PHYTHON LABORATORY MANUAL

A.Rathina Kumari M.E

Regulation 2021

ISBN 978-935915284-4

PROBLEM SOLVING IN PYTHON LABORATORY MANUAL

(ARTIFICIAL INTELLIGENCE AND DATA SCIENCE ENGINEERING)

REGULATION – 2021

AUTHOR:

Mrs.A.Rathina Kumari.,MCA.,M.E.,

GENERAL GUIDELINES AND SAFETY INSTRUCTIONS

1. Sign in the log register as soon as you enter the lab and strictly observe your lab timings.
2. Strictly follow the written and verbal instructions given by the teacher / Lab Instructor. If you do not understand the instructions, the handouts and the procedures, ask the instructor or teacher.
3. **Never work alone!** You should be accompanied by your laboratory partner and / or the instructors / teaching assistants all the time.
4. It is mandatory to come to lab in a formal dress and wear your ID cards.
5. Do not wear loose-fitting clothing or jewels in the lab. Rings and necklaces are usual excellent conductors of electricity.
6. Mobile phones should be switched off in the lab. Keep bags in the bag rack.
7. Keep the labs clean at all times, no food and drinks allowed inside the lab.
8. Intentional misconduct will lead to expulsion from the lab.
9. Do not handle any equipment without reading the safety instructions. Read the handout and procedures in the Lab Manual before starting the experiments.
10. Do your wiring, setup, and a careful circuit checkout before applying power. Do not make circuit changes or perform any wiring when power is on.
11. Avoid contact with energized electrical circuits.
12. Do not insert connectors forcefully into the sockets.
13. **NEVER** try to experiment with the power from the wall plug.
14. Immediately report dangerous or exceptional conditions to the Lab instructor / teacher: Equipment that is not working as expected, wires or connectors are broken, the equipment that smells or “smokes”. If you are not sure what the problem is or what's going on, switch off the Emergency shutdown.
15. Never use damaged instruments, wires or connectors. Hand over these parts to the Lab instructor/Teacher.
16. Be sure of location of fire extinguishers and first aid kits in the laboratory.
17. After completion of Experiment, return the bread board, trainer kits, wires, CRO probes and other components to lab staff. Do not take any item from the lab without permission.
18. Observation book and lab record should be carried to each lab. Readings of current lab experiment are to be entered in Observation book and previous lab experiment should be written in Lab record book. Both the books should be corrected by the faculty in each lab.
19. Special Precautions during soldering practice
    a. Hold the soldering iron away from your body. Don't point the iron towards you. b. Don't use a spread solder on the board as it may cause short circuit.
    c. Do not overheat the components as excess heat may damage the components/board.
    d. In case of burn or injury seek first aid available in the lab or at the college dispensary.

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE ENGINEERING
REGULATION – 2021

GE3171 – PROBLEM SOLVING IN PYTHON LABORATORY

Mrs.A.RATHINA KUMARI.,MCA.,M.E.,
Assistant Professor/ Artificial Intelligence and Data Science Engineering
Annai Mira College of Engineering and Technology
Ranipet – 632 517

PREFACE

This book on **"PROBLEM SOLVING IN PYTHON LABORATORY MANUAL (CSE ,AI&DS)"** Covers complete syllabus prescribed by the Anna University, Chennai for the first semester B.E[CSE],B.TECH[AI&DS]. Degree course under **Outcome Based Education Credit System with the new regulation 2021**.

This book covers flow charts for simple python program, Logical concepts like sin(), cos() and circulate n-variables, display different types of pyramid patterns ,mathematical functions, Library functions and Python Gaming tools were developed using python programming Language.

We hope that this book will be useful to both teachers and students. Finally we would request the readers to kindly send their valuable comments and suggestions towards the improvement of the book and the same will be gratefully acknowledge.

Any suggestion from the reader for the betterment of this book can be dropped into kavirathna84@gmail.com

Mrs.A.RATHINA KUMARI., MCA.,M.E

ACKNOWLEDGEMENT

We are thankful to and fortunate enough to get constant encouragement, support and guideline from Chairman **Thiru.S.Ramadoss Ayya**, Secretary & Treasurer **Mr.G.Thamothiran** for his blessings to complete the book successfully.

We would not forget to remember our Principal **Dr.T.K.Gopinathan.,Ph.D.,** for his constant assistance in preparing this book.

ANNAI MIRA COLLEGE OF ENGINEERING AND TECHNOLOGY

NH-46, Chennai-Bengaluru National Highways, Arapakkam,

Vellore-632517, Tamil Nadu, India

Telephone: 04172-292925 Fax: 04172-292926

Email:amcet.rtet@gmail.com/info@amcet.in Web: www.amcet.in

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE ENGINEERING

GE3171- PROBLEM SOLVING AND PYTHON

PROGRAMMING LABORATORY

PREPARED BY

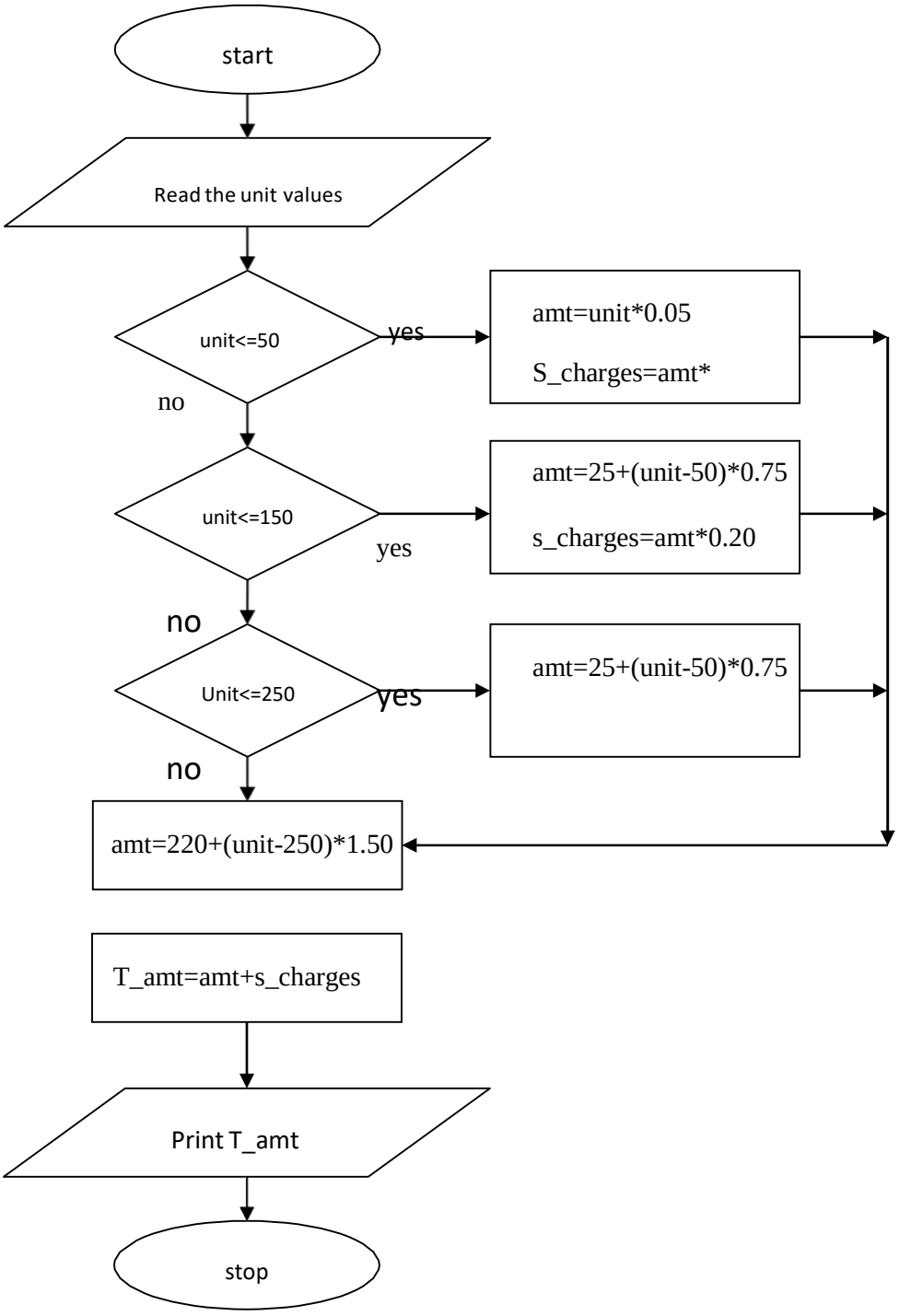
Mrs.A.RATHINA KUMARI.,MCA.,ME

APPROVED BY

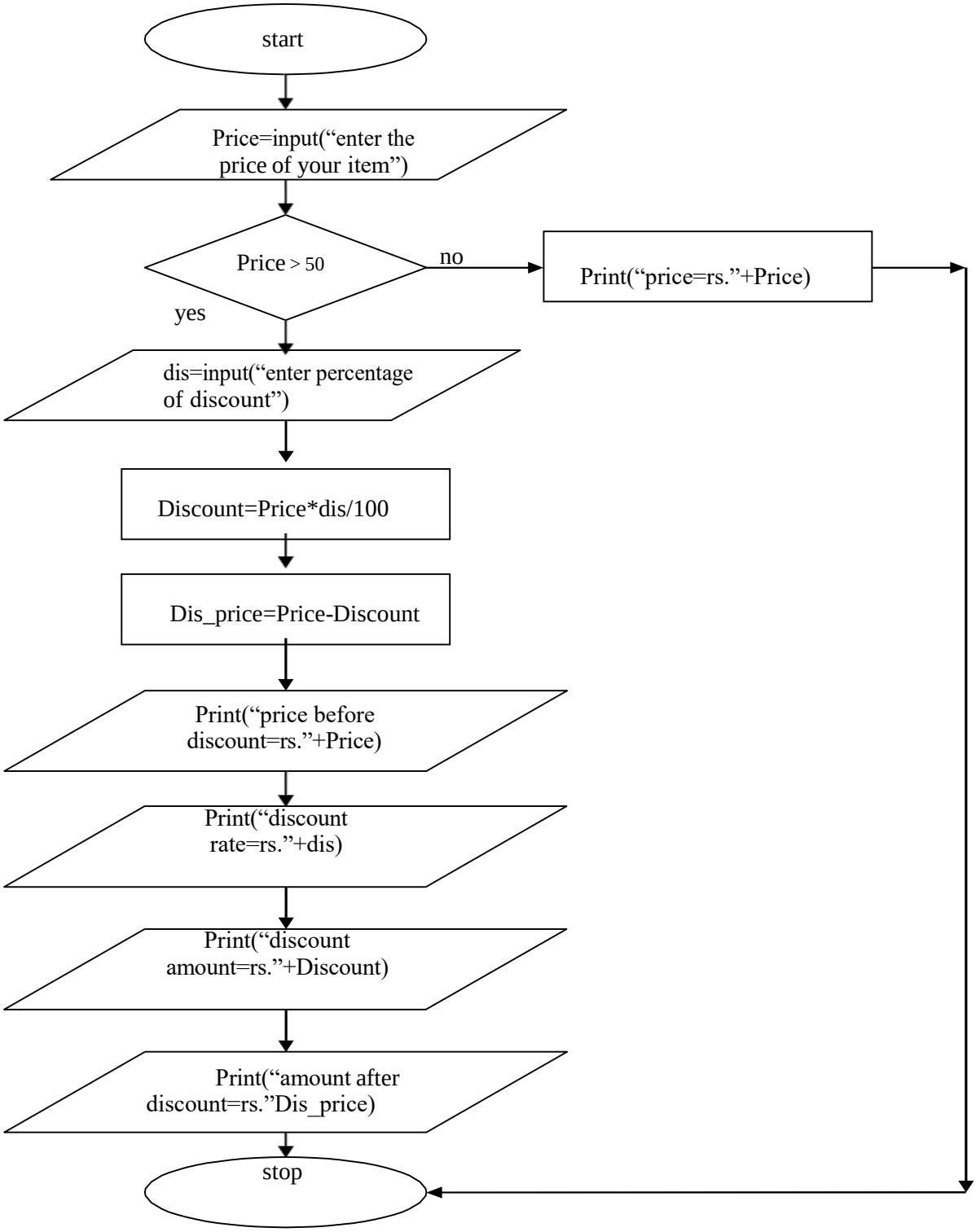
Mrs.M.DIVYA.,ME.,

TABLE OF CONTENT

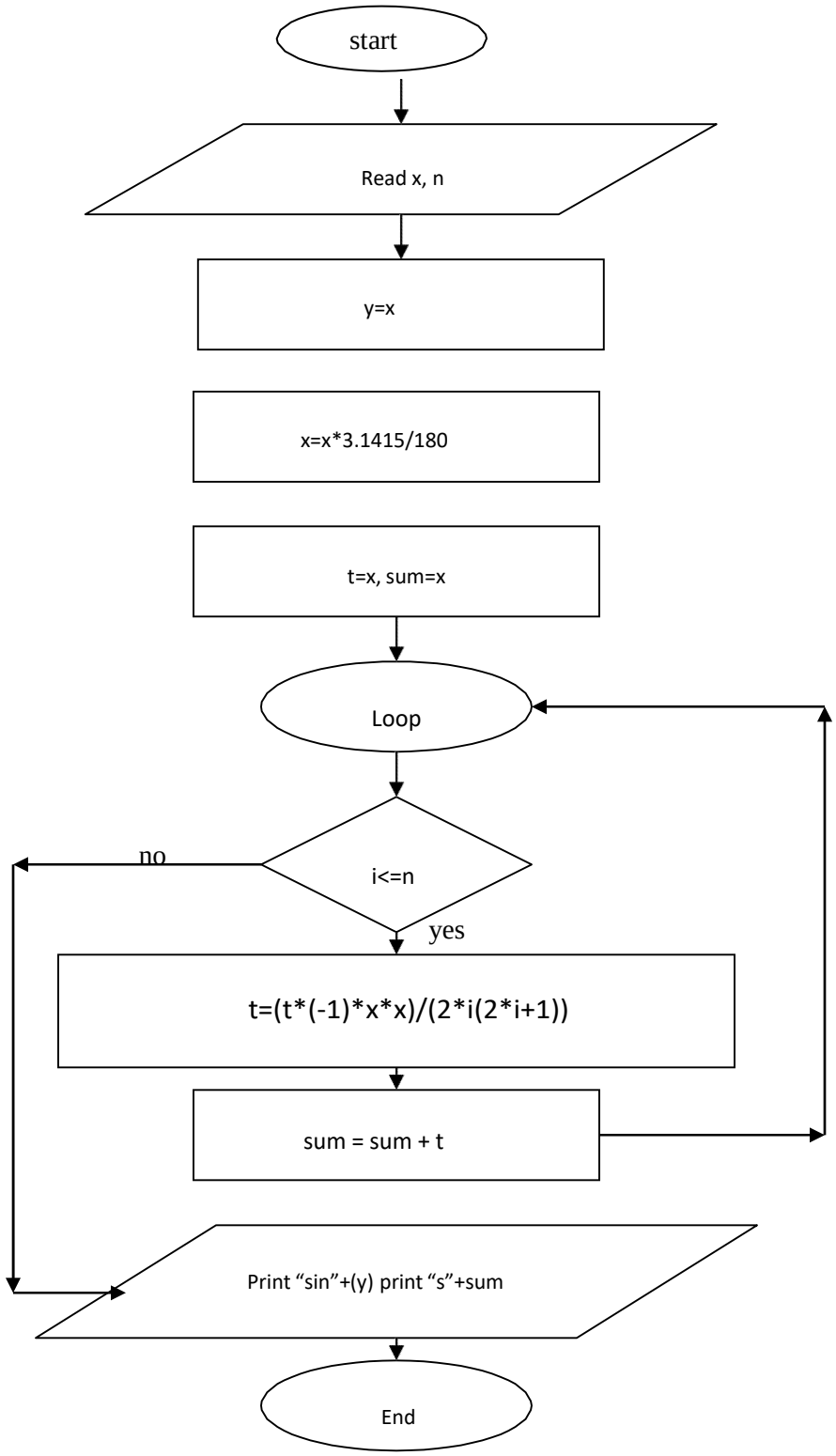
Ex. No:1a	DEVELOP A FLOWCHART FOR ELECTRICITY BILLING
Date:	



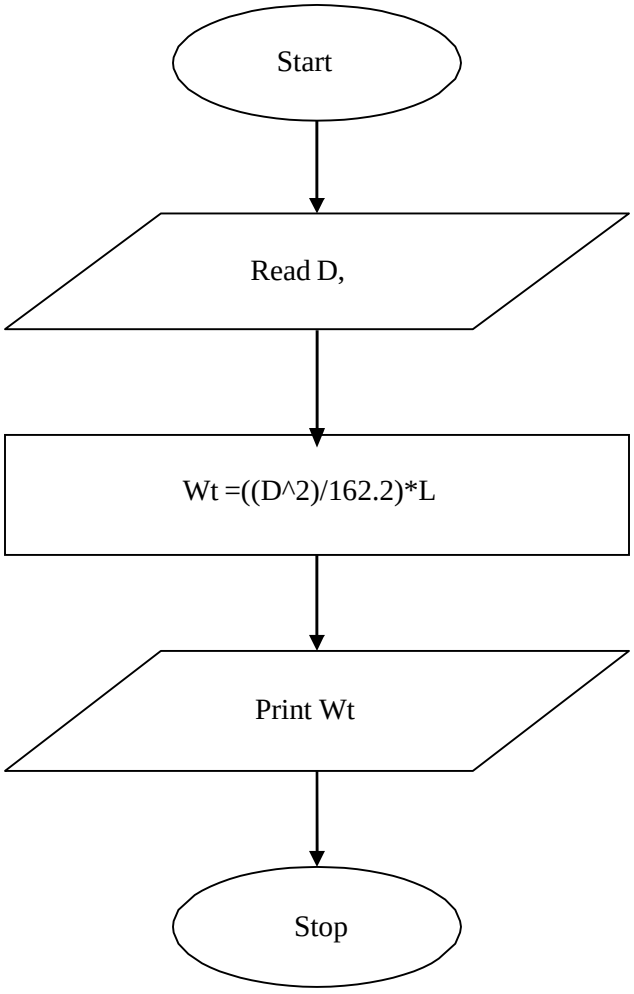
Ex. No:1b	DEVELOP A FLOWCHART FOR SHOP BILLING
Date:	



Ex. No:1c	DEVELOP A FLOWCHART FOR SINE SERIES
Date:	



Ex. No:1d	**DEVELOP A FLOWCHART FOR WEIGHT OF STEEL BAR**
Date:	



Ex. No:2a	EXCHANGE THE VALUES OF TWO VARIABLES
Date:	

Aim :

To write a Python program for exchange the values of two variables.

Algorithm :

Step 1 : Start the program.

Step 2 : Get the values from the user.

Step 3 : Swap both the values using temporary variables.

Step 4 : Print the swapped output.

Step 5 : Stop the program.

Program:

```
a=int(input("Enter the value for A:"))
b=int(input("Enter the value for B:"))
print("Before Exchanging the values")
print("A=",a,"B=",b)
temp=a
a=b
b=temp
print("After Exchanging the values of variables")
print("A=",a,"B=",b)
```

Output:

Enter the value for A: 10

Enter the value for B: 15

Before Exchanging thevalues

A= 10 B= 15

After Exchanging the values of variables

A= 15 B= 10

Result :

Thus the Python program for swapping two variables was executed successfullywithout any error.

Ex. No:2b	CIRCULATE THE VALUES OF N VARIABLES
Date:	

Aim :

To write a Python program for circulate the values of n variables.

Algorithm :

Step 1 : Start the program.

Step 2 : Get the number of values from user.

Step 3 : Declare the array variable.

Step 4 : Get the list values from user and appendit.

Step 5 : Circulate the list values by using for loop.

Step 6 : Print the circulated output.

Step 7 : Stop the program.

Program :

```
n=int(input("Enter number of values:"))
list1=[]
for val in range(0,n,1):
 ele=int(input("Enter integer:"))
 list1.append(ele)
 print("Circulating the elements of list",list1)
for val in range(0,n,1):
 ele=list1.pop(0)
 list1.append(ele)
 print(list1)
```

Output:

```
Enter number of values:4
Enter integer:10
Circulating the elements of list [10]
Enter integer:20
Circulating the elements of list [10, 20]
Enter integer:30
Circulating the elements of list [10, 20, 30]
Enter integer:40
Circulating the elements of list [10, 20, 30, 40]
[20, 30, 40, 10]
[30, 40, 10, 20]
[40, 10, 20, 30]
[10, 20, 30, 40]
```

Result :

Thus the Python program for circulate the values of n variables was executed successfully without any error.

Ex. No:2c	FIND THE DISTANCE BETWEEN TWO POINTS
Date:	

Aim :

To write a Python program for distance between two points using built infunctions.

Algorithm :

Step 1 : Start the program.

Step 2 : Import the math module.

Step 3 : Get the four values fromuser.

Step 4 : Calculate the distance between two points by using math.sqrt built in function.

Step 5 : Print the distance between two points output.

Step 6 : Stop the program.

Program:

```
import math
x1=int(input("Enter the value for X1:"))
y1=int(input("Enter the value for Y1:"))
x2=int(input("Enter the value for X2:"))
y2=int(input("Enter the value for Y2:"))
distance=math.sqrt(((x2-x1)**2)+((y2-y1)**2))
print("The Distance between two point is ",distance)
```

Output:

```
Enter the value for X1:2
Enter the value for Y1:3
Enter the value for X2:5
Enter the value for Y2:6
The Distance between two point is 4.242640687119285
```

Result :

Thus the python program for distance between two points was executed successfully without any error.

Ex. No:3a	NUMBER SERIES
Date:	

Aim :

To write a Python program for number series using conditionals and iterativeloops.

Algorithm :

Step 1 : Start the program.
Step 2 : Initialize the array variables.
Step 3 : Get the values from the user.
Step 4 : Check the range value by using for loop.
Step 5 : Process the N series of numbers until the loop ends.
Step 6 : Append all the numbers one by one.
Step 7 : Print the result.
Step 8 : Stop the program.

Program :

```
n=int(input("Enter a number:"))
n = n+1
a=[]
for i in range(1,n+1):
   print(i,sep="",end="")
   if(i<n):
   print("+",sep="",end="")
   a.append(i)
   print("=",sum(a))
     print()
```

Output:

```
Enter a number:6
   1+= 1
   2+= 3
   3+= 6
   4+= 10
   5+= 15
   6+= 21
   7
```

Result :

Thus the python program for number series using conditionals and iterative loopswere executed successfully.

Ex. No:3b	NUMBER PATTERNS
Date:	

Aim :

To write a python program for number patterns using iterative loops.

Algorithm :

Step 1 : Start the program.

Step 2 : Initialize the number of row values.

Step 3 : Check the range value by using for loop.

Step 4 : Process the number patterns using iterative loops.

Step 5 : Print the result.

Step 6 : Stop the program.

Program :

```
rows=int(input("Enter the row value:"))
for i in range(1,rows+1):
    for j in range(1,i+1):
        print(j,end="")
    print()
```

output:

```
Enter the row value:5
1
12
123
1234
 12345
```

Result:

Thus the python program for number patterns using iterative loops were executed successfully.

Ex. No:3c	PYRAMID PATTERN
Date:	

Aim :

To write a Python program for pyramid pattern using iterative loops.

Algorithm :

Step 1 : Start the program.

Step 2 : Initialize the variables.

Step 3 : Get the values from the user.

Step 4 : Check the range value by using for loop.

Step 5 : Calculate the space for pyramid pattern.

Step 6 : Each iteration increment the pattern like * until the loops end

Step 7 : Print the pyramid pattern.

Step 8 : Stop the program.

Program :

```
rows=int(input("Enter the row value:"))
for i in range(1,rows+1):
    for j in range(1,i+1):
        print("*",end="")
    print()
```

Output:

```
Enter the row value:5

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

Result :

Thus the python program for pyramid pattern using iterative loops were executed successfully.

Ex. No:4a	LIBRARY DETAILS
Date:	

Aim :

To write a Python program for library details using the concept of tupleoperations.

Algorithm :

Step 1 : Start the program.

Step 2 : Initialize the tuple values.

Step 3 : Do the tuple operations such as index, length, concatenation, repetition,membership operations.

Step 4 : Process the slicing in the library tuple values.

Step 5 : Print all outputs.

Step 6 : Stop the program.

Program :

```
library1=("deptbooks","deptjournals","uniquestpapers")
library2=("competitveexamguides","ebooks","ejournals")
print("Indexing position of library1:",library1[1])
print("Numbers of components in library 1",len(library1))
print("Numbers of components in library 2",len(library2))
print("Concatenation of two library",library1+library2)
print("Repetition of library1",library1*2)
print("Membership operator of library1:","ebooks"in library1)
print("Membership operator of library2:","ebooks"in library2)
print("Slicing of library2:",library2[0:2])
```

Output:

Indexing position of library1:
deptjournalsNumbers of components in library 1 3 Numbers of components in library 2 3
Concatenation of two library ('deptbooks', 'deptjournals', 'uniquestpapers', 'competitveexamguides', 'ebooks', 'ejournals')
Repetition of library1 ('deptbooks', 'deptjournals', 'uniquestpapers', 'deptbooks','deptjournals', 'uniquestpapers')
Membership operator of library1: FalseMembership operator of library2: True
Slicing of library2: ('competitveexamguides', 'ebooks')

Result :

Thus the Python program of tuple operation for library details was executedsuccessfully without any error

Ex. No:4b	MATERIALS REQUIRED FOR CONSTRUCTION OF A BUILDING
Date:	

Aim :

To write a Python tuple operations for materials required for construction of abuilding.

Algorithm :

Step 1 : Start the program.
Step 2 : Initialize the tuple values.
Step 3 : Do the tuple operations such as length, concatenation, repetition, membershipoperations.
Step 4 : Process the slicing in the tuple values.
Step 5 : Print all outputs.
Step 6 : Stop the program.

Program :

```
building1=("bricks","sand","cement")
building2=("tiles","paint","wood")
print("Indexing position of building1:",building1[1])
print("Numbers of components in building1",len(building1))
print("Numbers of components in building2",len(building2))
print("Concatenation of two building",building1+building2)
print("Repetition of building1",building1*2)
print("Membership operator of building1:","components" in building1)
print("Membership operator of building2:","components" in building2)
print("Slicing of building1:",building1[0:1])
```

Output:

Indexing position of building1: sandNumbers of components in building1 3 Numbers of components in building2 3
Concatenation of two building ('bricks', 'sand', 'cement', 'tiles', 'paint', 'wood')Repetition of building1 ('bricks', 'sand', 'cement', 'bricks', 'sand', 'cement') Membership operator of building1: False
Membership operator of building2: FalseSlicing of building1: ('bricks',)

Result :

Thus the Python tuple operations for materials required for construction of abuilding were executed successfully.

Ex. No:4c	COMPONENTS OF A CAR
Date:	

Aim :

To write a Python program to create a component of a car using list operationsfor.

Algorithm:

Step1 : Start the program.
Step 2 : Initialize the list values.
Step 3 : Process all the list operations such as length, concatenation, repetition, membership operations.
Step 4 : Slice some of the car components in the list values.
Step 5 : Print all outputs.
Step 6 : Stop the program.

Program :

```
car1=["steering","wheels","brake","engine","seats"]
car2=["accelerator","clutch","gear","horn","indicator","battery"]
print("Indexing position of car2:",car2[1])
print("Numbers of components in car1",len(car1))
print("Numbers of components in car 2",len(car2))
print("Concatenation of two car",car1+car2)
print("Repetition of car1",car1*2)
print("Membership operator of car1:","components"in car1)
print("Membership operator of car2:","components"in car2)
print("Slicing of car2:",car2[0:2])
```

Output:

```
Indexing position of car2: clutch
Numbers of components in car1 5
Numbers of components in car 2 6
Concatenation of two car ['steering', 'wheels', 'brake', 'engine', 'seats', 'accelerator',
'clutch', 'gear', 'horn', 'indicator', 'battery']
Repetition of car1 ['steering', 'wheels', 'brake', 'engine', 'seats', 'steering', 'wheels',
'brake', 'engine', 'seats']
Membership operator of car1: False
Membership operator of car2: False
Slicing of car2: ['accelerator', 'clutch']
```

Result :

Thus the Python list operation for components of a car were executed successfully.

Ex. No:5a	IMPLEMENT THE PROGRAMMING LANGUAGE DETAILS USING SET OPERATIONS
Date:	

Aim :

To write a Python program for implementing the programming language detailsusing set operations.

Algorithm :

Step 1 : Start the program.

Step 2 : Initialize the set values from user.

Step 3 : Implement the set operations such as union, intersection and set difference.

Step 4 : Print all outputs.

Step 5 : Stop the program.

Program :

```
lang1={"c","c++","python","java"}
lang2={"c","python","c#","net"}
print("The set1 language are",lang1)
print("The set2 language are",lang2)
print("Union of two sets are",lang1.union(lang2))
print("Intersection of two sets are",lang1.intersection(lang2))
print("Difference of two sets are",lang1.difference(lang2))
```

Output:

```
The set1 language are {'c', 'java', 'c++', 'python'}
The set2 language are {'c', 'net', 'c#', 'python'}
Union of two sets are {'net', 'java', 'c++', 'python', 'c', 'c#'}
Intersection of two sets are {'c', 'python'}
Difference of two sets are {'java', 'c++'}
```

Result :

Thus the python set operation for programming language details was executed successfully.

Ex. No:5b	IMPLEMENT THE COMPONENTS OF AN AUTOMOBILE USING DICTIONARIES
Date:	

Aim :

To write a Python program for implementing the components of an automobile using dictionaries.

Algorithm :

Step 1 : Start the program.

Step 2 : Initialize the dictionaries values from user.

Step 3 : Implement the dictionaries operations such as add items, accessing the items and check the availability status using membership operator.

Step 4 : Print all outputs.

Step 5 : Stop the program.

Program :

```
auto1={"transmission":"clutch","body":"steering system","auxiliary":"seats"}
print("Items in the dictionaries:",auto1)
auto1["body2"]="wheels"
print("Add element in the dictionaries:",auto1)
print("Accessing single elements in the dictionaries:",auto1["body"])
print("Item in the dictionaries or not:","body" in auto1)
print("Item in the dictionaries or not:","head" in auto1)
```

Output:

```
Items in the dictionaries: {'transmission': 'clutch', 'body': 'steering system', 'auxiliary':
'seats'}
add element in the dictionaries: {'transmission': 'clutch', 'body': 'steering system',
'auxiliary': 'seats', 'body2': 'wheels'}
Accessing single elements in the dictionaries: steering system
Item in the dictionaries or not: True
Item in the dictionaries or not: False
```

Result :

Thus the Python dictionaries operation for components of an automobile wasexecuted successfully

Ex.No: 6a	FACTORIAL USING FUNCTIONS
Date:	

Aim :

To write a Python program to find factorial using user defined functions.

Algorithm :

Step 1 : Start the program.
Step 2 : Initialize the value for num variable.
Step 3 : Define the user defined function factorial.ite
Step 4 : Check the value using conditional statement.
Step 5 : Calculate the factorial by using while loop.
Step 6 : Print the factorial result.
Step 7 : Stop the program.

Program :

```
def fact(n):
   if n<0:
      return 0
   elif n==0 or n==1:
      return 1
   else:
      fact=1
   while(n>1):
     fact*=n
      n-=1
   return fact
num=int(input("Enter the number:"));
print("Factorial of :",num,"is",fact(num))
```

Output :

```
Enter the number:4
Factorial of: 4 is 24
```

Result :

Thus the python program for factorial using user defined functions was executed successfully.

Ex.N0: 6b	CALCULATE THE LARGEST
Date:	NUMBER

Aim :
To write a Python program to calculate the largest number in the list using functions.

Algorithm :
Step 1 : Start the program.
Step 2 : Initialize the list values.
Step 3 : Call the appropriate user defined function (function name: large()).
Step 4 : Declare the first element as max withinthe def function.
Step 5 : Compare each element by using for loop.
Step 6 : Find the largest element by using ifconditions.
Step 7 : Then return the largest element present in the initialized list.
Step 8 : Stop the program.

Program:

```
def large(arr):
    max = arr[0]
    for elem in arr:
        if(elem > max):
            max = elem
    return max
list_1 = [56528,1294,314,313,122]
result = large(list_1)
print(result)
```

Output :

```
56528
```

Result :

Thus the python program for largest number in the list using functions was executed successfully.

Ex.No: 6c	CALCULATE THE AREA OF DIFFERENT SHAPES USING FUNCTION
Date :	

Aim :

To write a Python program to calculate the area of different shapes using functions.

Algorithm :

Step 1 : Start the program.
Step 2 : Get the shape from user for calculating the area.
Step 3 : Call the appropriate user defined function.
Step 4 : Calculate the area of different shapes like rectangle, square, triangle, circle and parallelogram.
Step 5 : Print the area value.
Step 6 : Stop the program.

Program:

```
def calculate_area(name):
    name =name.lower()
    if(name=="rectangle"):
        l=int(input("Enter rectangle's length: "))
        b=int(input("Enter rectangle's breadth: "))
        r_area=l*b
        print(f"The area of rectangle is {r_area}")
    elif(name=="square"):
        s=int(input("Enter square's side length:"))
        sqt_area=s*s
        print(f"The area of square is {sqt_area}")
    elif(name=="triangle"):
        h=int(input("Enter square's height length: "))
        b=int(input("Enter square's breadth length: "))
        tri_area=0.5*b*h
        print(f"The area of triangle is {tri_area}")
    elif(name=="circle"):
        r=int(input("Enter circle's radius length: "))
        pi=3.14
        circle_area=pi*r*r
        print(f"The area of triangle is {circle_area}")
        b=int(input("Enter parallelogram's base length: "))
```

```
    else:
        print("sorry! This is shape is not available")
    if(name=="main"):
        print("Calculate Shape Area")
shape_name=input("Enter the name of shape whose area you want to find: ")
calculate_area(shape_name)
```

Output :

```
Enter the name of shape whose area you want to find:square
Enter square's side length:4
The area of rectangle is16
```

Result :

Thus the python program for area of different shapes using functions was executed successfully.

Ex.No: 7	IMPLEMENTING A PYTHON PROGRAM USING STRINGS
Date:	

Aim :

To write a python program for implementing the string methods for reversing the string, counting the character in a string, replacing a character in the string ad check whether the given string is palindrome or not.

Algorithm :

Step 1 : Start the program.
Step 2 : Get the string from the user in the variable(str).
Step 3 : Using the string method [::-1], for displaying the reverse of the string.
Step 4 : Using the function isPalindrome, check whether the string is palindrome or not.
Step 5 : Using the string method len(str), for finding the length of the given string.
Step 6 : Stop the program.

Program:

```
s=input('Enter a string:')
r=s[::-1]
if r==s:
   print('The given string is Palindrome')
else:
   print('The given string is NOT Palindrome')
  print('The length of a string:',len(s))
  print('The reverse string:',r)
```

Output:

```
Enter a string: Mailam
The given string is NOT palindrome
The length of a string:6
The reverse string:maliaM
```

Result :

Thus the python program for implementing the string methods for reversing the string, counting the character in a string and check whether the given string is palindrome or not was implemented successfully.

Ex. No: 8	MODULES AND PYTHON STANDARD LIBRARIES IMPLEMENTING PROGRAMS USING WRITTEN
Date:	

Pandas

The word pandas is an acronym which is derived from "python and data analysis"and "panel data ".

Pandas is a software library written for the python programming language. It is used for data manipulation and analysis.

It provides special data structures and operations for the manipulation ofnumerical tables and time series.Pandas is a free software released under the three-clause BSD license. Two important structures of pandas :

Series : A series is a one dimensional labeled array like object. It is capable of holding any data type, e.g. integers, floats, strings, python objects and so on. It can be seen as a data structure withtwo arrays: one functioning as the index, i.e.the labels, and the other one contains the actual data.

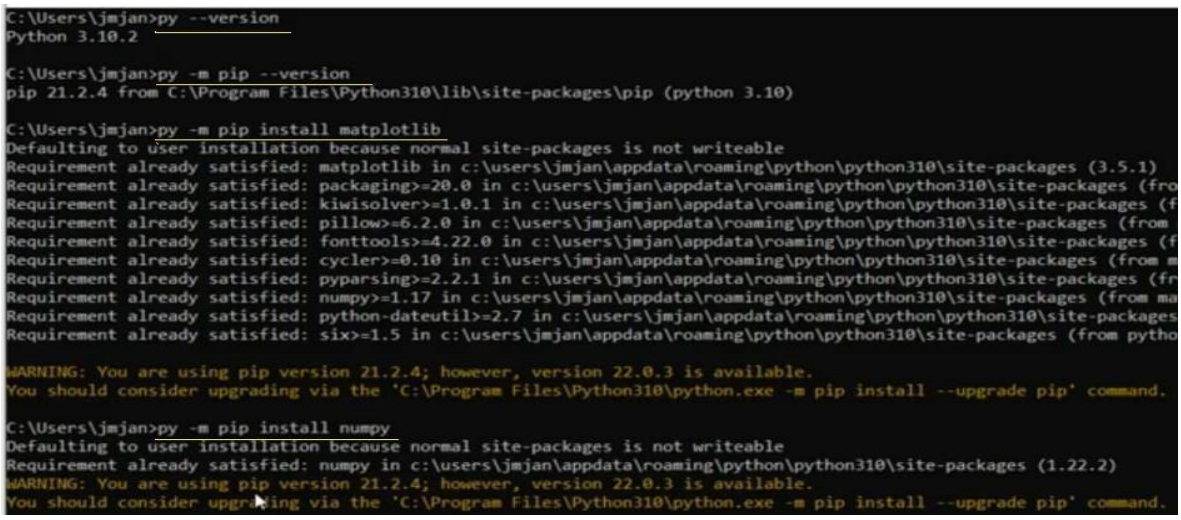
DataFrame : Like a spreadsheet or excel sheet, a DataFrame object contains anordered collectionof columns. Each column consists of a unique data type, but different columns can have different types, e.g. the first column may consist of integers, while the second one consists of Boolean values and so on.

To import pands and number in you python script, add the below piece of code :Import pandas aspd Import numpy as npNumpy

Numpy means " Numeric Python " or " Numerical Python ".

Numpy is the fundamental package for scientific computing in python.

It is a python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment ofroutines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, i/o, discrete fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.



Program :

```
import numpy as np
arr=np.array([[-1,2,0,4],[4,-0.5,6,0],[2.6,0,7,8],[3,-7,4,2.0]])
print("Intial array:")
print(arr) sliced_arr=ar
r[:2,::2]
print("Array with first two rows""alternate columns(0 and 2):\n",sliced_arr)
index_arr=arr[[1,1,0,3],[3,2,1,0]]
print("\n Elements at  indices(1,3),(1,2),(0,1),(3,0):\n",index_arr)
```

Output:

```
Intial array:
[[-1. 2. 0. 4. ]
[ 4. -0.5 6. 0. ]
[ 2.6 0. 7. 8. ]
[ 3. -7. 4. 2. ]]
  Array with first two rows alternate columns(0 and2):
[[-1. 0.]
[ 4. 6.]]
Elements at indices(1,3),(1,2),(0,1),(3,0):[0. 6. 2. 3.]
```

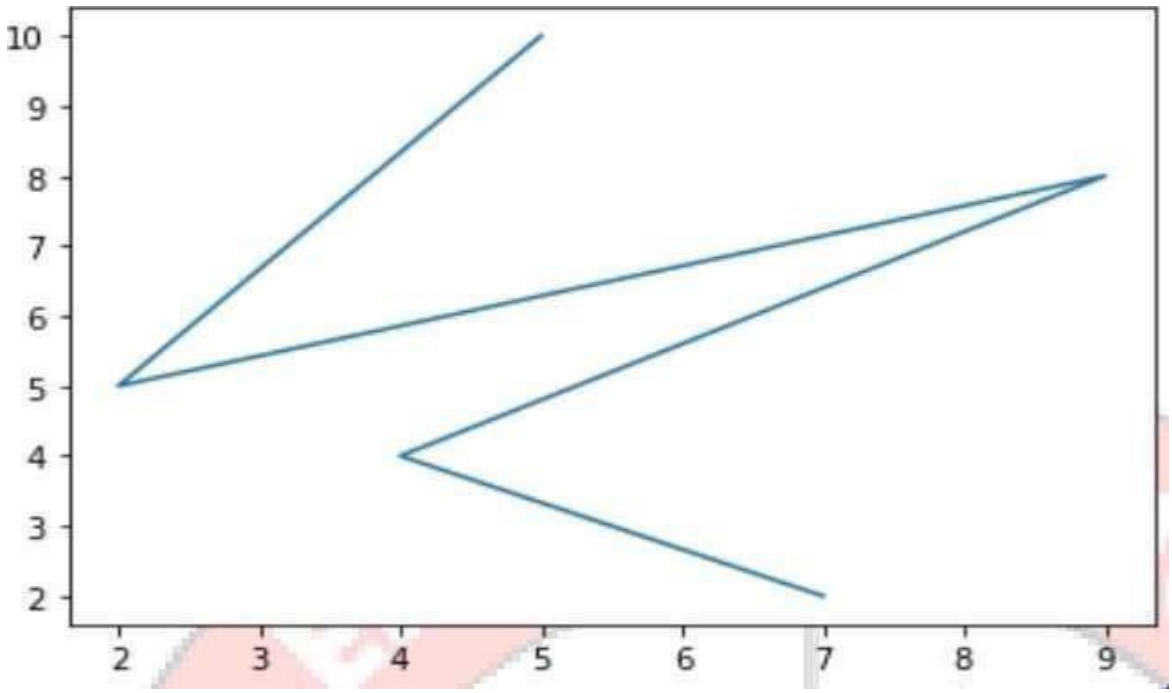
Matplotlib

Matplotlib is a python 2d plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
It is an amazing visualization library in python for 2d plots of arrays.
Matplotlib is a multi-platform data visualization library built on numpy arrays and designed to work with the broader scipy stack.
One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib cosists of several plots like line, bar, scatter, histogram, etc.
Importing matplotlib :
From matplotlib import pyplot as plt
orImport matplotlib.pyplot as plt
Basic plots in matplotlib :
Matplotlib comes with a wide variety of plots.
Plots help to understand trends, patterns, and to make correlations.
They're typically instruments for reasoning about quantitative information. Some of the sample plots are covered here.

Program 1:

```
from matplotlib import pyplot as
pltx=[5,2,9,4,7]
y=[10,5,8,4,2]
plt.plot(x,y)
plot.show()
```

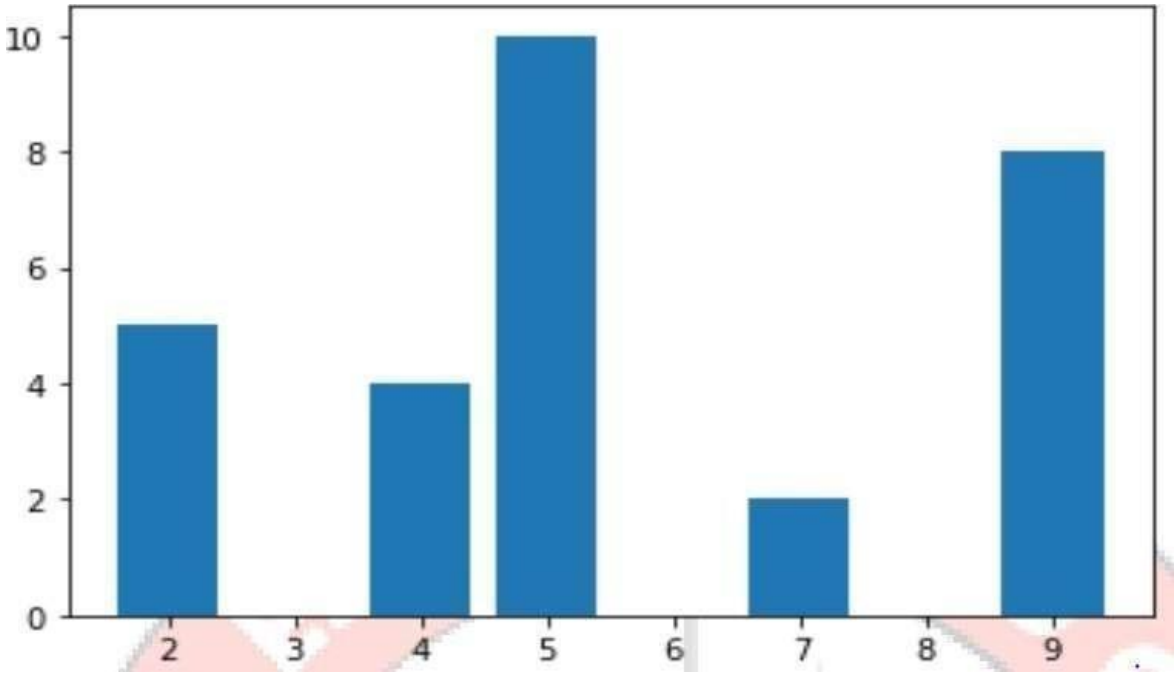
Output:



Program 2 :

```
from matplotlib import pyplot as
pltx=[5,2,9,4,7]
y=[10,5,8,4,2]
plt.bar(x,y)
plt.show()
```

Output:



Scipy

Scipy stands for scientific python.
Scipy is a free and open source python library used for scientific computing and technical computing.
It is a collection of mathematical algorithms and convenience functions built on the numpy extensions of python.
It adds significant power to the interactive python session by providing the user with high-level commands and classesfor manipulating and visualizing data.
As mentioned earlier, scipy builds on numpy and therefore if you import scipy, there is no need to import numpy.
Install scipy using pip
Install the scipy library by using the pip command.
Pip is basically a recursive acronym which stands for 'pip installs packages'.
It is a standard package manager which can be installed in most of the operating systems.
To install, run the following command in the terminal: pip install scipy
Install scipy using anaconda
Conda install –c anaconda scipy
Sub packages in scipy
Many dedicated software tools are necessary for python scientific computing, and scipy is one such tool or library offering many pythn modules that we can work with in order to perform complex operations.

The following table shows some of the modules or sub-packages that can be used for computing:

SI.NO.	SUB-PACKAGE	DESCRIPTION
1.	scipy.cluster	Cluster algorithms are used to vectorquantization/kmeans
2.	scipy.constants	It represents physical and mathematical constants.
3.	scipy.fftpack	It is used for fourier transform.
4.	scipy.integrate	Integration routines.
5.	scipy.interpolation	Interpolation.
6.	scipy.linalg	It is used for linear algebra routine.
7.	scipy.io	It is used for data input and output.
8.	scipy.ndimage	It is used for the n-dimension image.
9.	scipy.odr	Orthogonal distance regression.
10.	scipy.optimize	It is used for optimization.
11.	scipy.signal	It is used in signal processing.
12.	scipy.sparse	Sparse matrices and associated routines.
13.	scipy.spatial	Spatial data structures and algorithms.
14.	scipy.special	Special function.
15.	scipy.stats	Statistics.
16.	scipy.weaves	It is a tool for writing

Result :

Thus, we study about pandas, numpy, matplotlib, scipy applications, installationprocedure and also implement the sample program successfully.

Ex.No: 9	IMPLEMENT COPY FROM ONE FILE TO ANOTHER, WORD COUNT AND FIND
Date:	THE LARGEST WORD USING FILE HANDLING

Aim :

To write a python program for implementing the file handling operation for performing word count in the file, longest word in the file and copy the content of one file into another.

Algorithm :

Step 1 : Start the program.
Step 2 : Create a text file as text1.txt and type the content in it.
Step 3 : Open the filef1 (“text1.txt”) in the read mode.
Step 4 : Create the filef2 (“text2.txt”) in the write mode.
Step 5 : Read the content from the file1 and write into file2.
Step 6 : Close the two files f1 and f2.
Step 7 : Open the filef3 (“text1.txt”) in the read mode.
Step 8 : Read the contents from the file f3 using the method read().split() and stored it in the variables “words”.
Step 9 : Print the word count in the file.
Step 10 : Using for loop and if statement, print the longest word in the file.
Step 11 : Stop the program.

Program:

```
from shutil import copyfile
sfile = input("Enter Source File's Name: ")
tfile = input("Enter Target File's Name: ")
copyfile(sfile, tfile)
print(“File successfully copied”)
f = open(tfile, "r")
data = f.read()
words = data.split(" ")
i=0
wordcount= {}
while i < len(words):
        item = words[i]
        wordcount[item] = words.count(item)
        i=i+1
        print(wordcount)
        f.close()
```

Output:

File successfully copied

#myfile.txt

hello world hello python

{'hello': 2, 'world': 1, 'python': 1}

Result :

Thus the python program for implementing the file handling operation for performing word count in the file, longest word in the file and copy the content of one file into another was implemented successfully.

Ex.No: 10a	IMPLEMENTING DIVIDE BY ZERO ERROR USING EXCEPTION HANDLING
Date:	

Aim :

To write a python program to implement divide by zero error using exception handling.

Algorithm :

Step 1 : Start the program.

Step 2 : Get the dividend and divisor values from the user.

Step 3 : Calculate the quotient value using try block.

Step 4 : Check the divide by zero error in catch block.

Step 5 : Print the exception handling output.

Step 6 : Stop the program.

Program:

```
n=int(input("Enter the value of n:"))
d=int(input("Enter the value of d:"))
c=int(input("Enter the value of c:"))
try:
    q = n/(d-c)
    print(“Quotient:”,q)
except ZeroDivisionError:
    Print(“Division by Zero !”)
```

Output:

```
Enter the value of n: 10
Enter the value of d: 5
Enter the value of c: 5
Division by Zero!
```

Result :

Thus the python program for divide by zero error using exception handling was implemented successfully.

Ex.No: 10b	WRITE A PYTHON PROGRAM TO CHECK THE VOTER'S AGE
Date:	

Aim :

To write a python program to check the voter's age validity.

Algorithm :

Step 1 : Start the program.

Step 2 : Get the voter's name and age from the user.

Step 3 : Calculate the eligibility of voter's age in try block.

Step 4 : Check the value error in catch block.

Step 5 : Print the eligibility status of voter's age using exception handling.

Step 6 : Stop the program.

Program:

```
#this program check voting eligibility
def main():
#single try statement can have multiple except statements.
  try:
    age=int(input("Enter your age"))
    if age>18:
      print("Eligible to vote")
    else:
      print("Not eligible to vote")
  except ValueError:
      print("age must be a valid number")
  except IOError:
      print("Enter correct value")
    #generic except clause, which handles any exception.
  except:
```

Output:

Enter your age25
Eligible to vote
Enter your age15
Not eligible to vote

Result :

Thus the program to check the voter's age validity was implemented and the program executed , verified successfully.

Ex.No: 10c	WRITE A PYTHON PROGRAM FOR STUDENT MARK RANGE VALIDATION
Date:	

AIM:

To write a python program for student mark range validation

ALGORITHM:

STEP 1. User must enter 5 different values and store it in separate variables.

STEP 2. Then sum up all the five marks and divide by 5 to find the average of the marks.

STEP 3. If the average is greater than 90, “Grade: A” is printed.

STEP 4. If the average is in between 80 and 90, “Grade: B” is printed.

STEP 5. If the average is in between 70 and 80, “Grade: C” is printed.

STEP 6. If the average is in between 60 and 70, “Grade: D” is printed.

STEP 7. If the average is anything below 60, “Grade: F” is printed.

PROGRAM:

```
sub1=int(input("Enter marks of the first subject: "))
sub2=int(input("Enter marks of the second subject: "))
sub3=int(input("Enter marks of the third subject: "))
sub4=int(input("Enter marks of the fourth subject: "))
sub5=int(input("Enter marks of the fifth subject: "))
avg=(sub1+sub2+sub3+sub4+sub4)/5
avg = round(avg)
if(avg>=90):
   print("Grade: A")
elif(avg>=80&avg<90):
    print("Grade: B")
elif(avg>=70&avg<80):
    print("Grade: C")
elif(avg>=60&avg<70):
    print("Grade: D")
else:
   print("Grade: F")
```

Output:

Case 1:

Enter marks of the first subject: 85

Enter marks of the second subject: 95

Enter marks of the third subject: 99

Enter marks of the fourth subject: 93

Enter marks of the fifth subject: 100

Grade: A

Case 2:

Enter marks of the first subject: 81

Enter marks of the second subject: 72

Enter marks of the third subject: 94

Enter marks of the fourth subject: 85

Enter marks of the fifth subject: 80

Grade: B

Result :

Thus the python program for student mark range validation was implemented and checked successfully.

Ex. No:11	EXPLORING PYGAME TOOL
Date:	

Exploring pygame 1 - Discovering the library

Published on January 15, 2017

Game development is one of the most common reasons to start to study programming. With me it was not different, despite not following the game developer path, this was always a field that caught my attention.

I'm creating this series of posts to learn more about the game development basics and to share my discoveries with everyone. I'll use the pygame library as tool and I will start by the most basic principles of game development until the creation of a single player **pong like** game.

To reach this goal I'll start with Structured Programming focused at the basics of **game** and **pygame** to make it more comfortable for beginners. As the series goes I'll refactor the code to introduce Object Oriented Programming concepts.

The code used on this series will be running at **Python 3**

Pygame

Pygame is a set of Python modules designed to make games. It uses SDL which is a cross-platform library that abstracts the multimedia components of a computer as audio and video and allows an easier development of programs that uses this resources.

Installation

Before starting the installation process you must have python installed at your system. In case you don't have it check this installation guide first.

The **pygame** installation itself can be different from an **OS** and can be checked for more details at its [official install guide](http://www.pygame.org/wiki/GettingStarted#Pygame Installation).

But, to install it into a **Debian** based system such as **Ubuntu** you must first make sure its dependencies are installed

1. Open command prompt in windows 7
2. C:\users\admin***\python –m pip install –U pygame***
3. **Numpy files are downloading from Internet and installed successfully.**

CODE

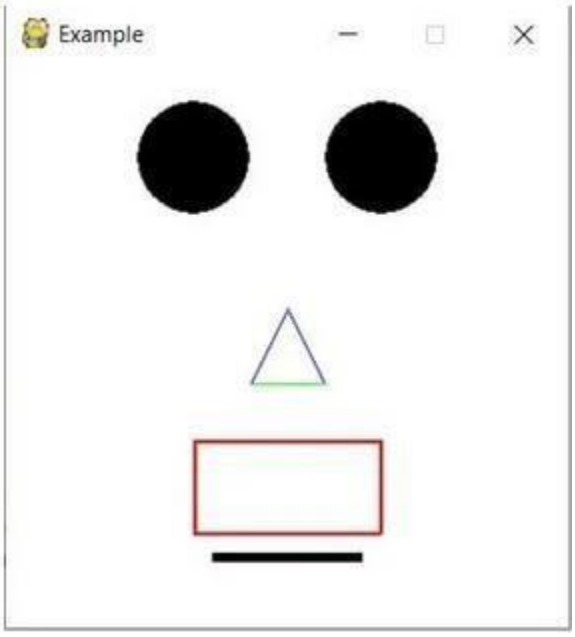
Creating Shapes

```
import pygame, sys
from pygame.locals import *
# Initialize program
pygame.init()
# Assign FPS a value
FPS = 30
FramePerSec = pygame.time.Clock()
# Setting up color objects
BLUE = (0, 0, 255)
RED  = (255, 0, 0)
GREEN = (0, 255, 0)
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
# Setup a 300x300 pixel display with caption
DISPLAYSURF = pygame.display.set_mode((300,300))
DISPLAYSURF.fill(WHITE)
pygame.display.set_caption("Example")
```

```
# Creating Lines and Shapes
pygame.draw.line(DISPLAYSURF, BLUE, (150,130), (130,170))
pygame.draw.line(DISPLAYSURF, BLUE, (150,130), (170,170))
pygame.draw.line(DISPLAYSURF, GREEN, (130,170), (170,170))
pygame.draw.circle(DISPLAYSURF, BLACK, (100,50), 30)
pygame.draw.circle(DISPLAYSURF, BLACK, (200,50), 30)
pygame.draw.rect(DISPLAYSURF, RED, (100, 200, 100, 50), 2)
pygame.draw.rect(DISPLAYSURF, BLACK, (110, 260, 80, 5))

# Beginning Game Loop
while True:
    pygame.display.update()
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    FramePerSec.tick(FPS)
```



Result:

Thus the pygame exploring program was executed successfully without any error and required output isverified.

Ex. No:12a	Game activity using Pygame car race
Date:	

Aim:

To write a Python program to car race in Pygame.

Algorithm:

Step 1: Import the required packages
Step 2: Define the required variables
Step 3: Define the screen space to display the car race in that space

Program:

```
import  random
from time import sleep
import pygame
class CarRacing:
    def__init_(self):
        pygame.init()
        self.display_width = 800
        self.display_height = 600
        self.black = (0, 0, 0)
        self.white = (255, 255, 255)
        self.clock = pygame.time.Clock()
        self.gameDisplay = None
        self.initialize()
    def initialize(self):
        self.crashed = False
        self.carImg = pygame.image.load('.\\img\\car.png')
        self.car_x_coordinate = (self.display_width * 0.45)
        self.car_y_coordinate = (self.display_height * 0.8)
        self.car_width = 49
        # enemy_car
        self.enemy_car = pygame.image.load('.\\img\\enemy_car_1.png')
        self.enemy_car_startx = random.randrange(310, 450)
        self.enemy_car_starty = -600
        self.enemy_car_speed = 5
        self.enemy_car_width = 49
        self.enemy_car_height = 100
        # Background
        self.bgImg = pygame.image.load(".\\img\\back_ground.jpg")
        self.bg_x1 = (self.display_width / 2) - (360 / 2)
        self.bg_x2 = (self.display_width / 2) - (360 / 2)
        self.bg_y1 = 0
        self.bg_y2 = -600
        self.bg_speed = 3
        self.count = 0
```

```
    def car(self, car_x_coordinate, car_y_coordinate):
        self.gameDisplay.blit(self.carImg, (car_x_coordinate, car_y_coordinate))
    def racing_window(self):
        self.gameDisplay = pygame.display.set_mode((self.display_width,
self.display_height))
        pygame.display.set_caption('Car Dodge')
        self.run_car()
    def run_car(self):
        while not self.crashed:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.crashed = True
                # print(event)
                if (event.type == pygame.KEYDOWN):
                    if (event.key == pygame.K_LEFT):
                        self.car_x_coordinate -= 50
                        print ("CAR X COORDINATES: %s" % self.car_x_coordinate)
                    if (event.key == pygame.K_RIGHT):
                        self.car_x_coordinate += 50
                        print ("CAR X COORDINATES: %s" % self.car_x_coordinate)
                    print ("x: {x}, y: {y}".format(x=self.car_x_coordinate, y=self.car_y_coordinate))
            self.gameDisplay.fill(self.black)
            self.back_ground_raod()
            self.run_enemy_car(self.enemy_car_startx, self.enemy_car_starty)
            self.enemy_car_starty += self.enemy_car_speed
            if self.enemy_car_starty > self.display_height:
                self.enemy_car_starty = 0 - self.enemy_car_height
                self.enemy_car_startx = random.randrange(310, 450)
            self.car(self.car_x_coordinate, self.car_y_coordinate)
            self.highscore(self.count)
            self.count += 1
            if (self.count % 100 == 0):
                self.enemy_car_speed += 1
                self.bg_speed += 1
            if self.car_y_coordinate < self.enemy_car_starty + self.enemy_car_height:
                if self.car_x_coordinate > self.enemy_car_startx and self.car_x_coordinate <
self.enemy_car_startx + self.enemy_car_width or self.car_x_coordinate + self.car_width >
self.enemy_car_startx and self.car_x_coordinate + self.car_width < self.enemy_car_startx
+ self.enemy_car_width:
                    self.crashed = True
                    self.display_message("Game Over !!!")
```

```
            if self.car_x_coordinate < 310 or self.car_x_coordinate > 460:
                self.crashed = True
                self.display_message("Game Over !!!")
            pygame.display.update()
            self.clock.tick(60)
    def display_message(self, msg):
        font = pygame.font.SysFont("comicsansms", 72, True)
        text = font.render(msg, True, (255, 255, 255))
        self.gameDisplay.blit(text, (400 - text.get_width() // 2, 240 - text.get_height() // 2))
        self.display_credit()
        pygame.display.update()
        self.clock.tick(60)
        sleep(1)
        car_racing.initialize()
        car_racing.racing_window()
    def back_ground_raod(self):
        self.gameDisplay.blit(self.bgImg,  (self.bg_x1,  self.bg_y1))
        self.gameDisplay.blit(self.bgImg,  (self.bg_x2,  self.bg_y2))
        self.bg_y1 += self.bg_speed
        self.bg_y2 += self.bg_speed
        if self.bg_y1 >= self.display_height:
            self.bg_y1 = -600
        if self.bg_y2 >= self.display_height:
            self.bg_y2 = -600
    def run_enemy_car(self, thingx, thingy):
        self.gameDisplay.blit(self.enemy_car, (thingx, thingy))
    def highscore(self, count):
        font = pygame.font.SysFont("arial", 20)
        text = font.render("Score : " + str(count), True, self.white)
        self.gameDisplay.blit(text, (0, 0))
    def display_credit(self):
        font = pygame.font.SysFont("lucidaconsole", 14)
        text = font.render("Thanks for playing!", True, self.white)
        self.gameDisplay.blit(text, (600, 520))
if__name__== '_main__':
    car_racing = CarRacing()
    car_racing.racing_window()
```

Result:

Thus the program executed successfully without any error and required output is verified.

Ex.No: 12b	BOUNCING BALL IN PYGAME
Date:	

Aim:
To write a Python program to bouncing ball in Pygame.

Algorithm:
Step 1: Import the required packages
Step 2: Define the required variables
Step 3: Define the screen space to display the bouncing balls in that space

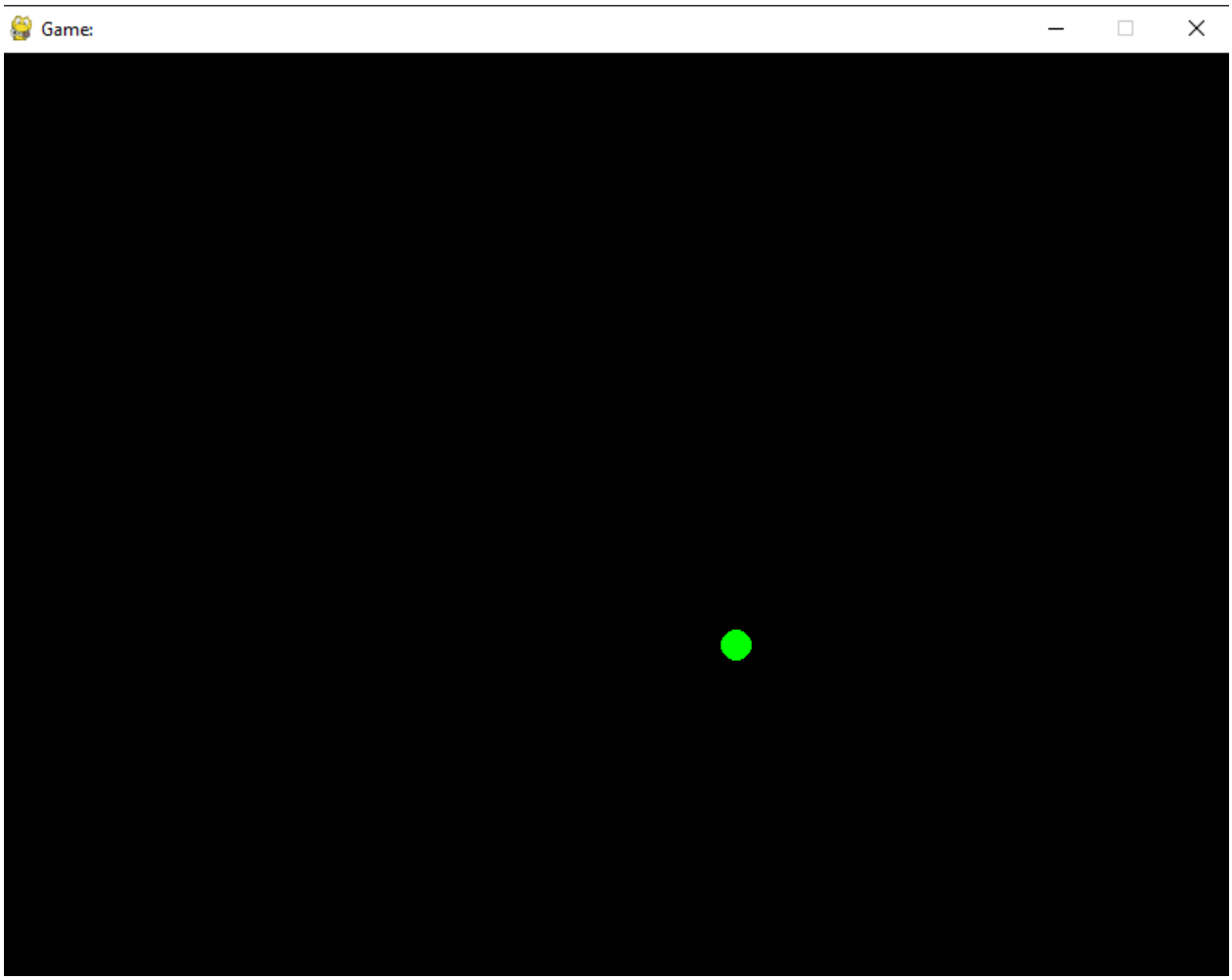
Program:

```
# SIMULATE BOUNCING BALL.
import pygame
pygame.init()
window_w=800
window_h=600
white=(255,255,255)
green=(0,255,0)
black=(0,0,0)
FPS=120
window=pygame.display.set_mode((window_w,window_h))
pygame.display.set_caption("Game: ")
clock = pygame.time.Clock()
BLACK=(0,0,0)

def game_looop():
    block_size=20
    lock_size=10
    radius=10
    velocity=[1,1]
    pos_x=window_w/2
    pos_y=window_h/2
    running = True
    while running:
        for event in pygame.event.get():
            if event.type==pygame.QUIT:
                pygame.quit()
                quit()
        pos_x +=velocity[0]
        pos_y +=velocity[1]
        if pos_x + block_size > window_w or pos_x <0:
            velocity[0]=-velocity[0]
        if pos_y + block_size > window_h or pos_y <0:
            velocity[1]=-velocity[1]
        #window.fill(white)
        window.fill(black)
        #pygame.draw.rect(window,black,[pos_x,pos_y,block_size,lock_size])
        pygame.draw.circle(window,green,[pos_x,pos_y],radius)
        pygame.display.update()
        clock.tick(FPS)

game_looop()
```

Output:



Result:

Thus the program executed successfully without any error and required output is verified.