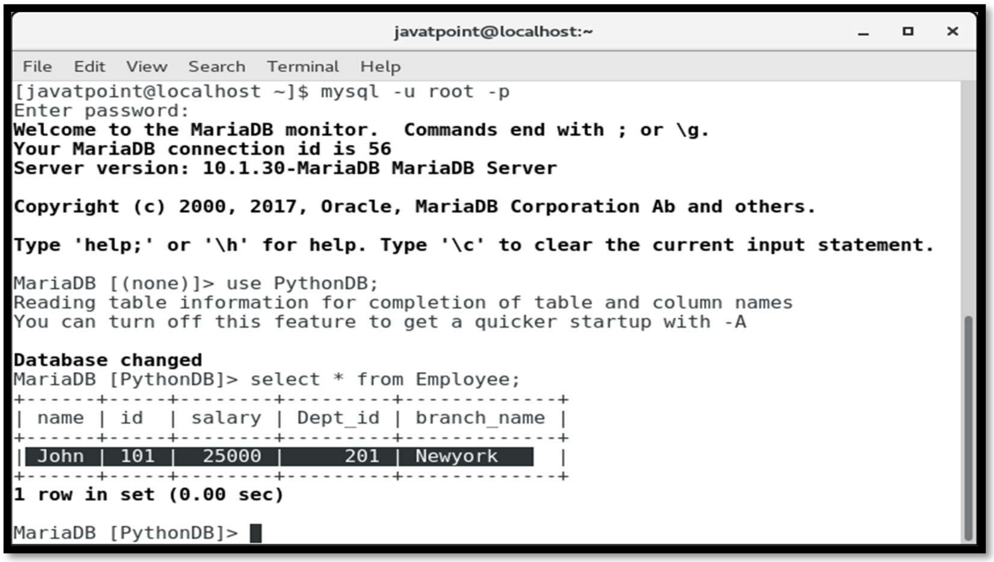
The **INSERT INTO** statement is used to add a record to the table. In python, wecan mention the format specifier (%s) in place of values

- We provide the actual values in the form of tuple in the execute() method ofthe cursor.

Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
= "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
sql = "insert into Employee(name, id, salary, dept_id, branch_name) values (
%s, %s, %s, %s, %s)"
#The row values are provided in the form of tuple8.
val = ("John", 110, 25000.00, 201, "Newyork")

try:
    #inserting the values into the table
    cur.execute(sql,val)
    #commit the transaction
    myconn.commi
t()    except:
    myconn.rollback()
print(cur.rowcount,"record
inserted!")myconn.close()
```

Output :-



```
[javatpoint@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 56
Server version: 10.1.30-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use PythonDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> select * from Employee;
+------+-----+--------+---------+-------------+
| name | id  | salary | Dept_id | branch_name |
+------+-----+--------+---------+-------------+
| John | 101 |  25000 |     201 | Newyork     |
+------+-----+--------+---------+-------------+
1 row in set (0.00 sec)

MariaDB [PythonDB]>
```

1 record inserted!

Insert multiple rows

- We can also insert multiple rows at once using the python script.
- The multiple rows are mentioned as the list of various tuples.
- Each element of the list is treated as one particular row, whereas each element of the tuple is treated as one particular column value (attribute).

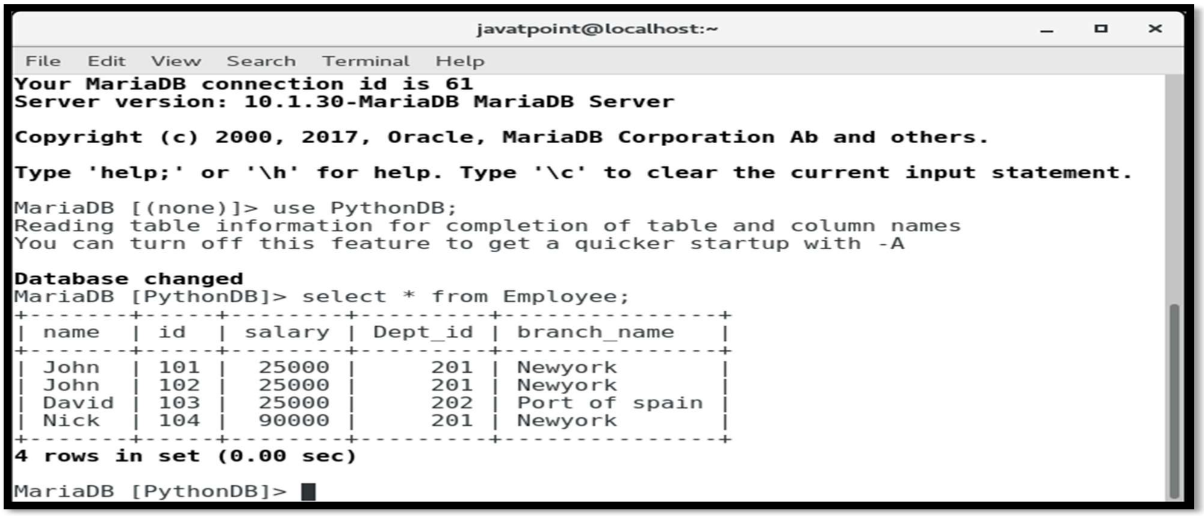
Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
```

```
sql = "insert into Employee(name, id, salary, dept_id, branch_name) values (
 %s, %s, %s, %s, %s)"


val = [("John", 102, 25000.00, 201, "Newyork"),("David",103,25000.00,202,"P
  ort of spain"),("Nick",104,90000.00,201,"Newyork")]
try:
    #inserting the values into the table
    cur.executemany(sql,val)
    #commit the transaction
    myconn.commit()
    print(cur.rowcount,"records inserted!")
except:
    myconn.rollback()
 myconn.close()
```

Output :-

3 records inserted!



```
Your MariaDB connection id is 61
Server version: 10.1.30-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use PythonDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> select * from Employee;
+-------+-----+--------+---------+---------------+
| name  | id  | salary | Dept_id | branch_name   |
+-------+-----+--------+---------+---------------+
| John  | 101 |  25000 |     201 | Newyork       |
| John  | 102 |  25000 |     201 | Newyork       |
| David | 103 |  25000 |     202 | Port of spain |
| Nick  | 104 |  90000 |     201 | Newyork       |
+-------+-----+--------+---------+---------------+
4 rows in set (0.00 sec)

MariaDB [PythonDB]> 
```

Row ID

- In SQL, a particular row is represented by an insertion id which is known asrow id.
- We can get the last inserted row id by using the attribute last rowid of thecursor object.

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
= "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
sql = "insert into Employee(name, id, salary, dept_id, branch_name) values (
%s, %s, %s, %s, %s)"

val = ("Mike",105,28000,202,"Guyana")
try:
    #inserting the values into the table

    cur.execute(sql,val)
    #commit the transaction
    myconn.commit()
    #getting rowid
    print(cur.rowcount,"recordinserted! id:",cur.lastrowid)
    except:
myconn.rollback
()myconn.close()
```

Output :-

```
1 record inserted! Id: 0
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	The________statement is used to add a record to the table.	insert into
2	- In SQL, a particular row is represented by an insertion id which is known as_______	row id
3	We can also insert multiple rows at once using________.	python script

Q-8Write note on Retrieving all the rows from a table.

Detail :-

Read Operation

- You can use either **fetchone()** method to fetch single record or **fetchall()** method to fetech multiple values from a database table.

- **fetchone()** − It fetches the next row of a query result set. A result set is anobject that is returned when a cursor object is used to query a table.
- **fetchall()** − It fetches all the rows in a result set. If some rows have already been extracted from the result set, then it retrieves the remaining rows fromthe result set.
- **rowcount** − This is a read-only attribute and returns the number of rows thatwere affected by an execute() method

- The SELECT statement is used to read the values from the databases. We canrestrict the output of a select query by using various clause in SQL like where,limit, etc.
- Python provides the fetchall() method returns the data stored inside the tablein the form of rows. We can iterate the result to get the individual rows.

- In this section of the tutorial, we will extract the data from the database byusing the python script. We will also format the output to print it on the console.

Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",
passwd= "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
    #Reading the Employee data
    cur.execute("select * from Employee")
    #fetching the rows from the cursor object
    result = cur.fetchall()
    #printing the result
    for x in result:
        print(x);
except:
myconn.rollback()
myconn.close()
```

/*Output:

```
('John', 101, 25000.0, 201, 'Newyork')
('John', 102, 25000.0, 201, 'Newyork')
('David', 103, 25000.0, 202, 'Port of spain')
('Nick', 104, 90000.0, 201, 'Newyork')
('Mike', 105, 28000.0, 202, 'Guyana')
```

Reading specific columns

- We can read the specific columns by mentioning their names instead of usingstar (*).

- In the following example, we will read the name, id, and salary from theEmployee table and print it on the console.

Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",
passwd= "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()


try:
    #Reading the Employee data
    cur.execute("select name, id, salary from Employee")

    #fetching the rows from the cursor object
    result = cur.fetchall()
    #printing the result
    for x in result:
        print(x);
except:

    myconn.rollbac
k()17.myconn.close()
```

Output :-

```
('John', 101, 25000.0)
('John', 102, 25000.0)
('David', 103, 25000.0)
('Nick', 104, 90000.0)
('Mike', 105, 28000.0)
```

The fetchone() method

- The fetchone() method is used to fetch only one row from the table.
- The fetchone() method returns the next row of the result-set.

Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
    #Reading the Employee data
    cur.execute("select name, id, salary from Employee")
    #fetching the first row from the cursor object
    result = cur.fetchone()
    #printing the result
    print(result)

except:
    myconn.rollback()
    myconn.close()
```

Output :-

```
'John', 101, 25000.0)
```

Formatting the result

- We can format the result by iterating over the result produced by the fetchall() or fetchone() method of cursor object since the result exists as the tuple objectwhich is not readable.

Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
= "google",database = "PythonDB")
#creating the cursor object

cur = myconn.cursor()
try:
    #Reading the Employee data
    cur.execute("select name, id, salary from Employee")
    #fetching the rows from the cursor object
    result = cur.fetchall()
    print("Name  id   Salary");
    for row in result:
        print("%s %d
        %d"%(row[0],row[1],row[2]))
except:
    myconn.rollback()
    myconn.close()
```

Output :-

```
Name   Id   Salary
John   101   25000
John   102   25000
David103     25000
```

```
Nick  104   90000
Mike 105     28000
```

Using where clause

- We can restrict the result produced by the select statement by using thewhere clause.
- This will extract only those columns which satisfy the where condition.

Example: printing the names that start with j

```
1.  import mysql.connector
2.  #Create the connection object
3.  myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
    = "google",database = "PythonDB")
4.  #creating the cursor object
5.  cur = myconn.cursor()

6.  try:
7.     #Reading the Employee data
8.     cur.execute("select name, id, salary from Employee where name like 'J%'")

9.     #fetching the rows from the cursor object
10.    result = cur.fetchall()
11.    print("Name  id   Salary");
12.    for row in result:
13.       print("%s %d
%d"%(row[0],row[1],row[2])) 14.except:
15.myconn.rollback
()16.myconn.close()
```

```
Name   id
Salary      John
          101
          25000
John   102   25000
```

Example: printing the names with id = 101, 102, and 103

Example :-

```
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:

   #Reading the Employee data
   cur.execute("select name, id, salary from Employee where id in (101,102,103)")
   #fetching the rows from the cursor object
   result = cur.fetchall()
   print("Name  id   Salary");
   for row in result:
      print("%s %d %d"%(row[0],row[1],row[2]))
except:
   myconn.rollback()


myconn.close()
```

Ordering the result

- The ORDER BY clause is used to order the result.

Example

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
   #Reading the Employee data
```

```
cur.execute("select name, id, salary from Employee order by name")
    #fetching the rows from the cursor object
    result = cur.fetchall()
    print("Name  id   Salary");
    for row in result:
        print("%s  %d      %d"%(row[0],row[1],row[2]))
except:
myconn.rollback
()myconn.close()
```

Output:

```
Name   id
       Salary
David  103
       25000
John   101   25000
John   102   25000
Mike   105   28000
 Nick  104    90000
```

Order by DESC

- This orders the result in the decreasing order of a particular column.

Example

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
= "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
    #Reading the Employee data
    cur.execute("select name, id, salary from Employee order by name desc")
    #fetching the rows from the cursor object
    result = cur.fetchall()
    #printing the result
    print("Name  id   Salary");
    for row in result:
print("%s        %d %d"%(row[0],row[1],row[2])) \

except:
    myconn.rollback()
myconn.close()
```

Output:

```
Name   id   Salary
Nick   104   90000
Mike   105   28000
John   101   25000
John   102   25000
David   103   25000
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	_________method can be used to fetch single record	fetchone()
2	______________method fetches all the rows in a result set	fetchall()
3	____________method is used to returns the number of rows that were affected.	rowcount()

Q-9Write note on Updating Rows in a table.

Detail :-

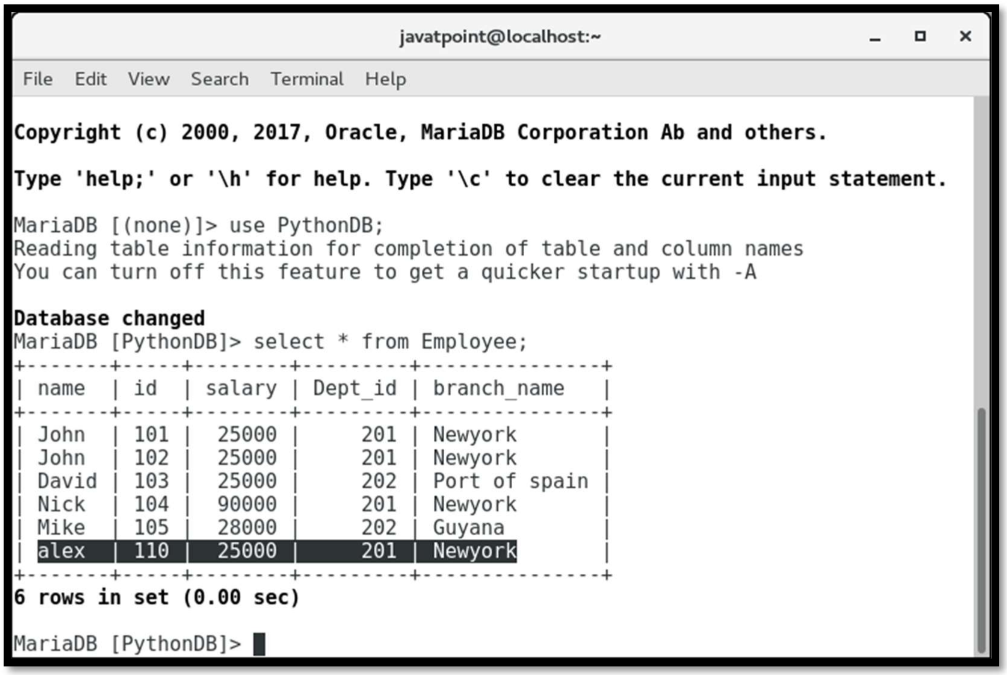
Update Operation

- The UPDATE-SET statement is used to update any column inside the table.
- The following SQL query is used to update a column.
- UPDATE Operation on any database means to update one or more records,which are already available in the database.

> update Employee set name = 'alex' where id = 110

Example :-

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",
passwd= "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
    #updating the name of the employee whose id is 110
    cur.execute("update Employee set name = 'alex' where id = 110")
    myconn.commit()
    except:
    myconn.rollback()
    myconn.close()
```



```
Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use PythonDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> select * from Employee;
+-------+-----+--------+---------+---------------+
| name  | id  | salary | Dept_id | branch_name   |
+-------+-----+--------+---------+---------------+
| John  | 101 |  25000 |     201 | Newyork       |
| John  | 102 |  25000 |     201 | Newyork       |
| David | 103 |  25000 |     202 | Port of spain |
| Nick  | 104 |  90000 |     201 | Newyork       |
| Mike  | 105 |  28000 |     202 | Guyana        |
| alex  | 110 |  25000 |     201 | Newyork       |
+-------+-----+--------+---------+---------------+
6 rows in set (0.00 sec)

MariaDB [PythonDB]> 
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	_______statement is used to update any column inside the table.	UPDATE-SET
2	________method can be used to save the updation on record.	commit()

Q-10Write note on Deleting Rows from a table.

Detail :-

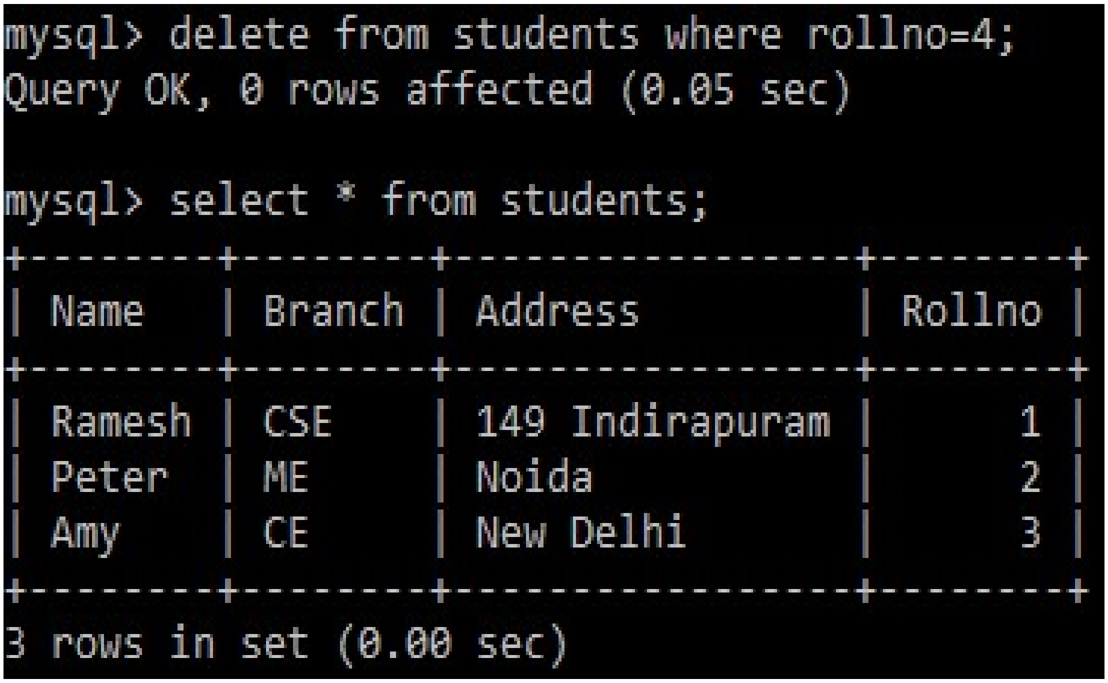
Delete Operation

- The DELETE FROM statement is used to delete a specific record from the table. Here, we must impose a condition using WHERE clause otherwise all the records from the table will be removed.
- The following SQL query is used to delete the employee detail whose id is 110from the table.

> delete from students where rollno = 4

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
    #Deleting the student details whose rollno is 4
```

```
cur.execute("delete from students where rollno = 4")
myconn.commit()

except:
myconn.rollback
()myconn.close()
```



```
mysql> delete from students where rollno=4;
Query OK, 0 rows affected (0.05 sec)

mysql> select * from students;
+--------+--------+-------------------+--------+
| Name   | Branch | Address           | Rollno |
+--------+--------+-------------------+--------+
| Ramesh | CSE    | 149 Indirapuram   |      1 |
| Peter  | ME     | Noida             |      2 |
| Amy    | CE     | New Delhi         |      3 |
+--------+--------+-------------------+--------+
3 rows in set (0.00 sec)
```

1 Word Question – Answer

SR.NO	QUESTION	ANSWER
1	The_____________statement is used to delete a specific record from the table.	DELETE FROM
2	__________clause must be used to remove or delete particular record from the table.	WHERE

PYTHON PROGRAM CODE WITH ANSWER :-

PROGRAM :- PYTHON PROGRAM TO PRINT DIFFERENT MESSAGES USING PRINT().

```
# it will print new line after the messages
print("Hello")
print("World")

# it will print new line
print()

# it will print new line after printing "Hello"
print("Hello",end="\n")
# it willprint new line after printing "World"
print("World")

# it will print new line
print()

# it will not print new line after printing "Hello"
# it will print space " "
print("Hello",end=" ")
# it will print new line after printing "World"
print("World")
```

Output

```
Hello
World

Hello
World

Hello World
```

PROGRAM :- PYTHON | PRINTING DIFFERENT VALUES (INTEGER, FLOAT, STRING, BOOLEAN)

```
# printing integer value
print(12)
# printing float value
print(12.56)
# printing string value
print("Hello")
# printing boolean value
print(True)
```

Output

```
12
12.56
Hello
True
```

Printing different types of variables along with the messages

```
# variable with integer value
a=12
# variable with float value
b=12.56
# variable with string value
c="Hello"
# variable with Boolean value
d=True
# printing values with messages
print("Integer\t:",a)
print("Float\t:",b)
print("String\t:",c)
print("Boolean\t:",d)
```

Output

```
Integer : 12
Float   : 12.56
String  : Hello
Boolean : True
```

PROGRAM :-PYTHON PROGRAM TO DEMONSTRATE VARIABLES SCOPE

```
# Python code to demonstrate example
# of variable scopes

# global variable
a = 100

# defining a function to test scopes
def func():
    # local variable
    b = 200

    # printing the value of global variable (a)
    # and, local variable (b)
    print("a: ", a, "b: ", b)

# main code
if __name__ == '__main__':
    # local variable of main
    c = 200

    # printing values of a, b and c
    print("a: ", a) #global
    # print("a: ", b) #local of text *** will give an error
    print("c: ", c) # local to main

    # calling the function
    func()
    # updating the value of global variable 'a'
    a = a+10

    # printing 'a' again
    print("a: ", a) #global
```

Output

```
a:  100
c:  200
a:  100 b:  200
a:  110
```

PROGRAM :-PYTHON CODE TO DETERMINE THE TYPE OF OBJECTS.

```
# Python code to determine the type of objects

# declaring objects and assigning values
a = 10
b = 10.23
c = "Hello"
d = (10, 20, 30, 40)
e = [10, 20, 30, 40]

# printing types of the objects
# using type() function
print("type(a): ", type(a))
print("type(b): ", type(b))
print("type(c): ", type(c))
print("type(d): ", type(d))
print("type(e): ", type(e))
# printing the type of the value
# using type() function
print("type(10): ", type(10))
print("type(10.23): ", type(10.23))

print("type(\"Hello\"): ", type("Hello"))
print("type((10, 20, 30, 40)): ", type((10, 20, 30, 40)))
print("type([10, 20, 30, 40]): ", type([10, 20, 30, 40]))
```

Output

```
type(a):  <class 'int'>
type(b):  <class 'float'>
type(c):  <class 'str'>
type(d):  <class 'tuple'>
type(e):  <class 'list'>
type(10):  <class 'int'>
type(10.23):  <class 'float'>
type("Hello"):  <class 'str'>
type((10, 20, 30, 40)):  <class 'tuple'>
type([10, 20, 30, 40]):  <class 'list'>
```

PROGRAM :-PYTHON | TYPECASTING INPUT TO INTEGER, FLOAT

```
# input a number
num = int(input("Input a value: "))

# printing input value
print ("num = ", num)
```

Output

```
Input a value: 10
num =  10
```

```
# input a number
num = float(input("Input a value: "))

# printing input value
print (“num = “, num)
```

Output

```
Input a value: 10.23
num =  10.23
```

PROGRAM :- PYTHON PROGRAM TO FIND SUM OF TWO NUMBERS

```
# python program to find sum of
# two numbers

num1 = 10
num2 = 20

# finding sum
sum = num1 + num2

# printing sum
print("sum of ", num1, " and ", num2, " is = ", sum)

# taking input from user
num1 = input("Enter first  number: ")
num2 = input("Enter second number: ")
# finding sum
sum = int(num1) + int(num2)
# printing sum
print("sum of ", num1, " and ", num2, " is = ", sum)
```

Output

```
sum of  10  and  20  is =  30
Enter first  number: 100
Enter second number: 200
sum of  100  and  200  is =  300
```

PROGRAM :- PYTHON PROGRAM TO FIND SIMPLE INTEREST.

```
# Python program to find simple interest
```

```
p = float(input("Enter the principle amount : "))
r = float(input("Enter the rate of interest : "))
t = float(input("Enter the time in the years: "))

# calculating simple interest
si = (p*r*t)/100

# printing the values
print("Principle amount: ", p)
print("Interest rate   : ", r)
print("Time in years   : ", t)
print("Simple Interest : ", si)
```

Output

```
First run:
Enter the principle amount : 10000
Enter the rate of interest : 3.5
Enter the time in the years: 1
Principle amount:  10000.0
Interest rate   :  3.5
Time in years   :  1.0
Simple Interest :  350.0
Second run:
Enter the principle amount : 250000
Enter the rate of interest : 36
Enter the time in the years: 1


Principle amount:  250000.0
Interest rate   :  36.0
Time in years   :  1.0
Simple Interest :  90000.0
```

PROGRAM :- PYTHON PROGRAM TO CHECK THE GIVEN YEAR IS A LEAP YEAR OR NOT.

```
# input the year
```

```
y=int(input('Enter the value of year: '))

# To check for non century year
if y%400==0 or y%4==0 and y%100!=0:
    print('The given year is a leap year.')
else:
    print('The given year is a non-leap year.')
```

Output

```
RUN 1:
Enter the value of year: 2020
The given year is a leap year.

RUN 2:
Enter the value of year: 2000
The given year is a leap year.
```

PROGRAM :- PYTHON PROGRAM TO CHECK THE GIVEN NUMBER IS ODD OR EVEN.

```
# Python function to check EVEN or ODD
def CheckEvenOdd(num):

  if (num % 2 == 0):
    print(num," is EVEN")
  else:
    print(num," is ODD")

# main code
CheckEvenOdd(11)
CheckEvenOdd(22)
CheckEvenOdd(33)
CheckEvenOdd(44)
```

Output

```
11  is ODD
22  is EVEN
33  is ODD
44  is EVEN
```

PROGRAM:- PYTHON PROGRAM TO INPUT AGE AND CHECK ELIGIBILITY FOR VOTING[IF….ELSE].

```
# input age
age = int(input("Enter Age : "))

# condition to check voting eligibility
if age>=18:
        status="Eligible"
else:
    status="Not Eligible"

print("You are ",status," for Vote.")
```

Output

```
Enter Age : 19
You are  Eligible  for Vote.
```

PROGRAM:- PYTHON PROGRAM TO DESIGN A SIMPLE CALCULATOR USING IF ELIF (JUST LIKE SWITCH CASE)

```
# menus
print("Calculator")
print("1.Add")
print("2.Substract")
print("3.Multiply")
print("4.Divide")

# input choice
ch=int(input("Enter Choice(1-4): "))

if ch==1:
    a=int(input("Enter A:"))
    b=int(input("Enter B:"))
    c=a+b
    print("Sum = ",c)
elif ch==2:
    a=int(input("Enter A:"))
    b=int(input("Enter B:"))
    c=a-b
    print("Difference = ",c)
elif  ch==3:
    a=int(input("Enter A:"))
    b=int(input("Enter B:"))
    c=a*b
    print("Product = ",c)
elif ch==4:
    a=int(input("Enter A:"))
    b=int(input("Enter B:"))
    c=a/b
    print("Quotient = ",c)
else:
    print("Invalid Choice")
```

Output

```
Calculator


1.Add
2.Substract
3.Multiply
4.Divide
Enter Choice(1-4): 3
Enter A:10
Enter B:20
Product =  200
```

PROGRAM:- PYTHON PROGRAM TO DEMONSTRATE AN EXAMPLE OF FOR LOOP.

```
print("Type 1")
for i in range(10):  # start=0 , end=10,step=1
   print(i,end="  ")

print("\nType 2")
for i in range(1,11):  # start=1 , end=10,step=1
   print(i,end="  ")
print("\nType 3")
for i in range(1,11,3):  # start=1 , end=10,step=3
   print(i,end="  ")
print("\nType 4")
for i in range(10,0,-1):  # start=10 , end=0,step=-1
   print(i,end="  ")
```

Output

```
Type 1
0  1  2  3  4  5  6  7  8  9
Type 2
1  2  3  4  5  6  7  8  9  10
Type 3
1  4  7  10
Type 4
10  9  8  7  6  5  4  3  2  1
```

PROGRAM :- PYTHON PROGRAM TO DEMONSTRATE SOME OF THE EXAMPLES OF LOOPS.

1. Print all the no. between 1 to n

```
n=int(input("Enter N: "))

for i in range(1,n+1):
   print(i)
```

Output

```
Enter N: 5
1
2
3
4
5
```

2. Print table of number

```
n=int(input("Enter N: "))

for i in range(1,11):
   print(n,"x",i,"=",i*n)
```

Output

```
Enter N: 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

3. Print sum of n number

```
n=int(input("Enter N: "))
s=0
for i in range(1,n+1):
   s=s+i
print("Sum = ",s)
```

Output

```
Enter N: 10
Sum = 55
```

4. Print factorial of n

```
n=int(input("Enter N: "))
f=1
for i in range(n,0,-1):
f=f*i
print("Factorial = ",f)
```

Output

```
Enter N: 4
Factorial =  24
```

PROGRAM:- TO DEMONSTRATE AN EXAMPLE OF FOR LOOP.

```
# declare and initialize a list
fruits = ["apple","mango","guava","grapes","pinapple"]

# pritning type of fruits
print (type(fruits))

# printing value
for fruit in fruits:
    print(fruit)
```

Output

```
<class 'list'>
apple
mango
guava
grapes
pinapple
```

PROGRAM :- TO DEMONSTRATE AN EXAMPLE OF WHILE LOOP TO PRINT 1 TO 5.

Example 1:

```
i=1
While (i<=5):
        Print(i)
        i=i+1
```

Output

```
1
2
3
4
5
```

PROGRAM :- TO DEMONSTRATE AN EXAMPLE OF BREAK STATEMENT.

Example 1:

```
for i in range(1,11):
   if(i==6):
      break
   print(i)
```

Output

```
1
2
3
4
5
```

Example 2: In this example, we are printing character by character of the value/string “Hello world” and terminating (using break), if the character is space.

```
for ch in "Hello world":
   if ch == " ":
      break
   print(ch)
```

Output

```
H
e
l
l
o
```

PROGRAM :- TO DEMONSTRATE AN EXAMPLE OF CONTINUE STATEMENT.

Example 1:

```
for i in range(1,11):
   if(i==6):
      continue
   print(i)
```

Output

```
1
2
3
4
5
7
8
9
10
```

Example 2: In this example, we are printing character by character of the value/string "Hello world" and continuing the loop execution, if the character is space.

```
for ch in "Hello world":
    if ch == " ":
        continue
    print(ch)
```

Output

```
H
e
l
l
o
w
o
r
l
d
```

PROGRAM :- TO FIND THE FACTORIAL OF A NUMBER USING RECURSION.

```
# Python code to find factorial using recursion

# recursion function definition
# it accepts a number and returns its factorial
def factorial(num):
    # if number is negative - print error
    if num < 0:
        print("Invalid number...")
    # if number is 0 or 1 - the factorial is 1
    elif num == 0 or num == 1:
        return 1
    else:
        # calling function itself i.e. recursive
        return num * factorial(num - 1)
```

```
# main code
if __name__ == '__main__':
    #input the number
    x = int(input("Enter an integer number: "))
    print("Factorial of ", x, " is = ", factorial(x))

    x = int(input("Enter another integer number: "))
    print("Factorial of ", x, " is = ", factorial(x))

    x = int(input("Enter another integer number: "))
    print("Factorial of ", x, " is = ", factorial(x))
```

Output

```
Enter an integer number: 5
Factorial of  5  is =  120
Enter another integer number: 0
Factorial of  0  is =  1
Enter another integer number: -3
Invalid number...
Factorial of  -3  is =  None
```

PROGRAM :- WRITE A PYTHON PROGRAM TO GET CURRENT DATE.

```
# Python program to get current date

# importing the date class
# from datetime module
from datetime import date

# getting the current date
current_date = date.today()

# printing the date
print("Current date is: ", current_date)
```

Output

```
Current date is:  2020-03-09
```

PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A STOP WATCH.

```
# Python code for a stopwatch

# importing the time module
import time

print("Press ENTER to start the stopwatch")
print("and, press CTRL + C to stop the stopwatch")

# infinite loop
while True:
    try:
        input() #For ENTER
        start_time = time.time()
        print("Stopwatch started...")

    except KeyboardInterrupt:
        print("Stopwatch stopped...")
        end_time = time.time()
        print("The total time:", round(end_time - start_time, 2),"seconds")
        break # breaking the loop
```

Output

```
Press ENTER to start the stopwatch
and, press CTRL + C to stop the stopwatch

Stopwatch started...
^CStopwatch stopped...
The total time: 15.81 seconds
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DECLARE AND PRINT A LIST.

```
# declaring list with integer elements
list1 = [10, 20, 30, 40, 50]

# printing list1
print "List element are: ", list1


# printing elements of list1 by index
print "Element @ 0 index:", list1[0]
print "Element @ 1 index:", list1[1]
print "Element @ 2 index:", list1[2]
print "Element @ 3 index:", list1[3]
print "Element @ 4 index:", list1[4]

# declaring list with string elements
list2 = ["New Delhi", "Mumbai", "Chennai", "calcutta"]


# printing list2
print "List elements are: ", list2
#printing elements of list2 by index
print "Element @ 0 index:",list2 [0]
print "Element @ 1 index:",list2 [1]
print "Element @ 2 index:",list2 [2]
print "Element @ 3 index:",list2 [3]


print " " # prints new line

# declaring list with mixed elements
list3 = ["Amit Shukla", 21, "New Delhi", 9876543210]

#printing list3
print "List elements are: ", list3
```

```
# printing elements of list3 by index
print "Element @ 0 index (Name) :", list3[0]
print "Element @ 1 index (Age ) :", list3[1]
print "Element @ 2 index (City) :", list3[2]
print "Element @ 3 index (Mob.) :", list3[3]
print "" # prints new line
```

Output

```
List element are:  [10, 20, 30, 40, 50]
Element @ 0 index: 10
Element @ 1 index: 20
Element @ 2 index: 30
Element @ 3 index: 40
Element @ 4 index: 50
List elements are:  ['New Delhi', 'Mumbai', 'Chennai', 'calcutta']
Element @ 0 index: New Delhi
Element @ 1 index: Mumbai
Element @ 2 index: Chennai
Element @ 3 index: Calcutta
List elements are:  ['Amit Shukla', 21, 'New Delhi', 9876543210
Element @ 0 index (Name) : Amit Shukla
Element @ 1 index (Age ) : 21
Element @ 2 index (City) : New Delhi
Element @ 3 index (Mob.) : 9876543210
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DECLARE AND PRINT A DICTIONARY.

```
# Adding elements to dictionary one at a time
Dict[0] = 'Peter'
Dict[2] = 'Joseph'
Dict[3] = 'Ricky'
print("\nDictionary after adding 3 elements: ")
print(Dict)

```

```
# Adding set of values
# with a single Key
# The Emp_ages doesn't exist to dictionary
Dict['Emp_ages'] = 20, 33, 24
print("\nDictionary after adding 3 elements: ")
print(Dict)

# Updating existing Key's Value
Dict[3] = 'JavaTpoint'
print("\nUpdated key value: ")
print(Dict)
```

Output

```
Dictionary after adding 3 elements:
 {0: 'Peter', 2: 'Joseph', 3: 'Ricky'}

 Dictionary after adding 3 elements:
 {0: 'Peter', 2: 'Joseph', 3: 'Ricky', 'Emp_ages': (20, 33, 24)}

 Updated key value:
 {0: 'Peter', 2: 'Joseph', 3: 'JavaTpoint', 'Emp_ages':
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DECLARE AND PRINT A TUPLE .

1.# Python program to show how to access tuple elements

# Creating a tuple

```
tuple_ = ("Python", "Tuple", "Ordered", "Collection")

print(tuple_[0])
print(tuple_[1])
# trying to access element index more than the length of a tuple
try:
    print(tuple_[5])
```

```
except Exception as e:
    print(e)
# trying to access elements through the index of floating data type
try:
    print(tuple_[1.0])
except Exception as e:
    print(e)

# Creating a nested tuple
nested_tuple = ("Tuple", [4, 6, 2, 6], (6, 2, 6, 7))
# Accessing the index of a nested tuple
print(nested_tuple[0][3])
print(nested_tuple[1][1])
```

Output

```
Python
Tuple
tuple index out of range
tuple indices must be integers or slices, not float
  1
  6
```

PROGRAM :- WRITE A PYTHON PROGRAM TO ACCESS AND PRINT CHARACTERS FROM THE STRING.

```
# access characters in string

# declare, assign string
str = "Hello world"

# print complete string
print "str:", str

# print first character
```

```
print "str[0]:", str[0]

# print second character
print "str[1]:", str[1]

# print last character
print "str[-1]:", str[-1]

# print second last character
print "str[-2]:", str[-2]


# print characters from 0th to 4th index i.e.
# first 5 characters
print "str[0:5]:", str[0:5]

# print characters from 2nd index to 2nd last index
print "str[2,-2]:", str[2:-2]

# print string character by character
print "str:"
for i in str:
        print i,
#comma after the variable
# it does not print new line
```

Output

```
str: Hello world
str[0]: H
str[1]: e
str[-1]: d
str[-2]: l
str[0:5]: Hello
str[2,-2]: llo wor
str:
H e l l o   w o r l d
```

PROGRAM :- WRITE A PYTHON PROGRAM TO PRINT WORDS WITH THEIR LENGTH OF A STRING.

```
# Function to split into words
# and print words with its length

def splitString (str):
        # split the string by spaces
        str = str.split (' ')
        # iterate words in string
        for words in str:


                print words," (", len (words), ")"

# Main code
# declare string and assign value
str = "Hello World How are you?"

# call the function
splitString(str)
```

Output

```
Hello  ( 5 )
World  ( 5 )
How  ( 3 )



are  ( 3 )
you?  ( 4 )
```

PROGRAM :- WRITE A PYTHON PROGRAM TO COUNT VOWELS IN THE GIVEN STRING.

```
# count vowels in a string

# declare, assign string
str = "Hello world"
```

```
# declare count
count = 0

# iterate and check each character
for i in str:
        # check the conditions for vowels
        if( i=='A' or i=='a' or i=='E' or i=='e'
        or i=='I' or i=='i' or i=='O' or i=='o'
        or i=='U' or i=='u'):
                count +=1;

# print count
print "Total vowels are: ", count
```

Output

```
Total vowels are:  3
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DEMONSTRATE DIFFERENT FILE HANDLING METHODS

```
# opens the file file.txt in read mode
fileptr = open("file.txt","r")

if fileptr:
    print("file is opened successfully")
```

```
# opens the file file.txt in read mode
fileptr = open("file.txt","r")
    # open the file.txt in append mode. Create a new file if no such file exists.
fileptr = open("file2.txt", "w")

# appending the content to the file
fileptr.write("'''Python is the modern day language. It makes things so simple.
It is the fastest-growing programing language''')

```

```
# closing the opened the file
fileptr.close()

if fileptr:
    print("file is opened successfully")

#closes the opened file
fileptr.close()
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DEMONSTRATE AN EXAMPLE OF USER DEFINED FUNCTION.

```
def sp(str1 ,str2):
        Print("hello sp")Print(str1)
        Print(str2)
return

sp("good","morning")
```

Output:

```
good
morning
```

Example - 1

PROGRAM :- WRITE A PYTHON PROGRAM TO DEMONSTRATE AN EXAMPLE OF MODULE.

```
# importing the complete math module using *
from math import *

# accessing functions of math module without using the dot operator
print( "Calculating square root: ", sqrt(25) )
print( "Calculating tangent of an angle: ", tan(pi/6) ) # here pi is also imported
```

from the math module

Output:

```
Calculating square root:  5.0
Calculating tangent of an angle:  0.5773502691896257
```

Example – 2   [module – support.py]

```
Def  print_func(x):
        Print(“hello:”,x)
        Return
```

- ○ import module support

```
import support
Support.print_func(“zara”)
```

Output:

```
Hello:  zara
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DEMONSTRATE AN EXAMPLE OF EXCEPTION HANDLING.

```
# Python code to catch an exception and handle it using try and except codeblocks
a = ["Python", "Exceptions", "try and except"]
try:
   #looping through the elements of the array a, choosing a range that goes beyond the length of the array
    for i in range( 4 ):
        print( "The index and element from the array is", i, a[i] )
#if an error occurs in the try block, then except block will be executed by the
```

```
 Python interpreter
except:
   print ("Index out of range")
```

Output:

```
The index and element from the array is 0 Python
The index and element from the array is 1 Exceptions
The index and element from the array is 2 try and except
Index out of range
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DEMONSTRATE AN EXAMPLE OF ASSERTION OR ASSERT STATEMENT.

```
#Python program to show how to use assert keyword
# defining a function
def square_root( Number ):
    assert ( Number < 0), "Give a positive integer"
    return Number**(1/2)
#Calling function and passing the values
print( square_root( 36 ) )
print( square_root( -36 ) )
```

Output:

```
      7 #Calling function and passing the values
----> 8 print( square_root( 36 ) )
      9 print( square_root( -36 ) )

Input In [23], in square_root(Number)
      3 def square_root( Number ):
----> 4     assert ( Number < 0), "Give a positive integer"
      5     return Number**(1/2)

AssertionError: Give a positive integer
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DEMONSTRATE AN EXAMPLE OF CLASS AND OBJECT.

```
# Python program to demonstrate an
# example of class

class Message(object):
    def __init__(self):
        # assign none to variable
        self.msg = None

    def assignValue(self):
        # assign any value
        self.msg = "Hello World"

    def getValue (self,str):
        # assign variable with parameter
        self.msg = str

    def printValue(self):
        # print the value
        print "msg = ",self.msg

# Main code
# creating object of the class
# Here, M is object nae
M = Message()

# print value
print "value after init. the object..."

M.printValue();
# assign value
M.assignValue()
# print value
print "value after assignValue ()...."
M.printValue();

# assign value using arguemnt
M.getValue("How are you?")
```

```
print "value after getValue ()..."
M.printValue();
```

Output

```
value after init. the object...
msg =  None
value after assignValue ()....
msg =  Hello World
value after getValue ()...
msg =  How are you?
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DEMONSTRATE AN EXAMPLE OF PUBLIC VARIABLES IN PYTHON.

```
# Python example for public variables
class person:
        def __init__(self):
                # default values
                self.name = "XYZ"
                self.age = 0

        def printValues(self):
                print "Name: ",self.name
                print "Age : ",self.age

# Main code
# declare object
p = person()
# print
p.printValues();

# since variables are public by default
# we can access them directly here
p.name = "Amit"
p.age = 21
# print
p.printValues ();
```

Output

```
Name:  XYZ
Age :  0
Name:  Amit
Age :  21
```

PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE EMPLOYEE CLASS WITH SOME ATTRIBUTES AND METHOD.

```
# employee class code in Python
# class definition
class Employee:
    __id=0
    __name=""
    __gender=""
    __city=""
    __salary=0
        # function to set data
    def setData(self,id,name,gender,city,salary):
        self.__id=id
        self.__name = name
        self.__gender = gender
        self.__city = city
        self.__salary = salary

        # function to get/print data
    def showData(self):
        print("Id\t\t:",self.__id)
        print("Name\t:", self.__name)
        print("Gender\t:", self.__gender)
        print("City\t:", self.__city)
        print("Salary\t:", self.__salary)
# main function definition
def main():
    #Employee Object
    emp=Employee()

    emp.setData(1,'pankaj','male','delhi',55000)
    emp.showData()
if __name__=="__main__":
```

```
main()
```

Output

```
Id      : 1
Name    : pankaj
Gender  : male
City    : delhi
Salary  : 55000
```

PROGRAM :- WRITE A PYTHON PROGRAM

TO IMPLEMENT ABSTRACTION USING ABSTRACT CLASS.

```
#Abstract Class
class Vehicle:
    def start(self,name=""):
        print(name,"is Started")
    def acclerate(self,name=""):
        pass
    def park(self,name=""):
        pass
    def stop(self,name=""):
        print(name,"is stopped")

class Bike(Vehicle):
    def acclerate(self, name=""):
        print(name,"is accelrating @ 60kmph")
    def park(self, name=""):
        print(name,"is parked at two wheeler parking")

class Car(Vehicle):
    def acclerate(self, name=""):
        print(name,"is accelrating @ 90kmph")
    def park(self, name=""):


        print(name,"is parked at four wheeler parking")
```

```
def main():
    print("Bike Object")
    b=Bike()
    b.start("Bike")
    b.acclerate("Bike")
    b.park("Bike")
    b.stop("Bike")
    print("\nCar Object")
    c = Car()
    c.start("Car")


    c.acclerate("Car")
    c.park("Car")
    c.stop("Car")
if __name__=="__main__":main()
```

Output

```
Bike Object
Bike is Started
Bike is accelrating @ 60kmph
Bike is parked at two wheeler parking
Bike is stopped

Car Object
Car is Started
Car is accelrating @ 90kmph
Car is parked at four wheeler parking
Car is stopped
```

PROGRAM :- WRITE A PYTHON PROGRAM

TO IMPLEMENT SINGLE INHERITANCE IN PYTHON.

```
# Python code to demonstrate example of
# single inheritance
```

```
class Details:
    def __init__(self):
        self.__id="<No Id>"
        self.__name="<No Name>"
        self.__gender="<No Gender>"
    def setData(self,id,name,gender):
        self.__id=id
        self.__name=name
        self.__gender=gender
    def showData(self):
        print("Id\t\t:",self.__id)
        print("Name\t\t:", self.__name)
        print("Gender\t\t:", self.__gender)

class Employee(Details): #Inheritance
    def __init__(self):
        self.__company="<No Company>"
        self.__dept="<No Dept>"
    def setEmployee(self,id,name,gender,comp,dept):
        self.setData(id,name,gender)
        self.__company=comp
        self.__dept=dept
    def showEmployee(self):
        self.showData()
        print("Company\t\t:", self.__company)
        print("Department\t:", self.__dept)
def main():
    e=Employee()
    e.setEmployee(101,"Prem Sharma","Male","New Delhi",110065)
    e.showEmployee()

if __name__=="__main__":
    main()
```

Output

```
Id          : 101
Name            : Prem Sharma
Gender          : Male
```

PROGRAM :- WRITE A PYTHON PROGRAM

WITH TWO CHILD (DERIVED) CLASSES IN PYTHON.

```
# Python code to demonstrate example of
# single inheritance with two child classes

class Details:
   def __init__(self):
      self.__id=0
      self.__name=""
      self.__gender=""
   def setDetails(self):
      self.__id=int(input("Enter Id: "))
      self.__name=input("Enter Name: ")
      self.__gender=input("Enter gender: ")
   def showDetails(self):
      print("Id: ",self.__id)
      print("Name: ",self.__name)
      print("Gender: ",self.__gender)

class Employee(Details):
   def __init__(self):
      self.__company=""
      self.__desig=""
   def setEmployee(self):
      self.setDetails()
      self.__company=input("Enter Compmany Name: ")
      self.__desig=input("Enter Designation: ")
   def showEmployee(self):
      self.showDetails()
      print("Company: ",self.__company)

      print("Designation: ",self.__desig)

class Doctor(Details):
   def __init__(self):
      self.__hospital=""
```

```
        self.__dept=""
    def setDoctor(self):
        self.setDetails()
        self.__hospital=input("Enter Hospital Name: ")
        self.__dept=input("Enter Department: ")
    def showDoctor(self):
        self.showDetails()
        print("Hospital: ",self.__hospital)
        print("Department",self.__dept)

def main():
    print("Employee Object: ")
    e = Employee()
    e.setEmployee()
    e.showEmployee()
    print("\nDoctor Object: ")
    d=Doctor()
    d.setDoctor()
    d.showDoctor()

if __name__=="__main__":
    main()
```

Output

```
Employee Object:
Enter Id: 101
Enter Name: Prem Sharma


Enter gender: Male
Enter Compmany Name: IncludeHelp
Enter Designation: Technical writer
Id:  101
Name:  Prem Sharma
Gender:  Male
Company:  IncludeHelp
Designation:  Technical writer
Doctor Object:
Enter Id: 201
Enter Name: Amit Shukla
Enter gender: Male
Enter Hospital Name: APOLLO
Enter Department: Doctor
Id:  201
Name:  Amit Shukla
Gender:  Male
Hospital:  APOLLO
Department Doctor
```

PROGRAM :- WRITE A PYTHON PROGRAM WITH MULTIPLE INHERITANCE IN PYTHON.

```
# Python code to demonstrate example of
# multiple inheritance

class Personel:
    def __init__(self):
        self.__id=0
        self.__name=""
        self.__gender=""
    def setPersonel(self):
        self.__id=int(input("Enter Id: "))
        self.__name = input("Enter Name: ")
        self.__gender = input("Enter Gender: ")
    def showPersonel(self):
        print("Id: ",self.__id)
        print("Name: ",self.__name)
        print("Gender: ",self.__gender)

class Educational:
    def __init__(self):
        self.__stream=""

        self.__year=""
    def setEducational(self):
        self.__stream=input("Enter Stream: ")
        self.__year = input("Enter Year: ")
    def showEducational(self):
        print("Stream: ",self.__stream)
        print("Year: ",self.__year)
```

```
class Student(Personel,Educational):
    def __init__(self):
        self.__address = ""
        self.__contact = ""
    def setStudent(self):
        self.setPersonel()
        self.__address = input("Enter Address: ")
        self.__contact = input("Enter Contact: ")
        self.setEducational()

    def showStudent(self):
        self.showPersonel()
        print("Address: ",self.__address)
        print("Contact: ",self.__contact)
        self.showEducational()

def main():
    s=Student()
    s.setStudent()
    s.showStudent()
if __name__=="__main__":main()
```

Output

```
Enter Id: 101
Enter Name: Prem Sharma
Enter Gender: Male


Enter Address: Nehru Place, New Delhi
Enter Contact: 0123456789
Enter Stream: Computer Science

Enter Year: 2010
Id:  101
Name:  Prem Sharma
Gender:  Male
```

PROGRAM - WRITE A PYTHON PROGRAM FOR BINARY SEARCH.

```
def binary_search(l, num_find):
    '''
    This function is used to search any number.
    Whether the given number is present in the
    list or not. If the number is present in list
    the list it will return TRUE and FALSE otherwise.
    '''
    start = 0
    end = len(l) - 1
    mid = (start + end) // 2

    # We took found as False that is, initially
    # we are considering that the given number
    # is not present in the list unless proven
    found = False
    position = -1

    while start <= end:
        if l[mid] == num_find:
            found = True
            position = mid
            break

        if num_find > l[mid]:
            start = mid + 1

            mid = (start + end) // 2
        else:
            end = mid - 1
            mid = (start + end) // 2
    return (found, position)

    # Time Complexity : O(logn)


# main code
if __name__=='__main__':

    l = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
    num = 6
    found = binary_search(l, num)
    if found[0]:
        print('Number %d found at position %d'%(num, found[1]+1))
    else:
        print('Number %d not found'%num)
```

Output

```
Number 6 found at position 7
```

PROGRAM :- WRITE A PYTHON PROGRAM TO PERFORM BUBBLE SORTING.

```
# Creating a bubble sort function
def bubble_sort(list1):
    # Outer loop for traverse the entire list
    for i in range(0,len(list1)-1):
        for j in range(len(list1)-1):

            if(list1[j]>list1[j+1]):
                temp = list1[j]
                list1[j] = list1[j+1]
                list1[j+1] = temp
    return list1

```

```
list1 = [5, 3, 8, 6, 7, 2]
print("The unsorted list is: ", list1)
# Calling the bubble sort function
print("The sorted list is: ", bubble_sort(list1))
```

Output:

```
The unsorted list is:  [5, 3, 8, 6, 7, 2]
The sorted list is:  [2, 3, 5, 6, 7, 8]
```

PROGRAM :- WRITE A PYTHON PROGRAM TO COPY ODD LINES OF ONE FILE TO ANOTHER FILE IN PYTHON.

```
# opening the file
file1 = open('file1.txt', 'r')

# creating another file to store odd lines
file2 = open('file2.txt', 'w')
# reading content of the files
# and writing odd lines to another file
lines = file1.readlines()
type(lines)
for i in range(0, len(lines)):
        if(i % 2 != 0):
                file2.write(lines[i])

# closing the files
file1.close()

file2.close()
+51426

# opening the files and printing their content
file1 = open('file1.txt', 'r')
file2 = open('file2.txt', 'r')

# reading and printing the files content
str1 = file1.read()
str2 = file2.read()
```

```
print("file1 content...")
print(str1)

print() # to print new line

print("file2 content...")
print(str2)

# closing the files
file1.close()
file2.close()
```

Output

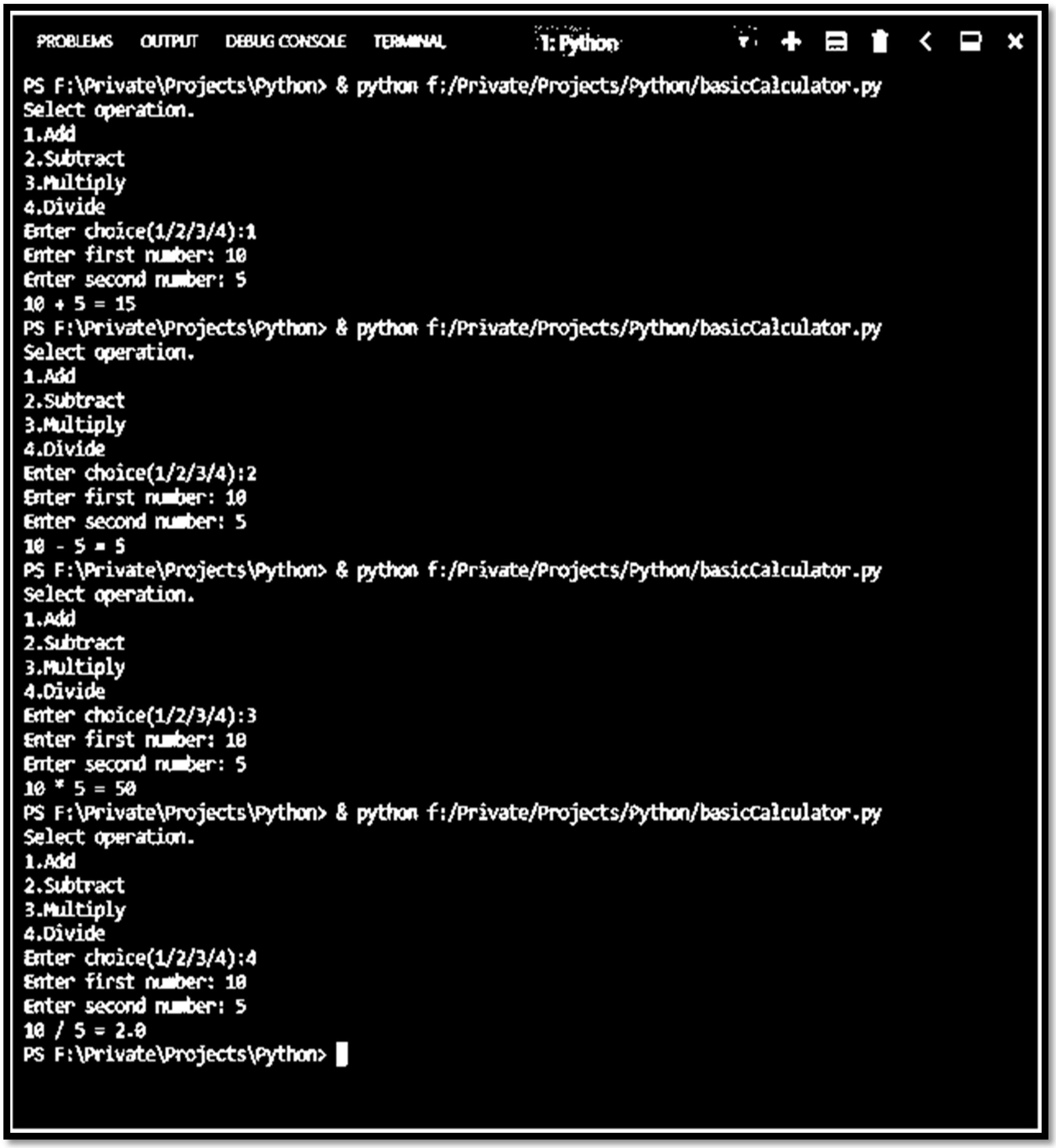
```
file1 content...
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DESIGN TRADITIONAL CALCULATOR IN PYTHON3.

```
# Program make a simple calculator that can
# add, subtract, multiply and divide using functions

# This function adds two numbers
def add(x, y):
  return x + y

# This function subtracts two numbers
def subtract(x, y):
  return x - y

# This function multiplies two numbers
def multiply(x, y):
  return x * y
```

```
# This function divides two numbers
def divide(x, y):
  return x / y

print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")

# Take input from the user
choice = input("Enter choice(1/2/3/4):")



num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

if choice == '1':
  print(num1,"+",num2,"=", add(num1,num2))

elif choice == '2':
  print(num1,"-",num2,"=", subtract(num1,num2))

elif choice == '3':
  print(num1,"*",num2,"=", multiply(num1,num2))

elif choice == '4':
  print(num1,"/",num2,"=", divide(num1,num2))
else:
  print("Invalid input")
```

Output



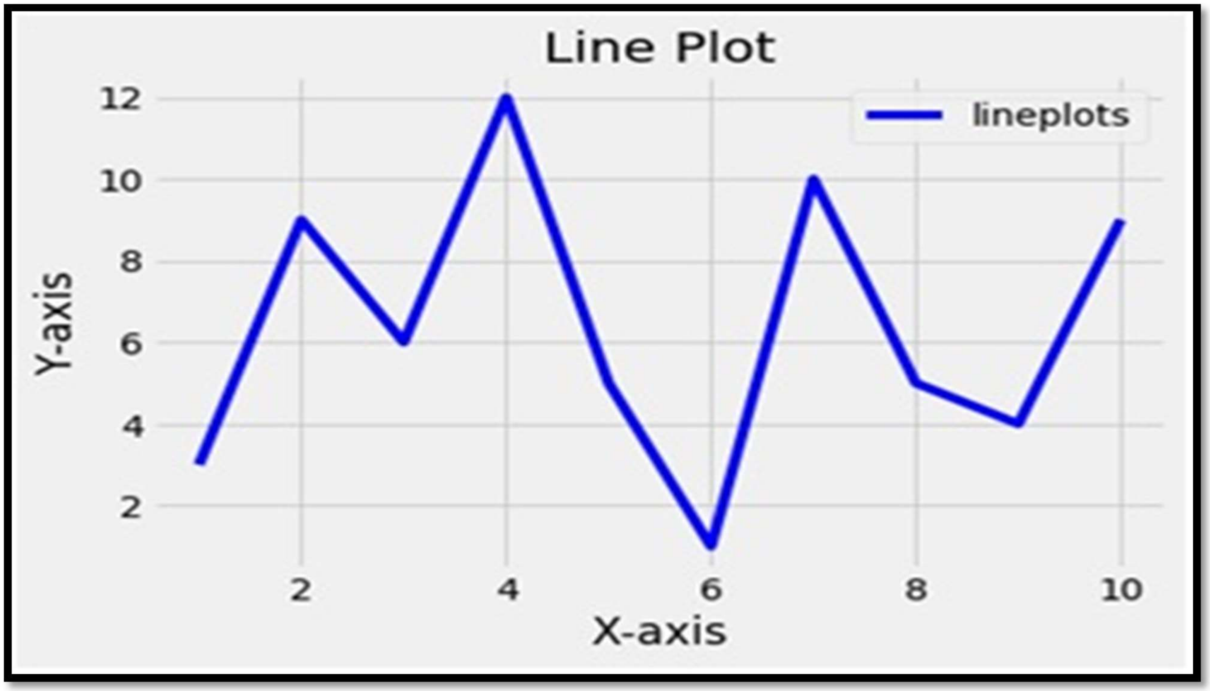
```
PS F:\Private\Projects\Python> & python f:/Private/Projects/Python/basicCalculator.py
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4):1
Enter first number: 10
Enter second number: 5
10 + 5 = 15
PS F:\Private\Projects\Python> & python f:/Private/Projects/Python/basicCalculator.py
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4):2
Enter first number: 10
Enter second number: 5
10 - 5 = 5
PS F:\Private\Projects\Python> & python f:/Private/Projects/Python/basicCalculator.py
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4):3
Enter first number: 10
Enter second number: 5
10 * 5 = 50
PS F:\Private\Projects\Python> & python f:/Private/Projects/Python/basicCalculator.py
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4):4
Enter first number: 10
Enter second number: 5
10 / 5 = 2.0
PS F:\Private\Projects\Python> █
```

PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A LINE PLOT USING MATPLOTLIB.PYPLOT.

```
import matplotlib.pyplot as plt
```

```
x = [1,2,3,4,5,6,7,8,9,10]
y = [3,9,6,12,5,1,10,5,4,9]
plt.plot(x,y, label='lineplots', color='b')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Line Plot')
plt.legend()
plt.show()
```

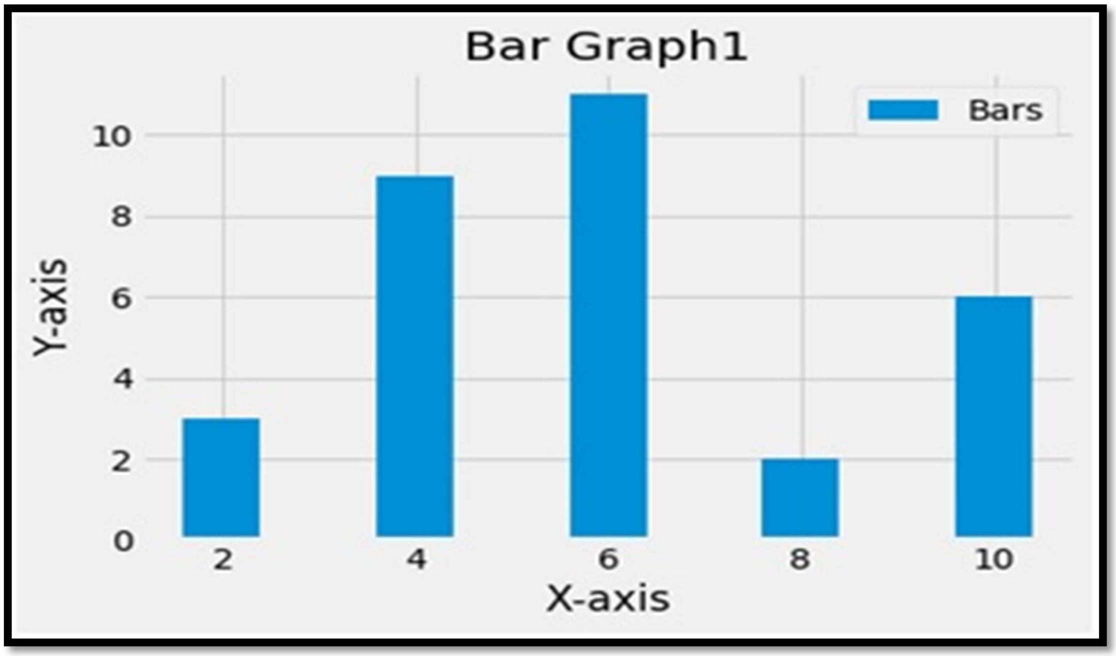
Output



PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A BAR CHART USING MATPLOTLIB.PYPLOT.

```
import matplotlib.pyplot as plt

x = [2,4,6,8,10]
y=[3,9,11,2,6]
plt.bar(x,y,label ='Bars')

plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Bar Graph1')
plt.legend()
plt.show()
```

Output

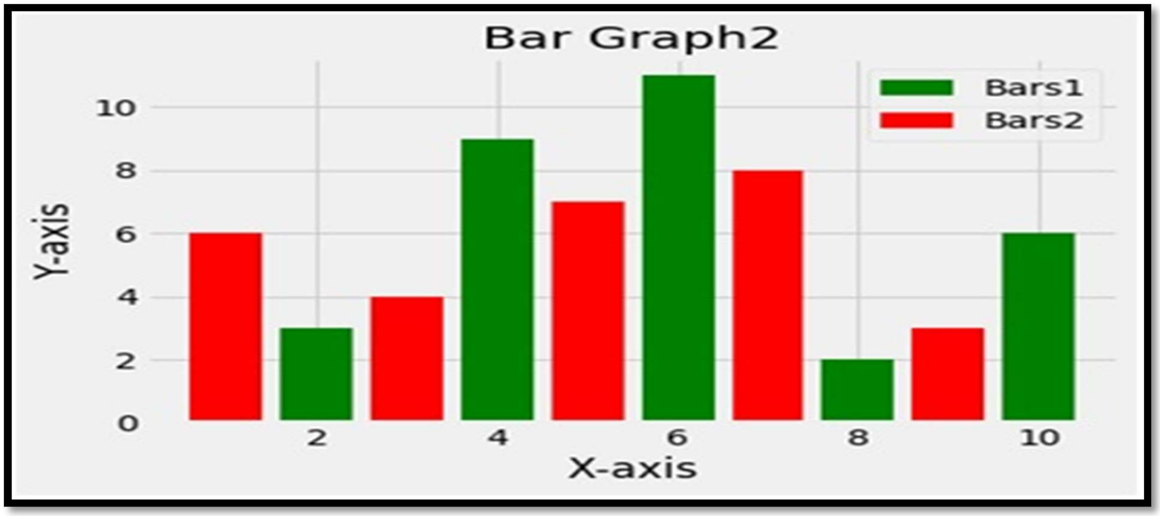


PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A BAR GRAPH USING MATPLOTLIB.PYPLOT.

```
import matplotlib.pyplot as plt
x1 = [2,4,6,8,10]
y1=[3,9,11,2,6]
```

```
x2=[1,3,5,7,9]
y2=[6,4,7,8,3]

plt.bar(x1,y1,label ='Bars1', color='g')
plt.bar(x2,y2,label = 'Bars2', color='r')

plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Bar Graph2')
plt.legend()
plt.show()
```

Output

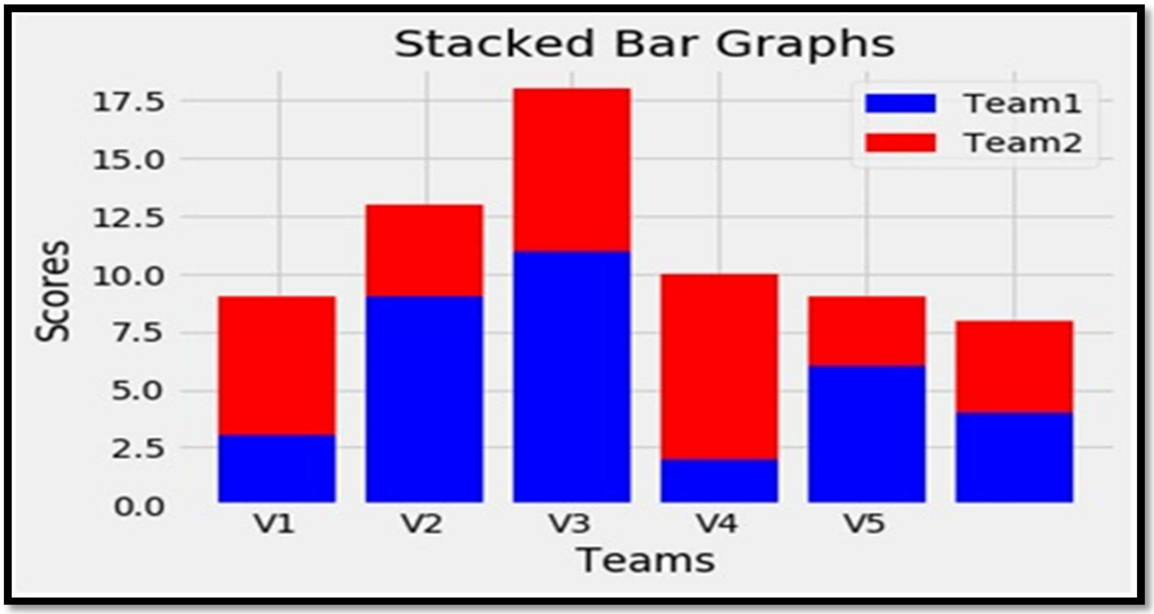


PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A STACKED BAR USING MATPLOTLIB.PYPLOT.

```
import matplotlib.pyplot as plt
import numpy as np
N=6
y1=[3,9,11,2,6,4]

y2=[6,4,7,8,3,4]
```

```
xvalues = np.arange(N)

plt.bar(xvalues,y1,color='b', label ='Team1')
plt.bar(xvalues,y2, color='r', bottom =y1, label = 'Team2')
plt.xticks(xvalues, ('V1', 'V2', 'V3', 'V4', 'V5'))

plt.xlabel('Teams')
plt.ylabel('Scores')
plt.title('Stacked Bar Graphs')
plt.legend()
```

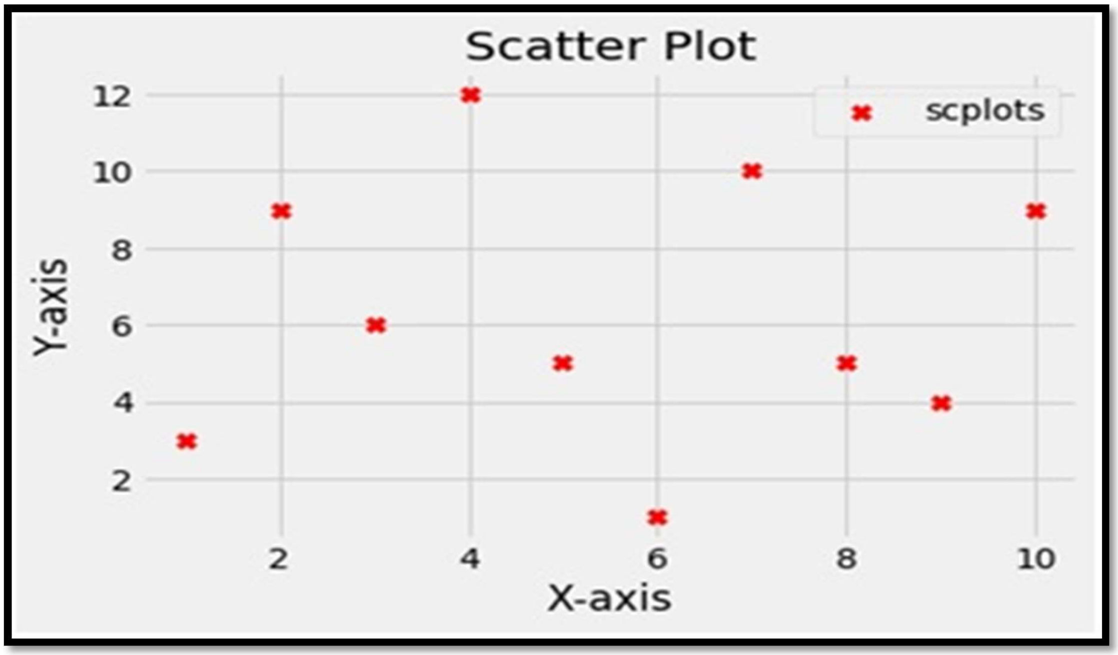
Output



PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A SCATTER PLOT USING MATPLOTLIB.PYPLOT.

```
import matplotlib.pyplot as plt

x = [1,2,3,4,5,6,7,8,9,10]
y = [3,9,6,12,5,1,10,5,4,9]

plt.scatter(x,y, label='scplots', color='r', s=60, marker="X")
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Scatter Plot')
plt.legend()
plt.show()
```

Output

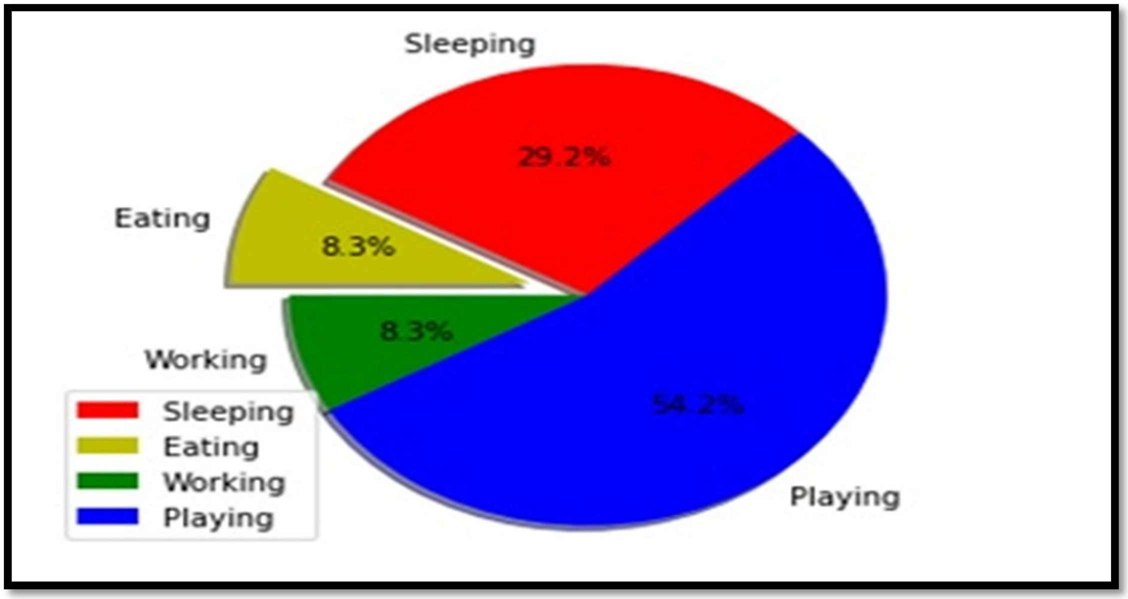


PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A PIE - CHART USING MATPLOTLIB.PYPLOT.

```
import matplotlib.pyplot as plt

days = [1, 2, 3, 4, 5]

slices = [7,2,2,13]
cols = ['r','y','g','b']


my_labels = ["Sleeping ", "Eating", "Working", "Playing"]
```

```
plt.pie(slices,
        labels=my_labels,
        colors = cols,
startangle=45,
        explode =(0,0.2,0,0),
        shadow = True,
autopct = '%1.1f%%')
plt.axis('equal')
plt.legend(loc=3)
plt.show()
```

Output

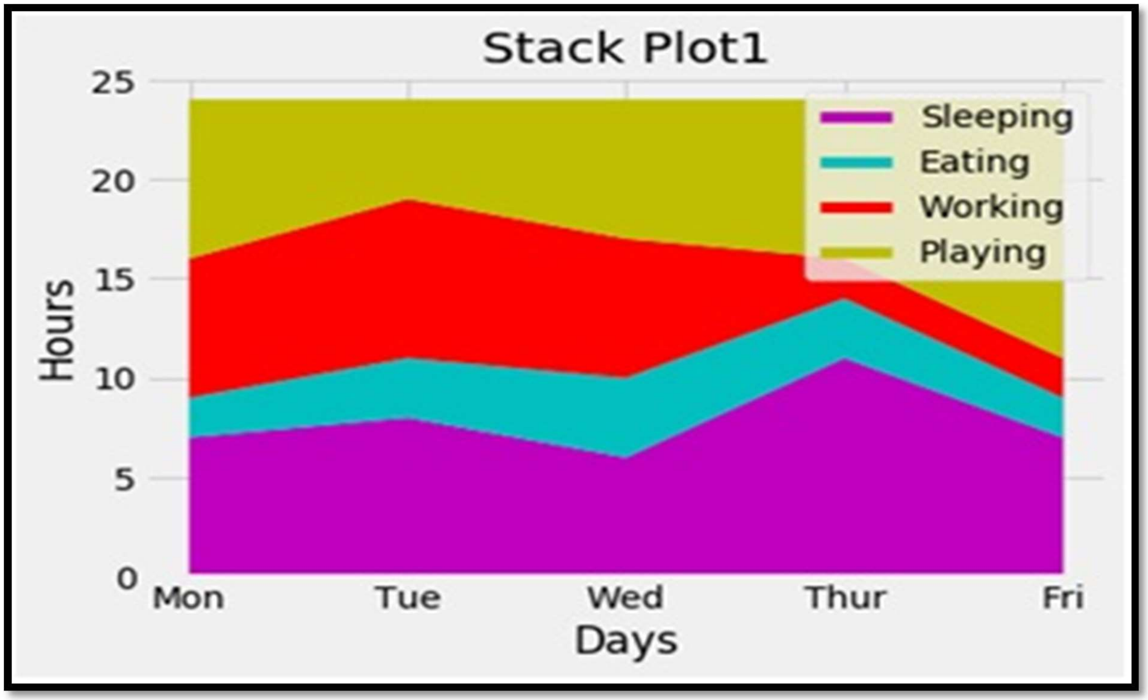


PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A STACK PLOT USING MATPLOTLIB.PYPLOT

```
import matplotlib.pyplot as plt

days=[1,2,3,4,5]

sleeping = [7,8,6,11,7]
eating = [2,3,4,3,2]
working = [7,8,7,2,2]
```

```
 playing = [8,5,7,8,13]

 plt.plot([],[],color ='m', label = 'Sleeping', linewidth = 5)
 plt.plot([],[],color ='c', label = 'Eating', linewidth = 5)
 plt.plot([],[],color ='r', label = 'Working', linewidth = 5)
 plt.plot([],[],color ='y', label = 'Playing', linewidth = 5)

 plt.stackplot(days, sleeping, eating, working, playing, colors =
['m','c','r','y'])

 plt.xlabel('Days')
 plt.ylabel('Hours')
 plt.title('Stack Plot1')
 plt.xticks(days, ('Mon', 'Tue', 'Wed', 'Thur', 'Fri'))

 plt.legend()
 plt.show()
```

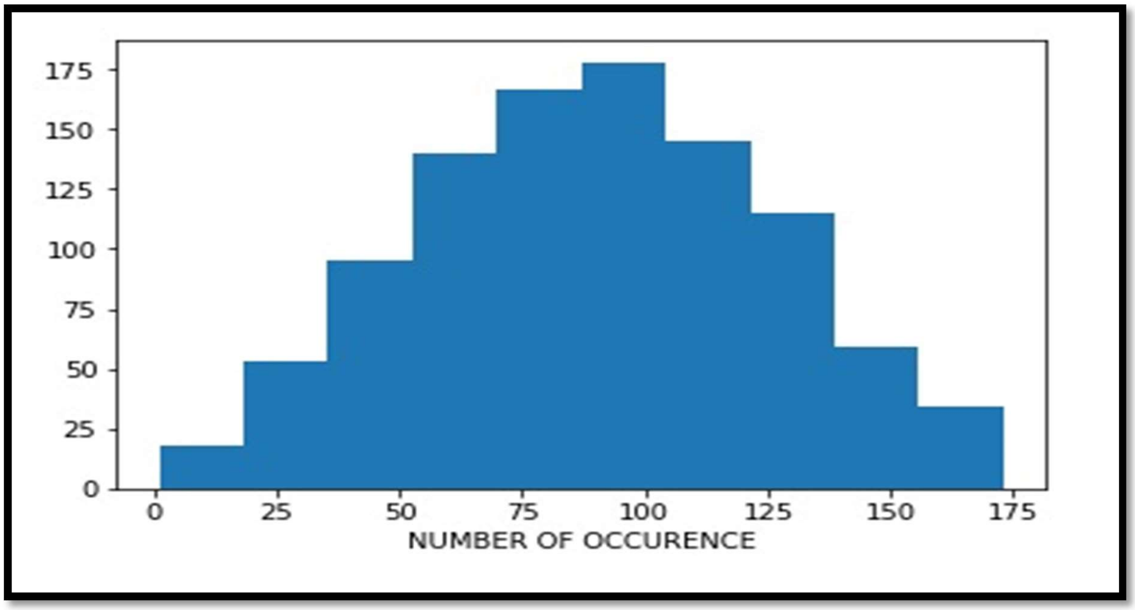
Output



PROGRAM :- WRITE A PYTHON PROGRAM TO PLOTTING IN SAME AND DIFFERENT GRAPH USING MATPLOTLIB.PYPLOT.

```
import random
import pylab
val = []

for n in range(1004):
    x = random.choice(range(0,90))
    y = random.choice(range(0,90))
    val.append(x+y)

pylab.hist(val, bins = 10)
pylab.xlabel('NUMBER OF OCCURENCE')
```

Output:

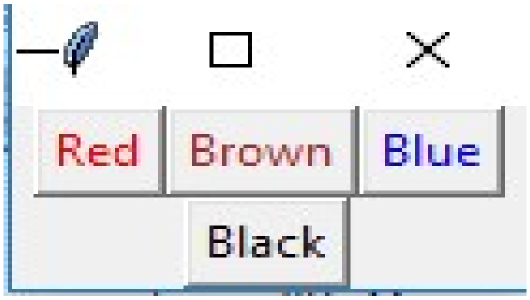


PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A SIMPLE BUTTON USING TKINTER PROGRAMMING.

```
from tkinter
import *root = Tk()
```

```
frame = Frame(root)

frame.pack()
bottomframe = Frame(root)
bottomframe.pack( side =
BOTTOM )
redbutton = Button(frame, text = 'Red', fg
='red')redbutton.pack( side = LEFT)
greenbutton = Button(frame, text = 'Brown', fg='brown')
greenbutton.pack( side = LEFT )
bluebutton = Button(frame, text ='Blue', fg ='blue')
bluebutton.pack( side = LEFT )
blackbutton = Button(bottomframe, text ='Black', fg ='black')
blackbutton.pack( side = BOTTOM)
root.mainloop()
```

Output :-

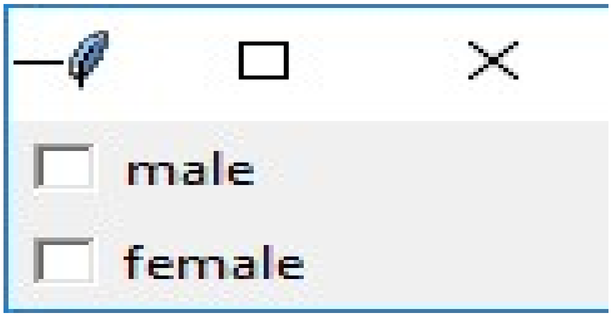


PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A CHECKBOX.

```
from tkinter import *
master = Tk()
va r1 = IntVar()
Checkbutton(master, text='male', variable=var1).grid(row=0,
 sticky=W)
var2 = IntVar() Checkbutton(master, text='female',
 variable=var2).grid(row=1, sticky=W)
```

```
mainloop()
```

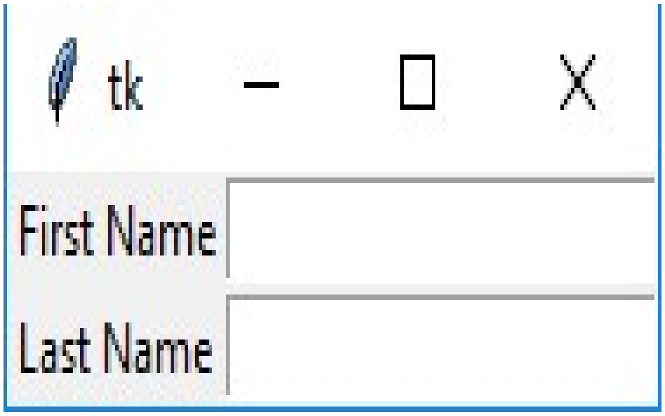
Output :-



PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A LABEL , ENTRYFIELD.

```
from tkinter import *
mas ter = Tk()
Label(master, text='First Name').grid(row=0)
Label(master, text='Last
Name').grid(row=1)e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
mainloop()
```

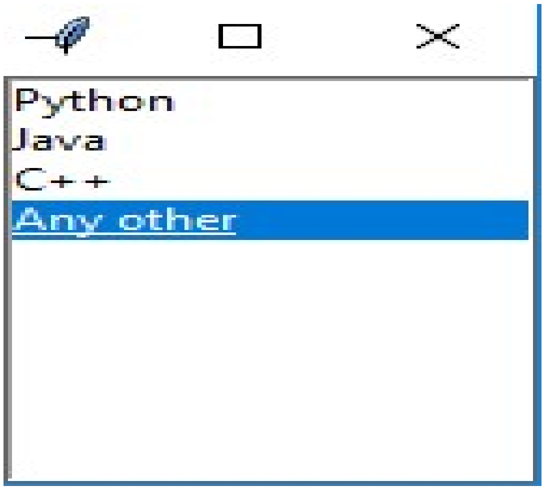
Output :-



PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A LISTBOX.

```
from tkinter
import *top = Tk()
Lb = Listbox(top)
Lb.insert(1, 'Python')

Lb.insert(2, 'Java')
Lb.insert(3, 'C++')
Lb.insert(4, 'Any other')
Lb.pack()
top.mainloop()
```

Output :-



```
Python
Java
C++
Any other
```

**PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A SIMPLE
SERVER.PY FILE.**

```
# This is server.py file
```

```
import socket             # Import socket module

s = socket.socket()        # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12345               # Reserve a port for your service.
s.bind((host, port))       # Bind to the port

s.listen(5)                # Now wait for client connection.



while True:
   c, addr = s.accept()     # Establish connection with client.
   print 'Got connection from', addr
   c.send('Thank you for connecting')
   c.close()                # Close the connection
```

PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A SIMPLE CLIENT.PY FILE.

```
# This is client.py file

import socket             # Import socket module

s = socket.socket()        # Create a socket object
host = socket.gethostname() # Get local machine name
port = 12345               # Reserve a port for your service.

s.connect((host, port))
print s.recv(1024)
s.close()                  # Close the socket when done
```

Output :-

```
# Following would start a server in background.
$ python server.py &

# Once server is started run client as follows:
$ python client.py
```

PROGRAM :- WRITE A PYTHON PROGRAM TO SEND AN E- MAIL USING SMTP.

```
import smtplib



sender = 'from@fromdomain.com'
receivers = ['to@todomain.com']

message = """From: From Person <from@fromdomain.com>
To: To Person <to@todomain.com>
Subject: SMTP e-mail test

This is a test e-mail message.
"""

try:
  smtpObj = smtplib.SMTP('localhost')
  smtpObj.sendmail(sender, receivers, message)
  print "Successfully sent email"
except SMTPException:
  print "Error: unable to send email"
```

PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A CONNECTION WITH DATABASE.

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google")
#printing the connection object
print(myconn)
```

Output:

```
<mysql.connector.connection.MySQLConnection object at 0x7fb142edd780>
```

PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A CONNECTION WITH DATABASE.

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",
passwd= "google", database = "mydb")
#printing the connection object
print(myconn)

#creating the cursor object
cur = myconn.cursor()
print(cr)
```

Output:

```
<mysql.connector.connection.MySQLConnection object at 0x7faa17a15748>
MySQLCursor: (Nothing executed yet)
```

PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A NEW DATABASE.

```
import mysql.connector

#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google")
#creating the cursor object
cur = myconn.cursor()
try:
    dbs = cur.execute("show databases")
except:
    myconn.rollback()

for x in cur:
    print(x)
myconn.close()
```

Output:

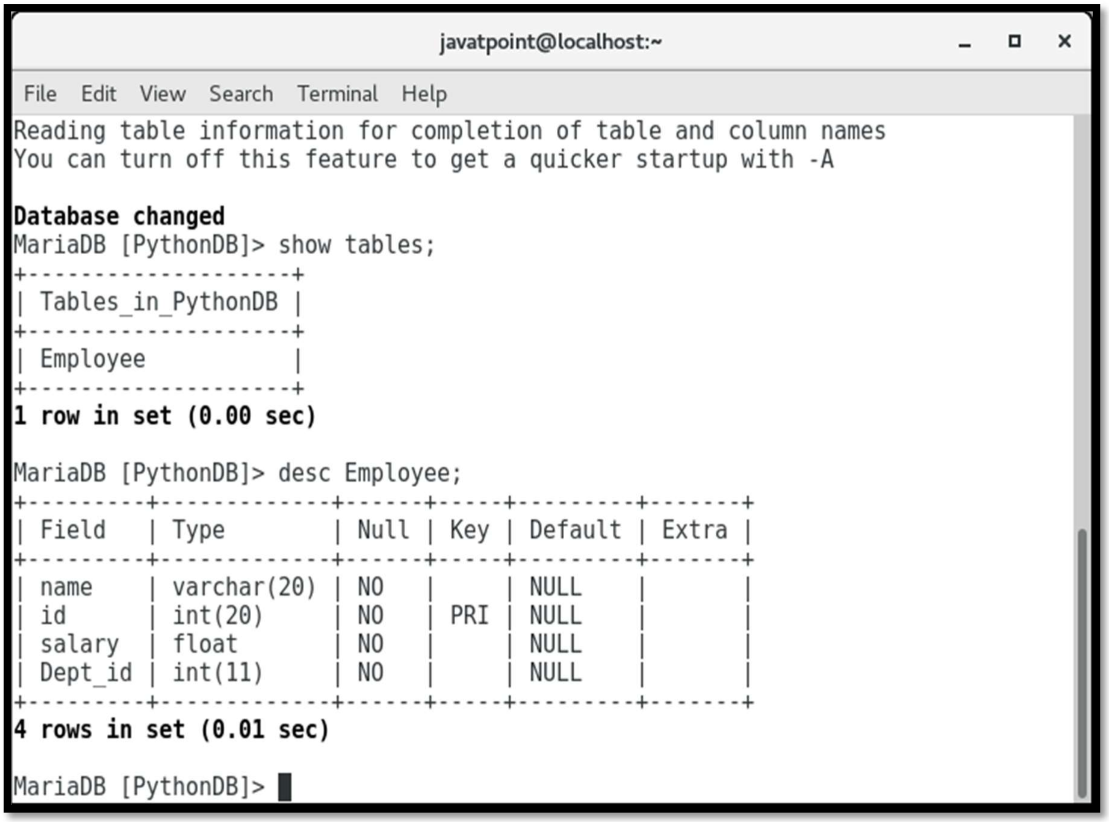
```
('EmployeeDB',)
('Test',)
('TestDB',)
('information_schema',)
('javatpoint',)
('javatpoint1',)
('mydb',)
('mysql',)
('performance_schema',
)
('testDB',)
```

PROGRAM :- WRITE A PYTHON PROGRAM TO CREATE A NEW TABLE EMPLOYEE.

```
create table Employee (name varchar(20) not null, id int primary

 key, salary float not null, Dept_Id int not null)
```

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
    #Creating a table with name Employee having four columns i.e., name, id, salary, and department id
    dbs = cur.execute("create table Employee(name varchar(20) not null, id int(20) not null primary key, salary float not null, Dept_id int not null)")
except:
    myconn.rollback()
    Myconn.close()
```

Output :

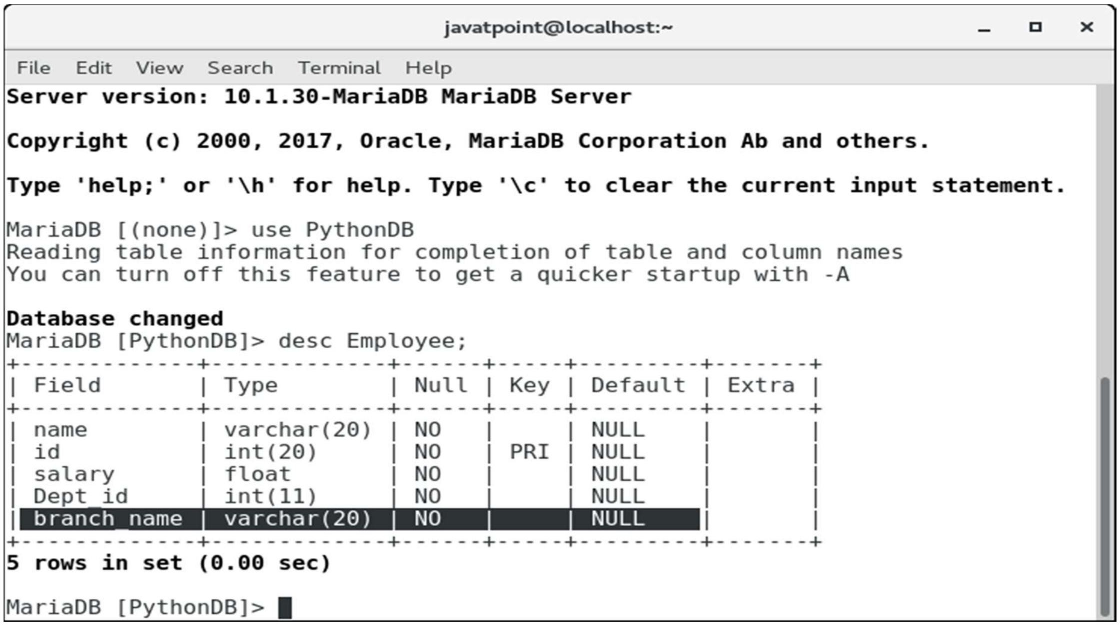


```
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> show tables;
+--------------------+
| Tables_in_PythonDB |
+--------------------+
| Employee           |
+--------------------+
1 row in set (0.00 sec)

MariaDB [PythonDB]> desc Employee;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| name    | varchar(20) | NO   |     | NULL    |       |
| id      | int(20)     | NO   | PRI | NULL    |       |
| salary  | float       | NO   |     | NULL    |       |
| Dept_id | int(11)     | NO   |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

MariaDB [PythonDB]>
```

PROGRAM :- WRITE A PYTHON PROGRAM TO ALTER NEW TABLE EMPLOYEE.

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")

#creating the cursor object

cur = myconn.cursor()
try:
    #adding a column branch name to the table Employee

    cur.execute("alter table Employee add branch_name varchar(20) not null")
except:
    myconn.rollback()
    Myconn.close()
```

Output :



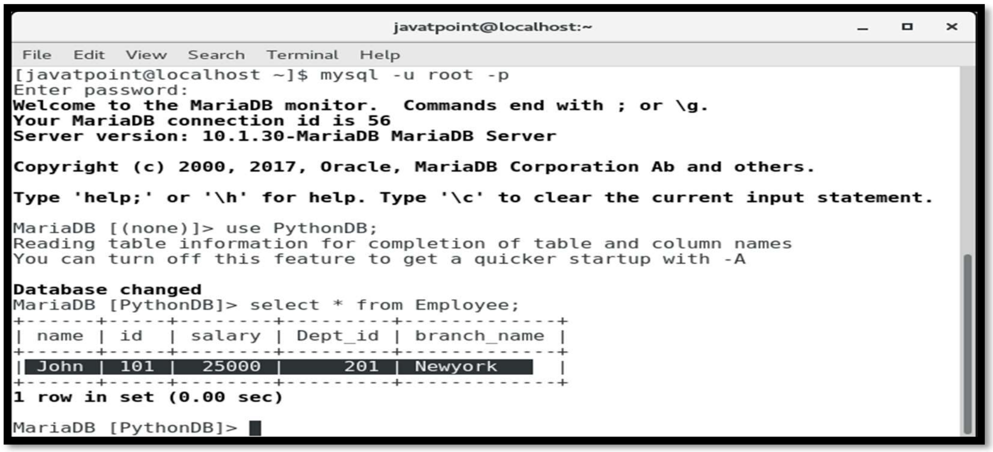
```
Server version: 10.1.30-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use PythonDB
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> desc Employee;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| name        | varchar(20) | NO   |     | NULL    |       |
| id          | int(20)     | NO   | PRI | NULL    |       |
| salary      | float       | NO   |     | NULL    |       |
| Dept_id     | int(11)     | NO   |     | NULL    |       |
| branch_name | varchar(20) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

MariaDB [PythonDB]>
```

PROGRAM :- WRITE A PYTHON PROGRAM TO INSERTING A SINGLE ROW INTO TABLE.

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")

#creating the cursor object
cur = myconn.cursor()
sql = "insert into Employee(name, id, salary, dept_id, branch_name) values ( %s, %s, %s, %s, %s)"
#The row values are provided in the form of tuple
val = ("John", 110, 25000.00, 201, "Newyork")
try:
    #inserting the values into the table
    cur.execute(sql,val)
    #commit the transaction
    myconn.commit()
except:
    myconn.rollback()
print(cur.rowcount,"record inserted!")
myconn.close()
```

Output :

1 record inserted!



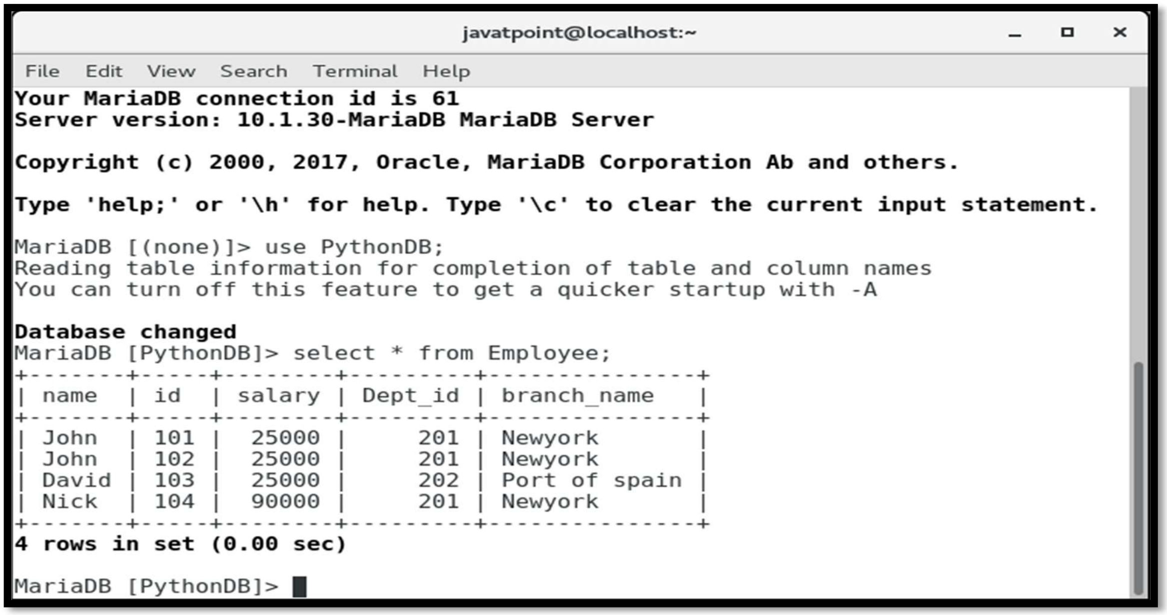
```
[javatpoint@localhost ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 56
Server version: 10.1.30-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use PythonDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> select * from Employee;
+------+-----+--------+---------+-------------+
| name | id  | salary | Dept_id | branch_name |
+------+-----+--------+---------+-------------+
| John | 101 |  25000 |     201 | Newyork     |
+------+-----+--------+---------+-------------+
1 row in set (0.00 sec)

MariaDB [PythonDB]> 
```

PROGRAM :- WRITE A PYTHON PROGRAM TO INSERTING A MULTIPLE ROWS INTO TABLE.

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
sql = "insert into Employee(name, id, salary, dept_id, branch_name) values ( %s, %s, %s, %s, %s)"


val = [("John", 102, 25000.00, 201, "Newyork"),("David",103,25000.00,202,"Port of spain"),("Nick",104,90000.00,201,"Newyork")]
try:
    #inserting the values into the table
    cur.executemany(sql,val)
    #commit the transaction
    myconn.commit()
    print(cur.rowcount,"records inserted!")
except:
    myconn.rollback()
    myconn.close()
```

Output :-

```
3 records inserted!
```



```
Your MariaDB connection id is 61
Server version: 10.1.30-MariaDB MariaDB Server

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use PythonDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> select * from Employee;
+-------+-----+--------+---------+---------------+
| name  | id  | salary | Dept_id | branch_name   |
+-------+-----+--------+---------+---------------+
| John  | 101 |  25000 |     201 | Newyork       |
| John  | 102 |  25000 |     201 | Newyork       |
| David | 103 |  25000 |     202 | Port of spain |
| Nick  | 104 |  90000 |     201 | Newyork       |
+-------+-----+--------+---------+---------------+
4 rows in set (0.00 sec)

MariaDB [PythonDB]> 
```

PROGRAM :- WRITE A PYTHON PROGRAM TO RETRIEVING ALL THE ROWS FROM TABLE.

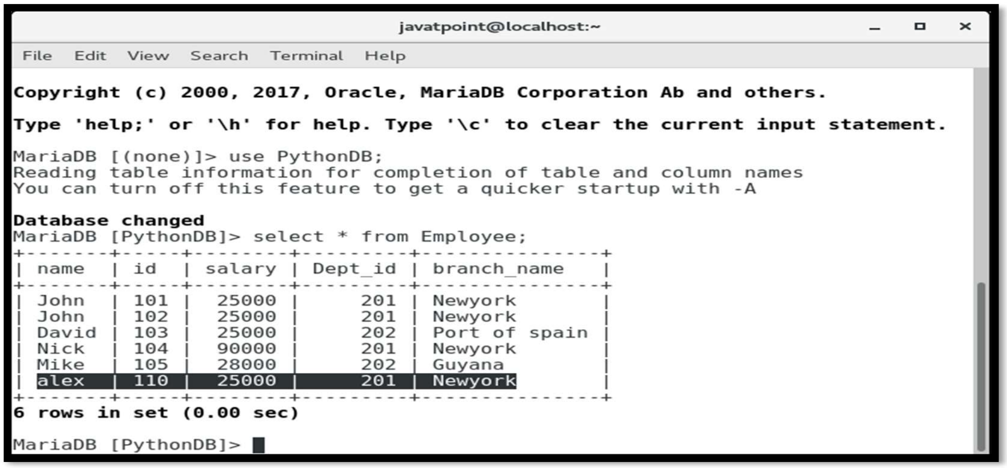
```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
    #Reading the Employee data
    cur.execute("select * from Employee")
    #fetching the rows from the cursor object

    //result = cur.fetchall()
    #printing the result
    for x in result:
    print(x);
    14.except:
    myconn.rollback()
    myconn.close()
```

Output :-

```
('John', 101, 25000.0, 201, 'Newyork')
('John', 102, 25000.0, 201, 'Newyork')
('David', 103, 25000.0, 202, 'Port of spain')
('Nick', 104, 90000.0, 201, 'Newyork')
('Mike', 105, 28000.0, 202, 'Guyana')
```

PROGRAM :- WRITE A PYTHON PROGRAM TO UPDATE A RECORD FROM THE TABLE.

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google",database = "PythonDB")
#creating the cursor object

cur = myconn.cursor()
try:
    #updating the name of the employee whose id is 110
    cur.execute("update Employee set name = 'alex' where id = 110")
    myconn.com
mit()10.except:
    myconn.rollback()
    myconn.close()
```

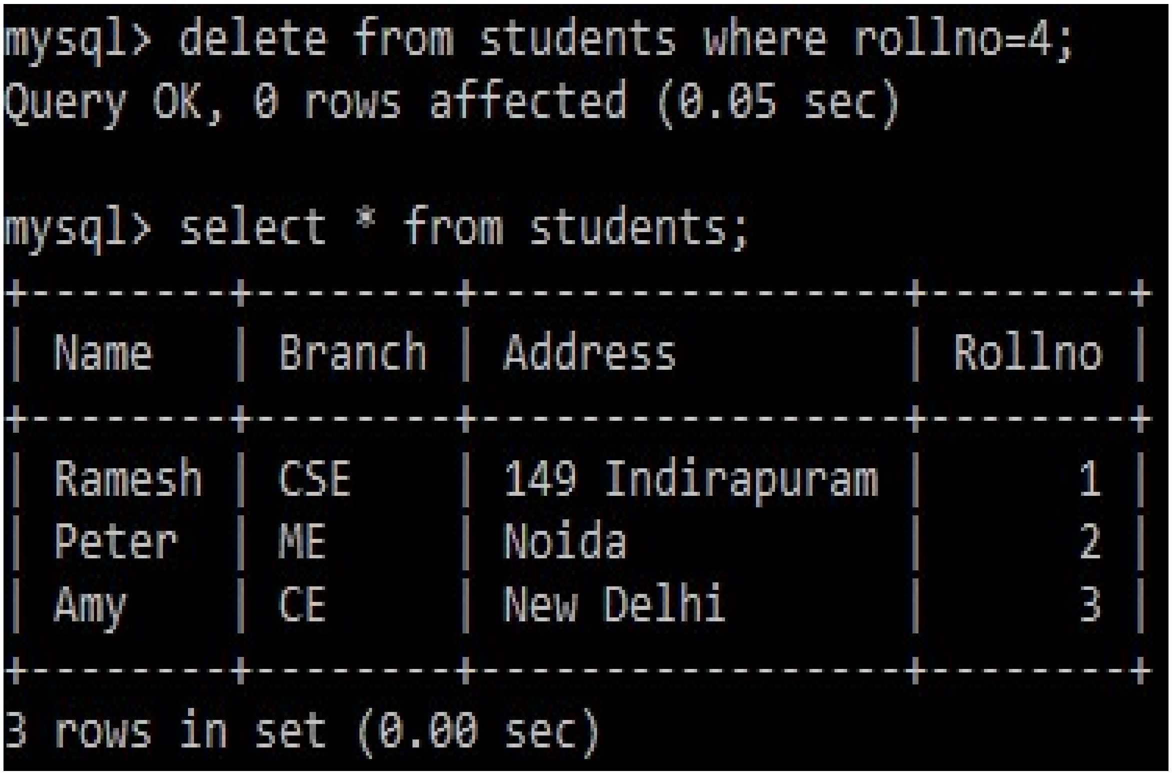


```
Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use PythonDB;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [PythonDB]> select * from Employee;
+-------+-----+--------+---------+---------------+
| name  | id  | salary | Dept_id | branch_name   |
+-------+-----+--------+---------+---------------+
| John  | 101 |  25000 |     201 | Newyork       |
| John  | 102 |  25000 |     201 | Newyork       |
| David | 103 |  25000 |     202 | Port of spain |
| Nick  | 104 |  90000 |     201 | Newyork       |
| Mike  | 105 |  28000 |     202 | Guyana        |
| alex  | 110 |  25000 |     201 | Newyork       |
+-------+-----+--------+---------+---------------+
6 rows in set (0.00 sec)

MariaDB [PythonDB]> 
```

PROGRAM :- WRITE A PYTHON PROGRAM TO DELETE A RECORD FROM THE TABLE.

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",
passwd= "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
    #Deleting the student details whose rollno is 4

    cur.execute("delete from students where rollno = 4")
    myconn.commit()
except:
myconn.rollback
()myconn.close()
```

Output :-



```
mysql> delete from students where rollno=4;
Query OK, 0 rows affected (0.05 sec)

mysql> select * from students;
+--------+--------+-----------------+--------+
| Name   | Branch | Address         | Rollno |
+--------+--------+-----------------+--------+
| Ramesh | CSE    | 149 Indirapuram |      1 |
| Peter  | ME     | Noida           |      2 |
| Amy    | CE     | New Delhi       |      3 |
+--------+--------+-----------------+--------+
3 rows in set (0.00 sec)
```