DMI COLLEGE OF ENGINEERING
PALANCHUR, CHENNAI

DEPARTMENT OF INFORMATION TECHNOLOGY

PROBLEM SOLVING AND

PYTHON PROGRAMMING LABORATORY

(GE3171)

LAB MANUAL

(Regulation-2021)

SUB CODE : GE3171

SUBJECT TITLE: Problem Solving and Python Programming

SEMESTER : I

YEAR : I

DEPARTMENT : INFORMATION TECHNOLOGY

SYLLABUS

GE3171 PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY

L T P C - 0 0 4 2

OBJECTIVES:

- To understand the problem solving approaches.
- To learn the basic programming constructs in Python.
- To practice various computing strategies for Python-based solutions to real world problems.
- To use Python data structures - lists, tuples, dictionaries.
- To do input/output with files in Python.

LIST OF PROGRAMS:

1. Identification and solving of simple real life or scientific or technical problems, and developing flow charts for the same. (Electricity Billing, Retail shop billing, Sin series, weight of a motorbike, Weight of a steel bar, compute Electrical Current in Three Phase AC Circuit, etc.)
2. Python programming using simple statements and expressions (exchange the values of two variables, circulate the values of n variables, distance between two points).
3. Problems using Conditionals and Iterative loops. (Number series, Number & pyramid pattern).
4. Implementing real-time/technical applications using Lists, Tuples. (Items present in a library/Components of a car/ Materials required for construction of a building).
5. Implementing real-time/technical applications using Sets, Dictionaries. (Language, components of an automobile, Elements of a civil structure, etc.)
6. Implementing programs using Functions. (Factorial, largest number in a list, area of shape)
7. Implementing programs using Strings. (Reverse, palindrome, count and replace characters)
8. Implementing programs using written modules and Python Standard Libraries (pandas, numpy, Matplotlib, scipy)
9. Implementing real-time/technical applications using File handling. (copy from one file to another, word count, longest word)
10. Implementing real-time/technical applications using Exception handling. (divide by zero error, voter's age validity, student mark range validation)
11. Exploring Pygame tool.
12. Developing a game activity using Pygame like bouncing ball, car race etc.

TOTAL: 60 PERIODS

Subject Code & Name: GE3171 - Problem Solving and Python Programming Laboratory

Branch : IT

Year/Semester : I/01

Course Outcomes

CO1	Develop algorithmic solutions to simple computational problems
CO2	Develop and execute simple Python programs
CO3	Implement programs in python using conditionals and loops for solving problems
CO4	Deploy functions to decompose a Python Program
CO5	Process compound data using python data structures
CO6	Utilize python packages in developing software applications

CO, PO, PSO Mappings.

Course Code and Course name	CO	Program Outcomes												PSO 3		
		1	2	3	4	5	6	7	8	9	10	11	12	1	2	3
GE3171 - Problem Solving and Python Programming Laboratory	CO1	3	3	3	3	2	-	-	-	-	-	2	2	3	3	-
	CO2	3	3	3	3	2	-	-	-	-	-	2	2	3	-	-
	CO3	3	3	3	3	2	-	-	-	-	-	2	-	3	-	-
	CO4	2	2	-	2	2	-	-	-	-	-	1	-	3	-	-
	CO5	1	2	-	-	1	-	-	-	-	-	1	-	2	-	-
	CO6	2	2	-	-	2	-	-	-	-	-	1	-	2	-	-
Average		2	3	3	3	2	-	-	-	-	-	2	2	3	3	-

Subject Code & Name : **GE3171 - Problem Solving and Python Programming Laboratory**

Branch : **IT**

Year/Semester : **I / 01**

LIST OF EXPERIMENTS (As per Syllabus)

S.NO	NAME OF THE EXPERIMENTS
1	Developing flowcharts for simple real life scientific or technical problems
2	Simple statements and expressions (swap two values, circulate the values of n variables, distance between two points)
3	Conditionals and iterative loops (number series, number patterns and pyramid pattern)
4	a. Operations on list
	b. Operations on tuple
5	a. Operations on sets
	b. Operations on dictionaries
6	Functions
7	Strings (reverse, palindrome, character count, replacing characters)
8	Programs with written modules from python standard libraries
9	Word count, longest word and file copy operations in file
10	Real-time/technical application for exception handling
11	Exploring pygame tool
12	Develop a game activity using pygame

INDEX

Sl. No.	Date	Program Name	Mark	Signature

EX.No: 1 **DATE:**	**DEVELOPING FLOWCHARTS FOR SIMPLE REAL LIFE SCIENTIFIC OR TECHNICAL PROBLEMS**

AIM:

To identify and solve simple real life scientific or technical problems, and developing flow charts for the problems such as Electricity Billing, Retail shop billing, Sin series, and Electrical Current in Three Phase AC Circuit, etc.

ELECTRICAL CURRENT IN THREE PHASE AC CIRCUIT

The formula for power in Three Phase AC Circuit is given as,

Power = √3 * Power factor * Voltage * Current

ALGORITHM:

Step 1: Start the program.

Step 2: Print "Enter Power Factor, Voltage and Current "

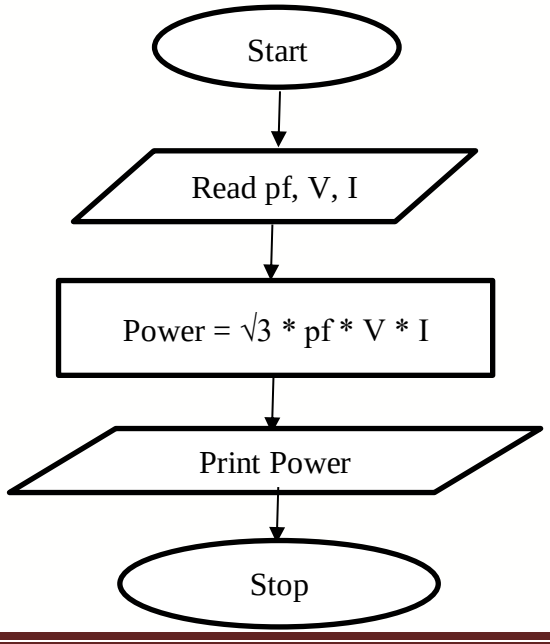
Step 3: Read pf, V, I

Step 4: Power = √3 * pf * V * I

Step 5: Print Power.

Step 6: Stop the program.

FLOWCHART:

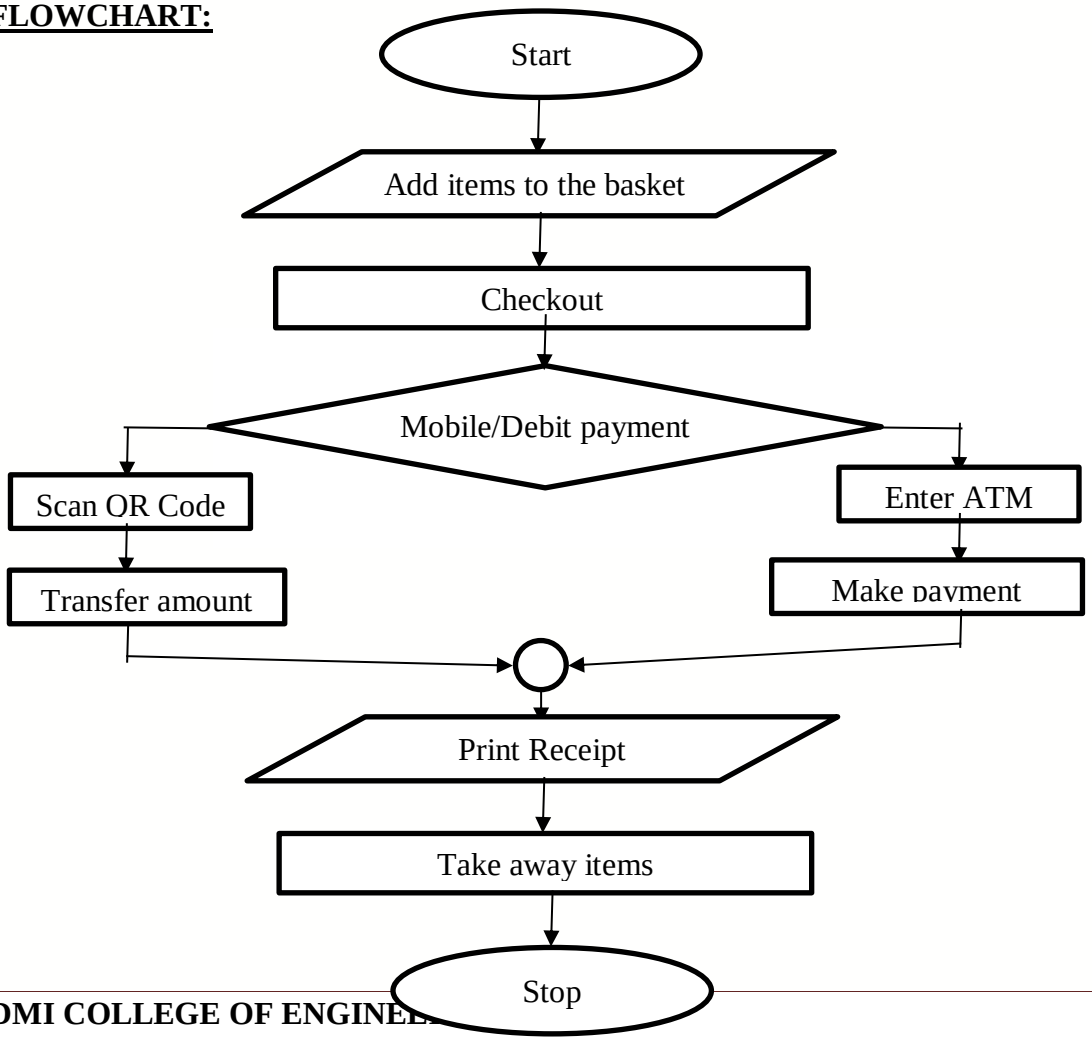


RETAIL SHOP BILLING

ALGORITHM:

1. Start the program.
2. Add items to the basket.
3. Checkout the items at the billing counter.
4. Select any one of the payment options, either mobile payment or debit card payment.
5. If mobile payment, scan the QR Code in the shop.
6. Transfer amount to the account.
7. If Debit card, Enter the ATM PIN.
8. Make payment from Debit card.
9. Print receipt.
10. Take away the items in the basket.
11. Stop the program.

FLOWCHART:

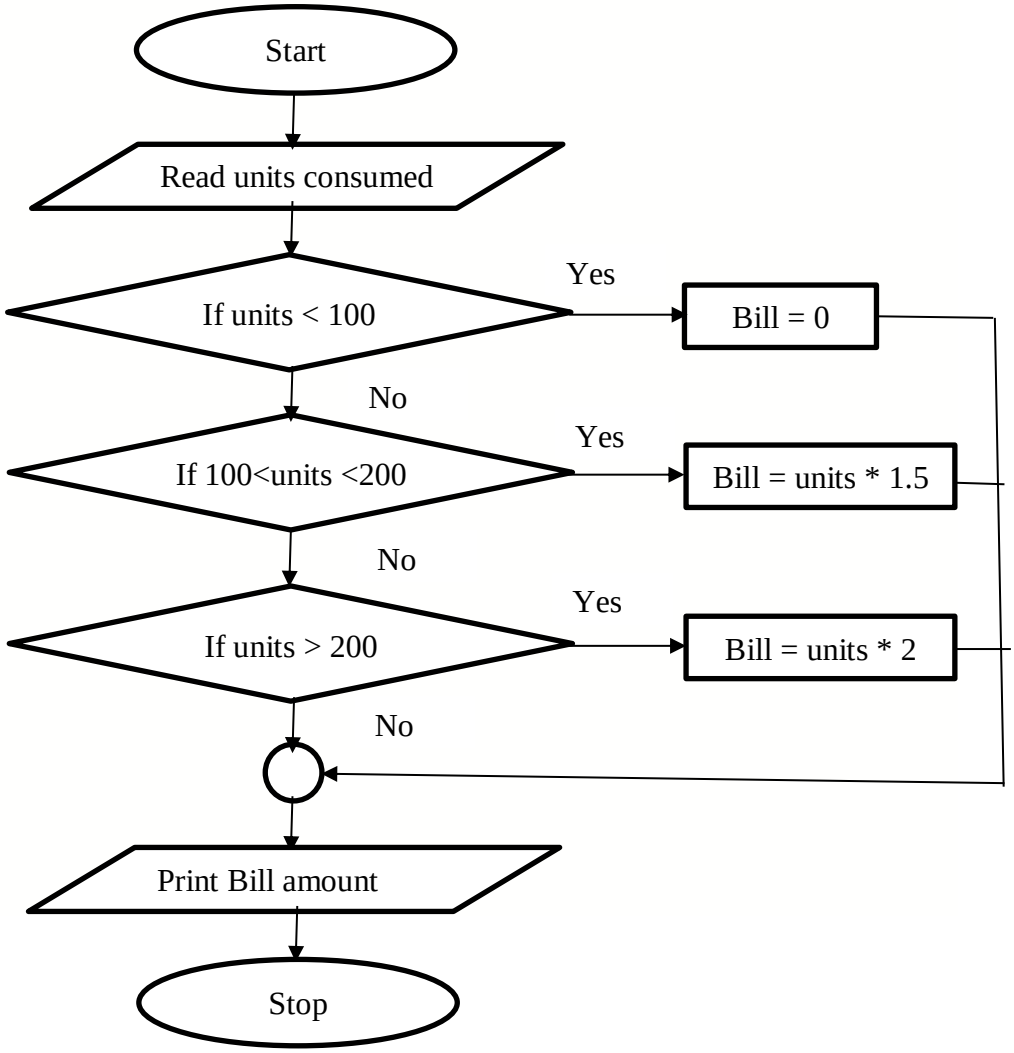


ELECTIRICITY BILLING

ALGORITHM:

Step 1: Start the program.
Step 2: Print "Enter units consumed ".
Step 3: Read units.
Step 4: If units < 100, then bill amount = 0.
Step 5: Else if 100 < units < 200, then bill amount = 1.5 * units.
Step 6: Else if units > 200, then bill amount = 2 * units.
Step 7: Print the bill amount.
Step 8 : Stop the program.

FLOW CHART:



SINE SERIES:

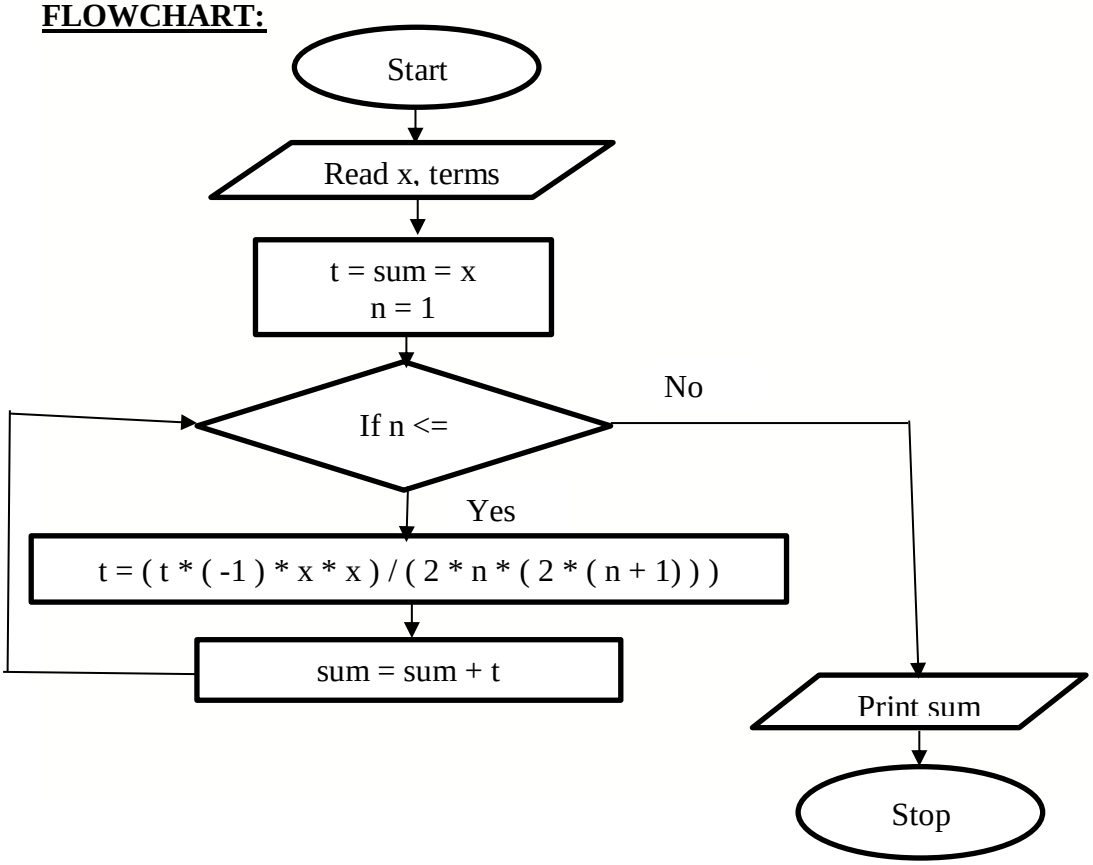
The sine series formula is given as

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \text{ for all } x$$

ALGORITHM:

Step 1: Start the program.
Step 2: Print "Enter the values of x and terms : ".
Step 3: Read x, terms.
Step 4: t = sum = x
Step 5: Initialize n = 1.
Step 6: If n <= terms then goto step 8.
Step 7: else goto step 11.
Step 8: t = (t * (-1) * x * x) / (2 * n * (2 * (n + 1)))
Step 9: sum = sum + t
Step 10: goto step 6.
Step 11: Print sum.
Step 12: Stop the program.

FLOWCHART:



RESULT:

The flowcharts were drawn to identify and solve simple real life scientific or technical problems successfully.

Ex.No :2	**SIMPLE STATEMENTS AND EXPRESSIONS** (Swap two values, circulate the values of n variables, distance between two points)
Date :	

AIM:

To write a Python program to swap two numbers, circulate the values of n variables and find distance between two points using function.

i) SWAP TWO VALUES:

ALGORITHM:

1. Start the program.
2. Read two numbers.
3. Print the two numbers before swapping.
4. Define a function swap(a,b) to exchange the values of two variables.
5. Swap the variables using tuple assignment ie)a,b=b,a.
6. Print the values after swapping.
7. Stop.

PROGRAM:

```
def swap( a, b ):
        a, b = b, a
        print "After Swap : "
        print " First Number : ", a
        print " Second Number : ", b
a = input( "Enter the first number : " )
b = input( "Enter the second number : " )
print "Before Swap : "
print " First Number : ", a
print "Second Number : ", b
```

swap(a, b)

OUTPUT:

Enter the first number : 2

Enter the second number : 3

Before Swap:

First Number : 2

Second Number : 3

After Swap:

First Number : 3

Second Number : 2

ii) CIRCULATE THE VALUES OF N VARIABLES:

ALGORITHM:

1. Start
2. Create a list with few values.
3. Print the original list.
4. Rotate the list by 1, 2, 3 and 4 values and print the list after each rotation.
5. Stop.

PROGRAM:

```
def  rotate(L,n):
     newlist=L[ n: ]+L[ :n ]
      return  newlist
list = [1,2,3,4,5]
print("The  original  list is:",list)
mylist=rotate(list,1)
print("List rotated clockwise by 1:",mylist)
mylist=rotate(list,2)
print("List rotated clockwise by 2:",mylist)
mylist=rotate(list,3)
print("List rotated clockwise by 3:",mylist)
```

```
mylist=rotate(list,4)
print("List rotated clockwise by 4:",mylist)
```

OUTPUT:

```
The original list is: [1, 2, 3, 4, 5]
List rotated clockwise by 1: [2, 3, 4, 5, 1]
List rotated clockwise by 2: [3, 4, 5, 1, 2]
List rotated clockwise by 3: [4, 5, 1, 2, 3]
List rotated clockwise by 4: [5, 1, 2, 3, 4]
```

iii) DISTANCE BETWEEN TWO POINTS:

ALGORITHM:

1. Start
2. Print "Enter the coordinates of two points : "
3. Input x1, y1, x2, y2.
4. distance = sqrt($(x2 - x1)^2 + (y2 – y1)^2$)
5. Print distance
6. Stop

PROGRAM:

```
import math
def  distance(x1,y1,x2,y2):
    dx=x2 - x1
    dy=y2 - y1
    dsquare=dx**2 + dy**2
    result=math . sqrt(dsquare)
    return result
x1=int(input("Enter the value of x1:"))
y1=int(input("Enter the value of y1:"))
x2=int(input("Enter the value of x2:"))
y2=int(input("Enter the value of y2:"))
print("The distance between two points:" ,distance(x1,y1,x2,y2))
```

OUTPUT:

```
Enter the value of x1:2
Enter the value of y1:4
Enter the value of x2:3
Enter the value of x1:6
The distance between two points:2.23
```

RESULT:

Thus the Python program to swap two numbers, circulate the values of n variables and find distance between two points using function was executed and verified successfully.

EX.No: 3	**CONDITIONALS AND ITERATIVE LOOPS** (Number Series, Number Patterns and Pyramid Pattern)
DATE:	

AIM:

To implement Python program to print Number Series, Number Patterns and Pyramid Pattern using Conditionals and Iterative loops.

i) NUMBER SERIES:

ALGORITHM:

1. Start
2. Print “Enter a number “.
3. Read n.
4. Using for loop, print all the numbers one by one from 1 to n.
5. Using for loop, print all odd numbers one by one from 1 to n.
6. Using for loop, print all even numbers one by one from 1 to n.
7. Stop.

PROGRAM:

```
n = int( input( "Enter a number : ") )
print ( "\n Numbers from 1 to n" )
for i in range(1,n+1,1):
  print ( i , end = " ")
print ( "\n Odd Numbers from 1 to n" )
for i in range(1,n+1,2):
  print ( i , end = " ")
print ( "\n Even Numbers from 1 to n" )
for i in range(2,n+1,2):
  print ( i , end = " ")
```

OUTPUT:

```
Enter a number : 5
 Numbers from 1 to n
1       2       3       4       5
 Odd Numbers from 1 to n
```

1 3 5

Even Numbers from 1 to n

2 4

ii) PYRAMID PATTERN:

ALGORITHM:

1. Start
2. Print “Enter a number “.
3. Read num.
4. Using two for loops, leave space before the *’s.
5. Using another for loop, print *’s to make a Pyramid Pattern.
6. Stop.

PROGRAM:

```
num = int( input( "Enter a number : " ) )
for i in range( num ):
    for j in range( ( num - i ) - 1 ) :
        print( end = " " )
    for j in range( i + 1 ):
        print( "*", end = " " )
    print()
```

OUTPUT:

```
Enter a number : 5
    *
   * *
  * * *
 * * * *
* * * * *
```

iii) NUMBER PATTERN:

ALGORITHM:

1. Start
2. Print “Enter a number “.
3. Read num.

4. Using two for loops, leave space before the numbers.
5. Using another for loop, print the number to make a Number Pattern.
6. Stop.

PROGRAM:

```
num = int(input("Enter a number : "))
for i in range( num ):
  for j in range( ( num - i ) - 1 ) :
    print( end = " " )
  for j in range(i + 1):
    print(j + 1, end=" ")
  print()
```

OUTPUT:

```
Enter a number : 5
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```

RESULT:

The Python program to implement Number Series, Number Patterns and Pyramid Pattern using Conditionals and Iterative loops were implemented and executed successfully.

Ex.No :4a Date :	**OPERATIONS ON LIST** **(Items Present in the Library)**

AIM :

To Perform the Operations of List on items present in the Library using Python Programming.

ALGORITHM :

1. Start the program.
2. Create two lists with the list of items present in the Library.
3. Perform operations such as concatenation, repetition, finding maximum and minimum in the list.
4. Add and remove elements to and from the list, copy the list.
5. Clear the list.
6. Stop the program.

PROGRAM:

```
libList1 = ["books", "newspapers", "ebooks", "journals", "magazine"]
libList2 = ["catalog", "computer", "entryRegister", "printer", "maps"]
libList = libList1 + libList2
print("Concatenated list : ", libList)
print("Repetition : ", libList2 * 2)
print("Length : ", len(libList1))
print("Max : ", max(libList1))
print("Min : ", min(libList1))
print("Membership (in) : ", "books" in libList1)
print("Membership (not in) : ", "ebooks" not in libList1)
lib = libList1.copy()
print("Copy : ", lib)
print("Index : ", libList1.index("books"))
libList1.sort()
```

```
print("Sorted List : ", libList1)
libList1.reverse()
print("Reversed List : ", libList1)
libList1.append("dvds")
print("Appended List : ", libList1)
libList1.insert(3, "cds")
print("Inserted List : ", libList1)
libList2.pop()
print("Popped List : ", libList2)
libList2.remove("computer")
print("Removed List : ", libList2)
libList2.clear()
print("Cleared List : ", libList2)
```

OUTPUT:

```
Concatenated list :  ['books', 'newspapers', 'ebooks', 'journals', 'magazine', 'catalog',
'computer', 'entryRegister', 'printer', 'maps']
Repetition :  ['catalog', 'computer', 'entryRegister', 'printer', 'maps', 'catalog',
'computer', 'entryRegister', 'printer', 'maps']
Length :  5
Max :  newspapers
Min :  books
Membership (in) :  True
Membership (not in) :  False
Copy :  ['books', 'newspapers', 'ebooks', 'journals', 'magazine']
Index :  0
Sorted List :  ['books', 'ebooks', 'journals', 'magazine', 'newspapers']
Reversed List :  ['newspapers', 'magazine', 'journals', 'ebooks', 'books']
Appended List :  ['newspapers', 'magazine', 'journals', 'ebooks', 'books', 'dvds']
Inserted List :  ['newspapers', 'magazine', 'journals', 'cds', 'ebooks', 'books', 'dvds']
Popped List :  ['catalog', 'computer', 'entryRegister', 'printer']
```

```
Removed List :  ['catalog', 'entryRegister', 'printer']
Cleared List :  []
```

RESULT:

Thus the Python Program to perform the operations of list was executed and verified successfully.

Ex.No :4b	**OPERATIONS ON TUPLE** **(Components of a Car)**
Date :	

AIM :

To perform the various operations of Tuple with the components of a Car using Python Programming.

ALGORITHM :

1. Start the program.
2. Create two tuples with the list of items present in the car.
3. Perform operations such as concatenation, repetition, finding maximum and minimum in the tuple.
4. Add and remove elements to and from the tuple and copy the tuple.
5. Clear the tuple.
6. Stop the program.

PROGRAM:

```
carSet1={}
print("Empty Set : ", carSet1)
carSet1 = {"Wheels", "Engine", "Brake", "Gear", "Sensors","Chassis"}
print("carSet1 : ", carSet1)
carSet2 = {"Speaker", "Accelator", "mirrors", "dashboard", "battery"}
print("carSet2: ", carSet2 )
carSet1.update(carSet2)
print("Concatenated Set : ", carSet1)
print("Repetition : ", carSet2 * 2)
print("Length : ", len(carSet1))
print("Max : ", max(carSet1))
print("Min : ", min(carSet1))
print("Membership (in) : ", "Engine" in carSet1)
print("Membership (not in) : ", "Brake" not in carSet1)
print("Index : ", carSet1.index("Gear"))
print("Count : ", carSet1.count("Gear"))
print("Sorted Tuple : ", sorted(carSet1))
T=set("CarJewels")
print("String to Tuple:",T)
L=['C', 'a', 'r', 'J', 'e', 'w', 'e', 'l', 's']
T1=tuple(L)
print("List to Tuple:",T1)
tup1 = ( 1, 2, 3 )
```

```
print("Sum of elements :",sum(tup1))

T2 = carTup1[::-1]

print("Reversed tuple:",T2)

del(carTup1)

print("Deleted Tuple:",carTup1)
```

OUTPUT:

```
Empty Tuple :  ()
carTup1 :  ('Wheels', 'Engine', 'Brake', 'Gear', 'Sensors', 'Chassis')
carTup2 :  ('Speaker', 'Accelator', 'mirrors', 'dashboard', 'battery')
Concatenated Tuple :  ('Wheels', 'Engine', 'Brake', 'Gear', 'Sensors', 'Chassis', 'Speaker',
'Accelator', 'mirrors', 'dashboard', 'battery')
Repetition :  ('Speaker', 'Accelator', 'mirrors', 'dashboard', 'battery', 'Speaker', 'Accelator',
'mirrors', 'dashboard', 'battery')
Length :  6
Max :  Wheels
Min :  Brake
Membership (in) :  True
Membership (not in) :  False
Index :  3
Count :  1
Sorted Tuple :  ['Brake', 'Chassis', 'Engine', 'Gear', 'Sensors', 'Wheels']
String to Tuple: ('C', 'a', 'r', 'J', 'e', 'w', 'e', 'l', 's')
List to Tuple: ('C', 'a', 'r', 'J', 'e', 'w', 'e', 'l', 's')
Sum of elements : 6
Reversed tuple: ('Chassis', 'Sensors', 'Gear', 'Brake', 'Engine', 'Wheels')
Traceback (most recent call last):
  File "E:/nisha/tuple.py", line 28, in <module>
```

```
    print("Cleared Tuple:",carTup1)
NameError: name 'carTup1' is not defined
```

RESULT:

Thus the Python Program to perform the operations of tuple was executed and verified successfully.

Ex.No :5a	OPERATIONS ON SETS
Date :	

AIM :

To implement Python program to perform the operations on Sets with the components of a Car.

ALGORITHM :

1. Start the program.
2. Create two sets with the list of items present in the car.
3. Perform operations such as concatenation, finding length, maximum and minimum of the set.
4. Discard, remove and pop elements from the set and clear the set.
5. Stop the program.

PROGRAM:

```
carSet1={}
print("Empty Set : ", carSet1)
carSet1 = {"Wheels", "Engine", "Brake", "Gear", "Sensors","Chassis"}
print("carSet1 : ", carSet1)
carSet2 = {"Speaker", "Accelator", "mirrors", "dashboard", "battery"}
print("carSet2: ", carSet2 )
carSet1.update(carSet2)
print("Concatenated Set : ", carSet1)
print("Length : ", len(carSet1))
print("Max : ", max(carSet1))
print("Min : ", min(carSet1))
print("Membership (in) : ", "Engine" in carSet1)
print("Membership (not in) : ", "Brake" not in carSet1)
print("Sorted Set : ", sorted(carSet1))
T=set("CarJewels")
```

```
print("String to Set:",T)
S=['C', 'a', 'r', 'J', 'e', 'w', 'e', 'l', 's']
T1=set(S)
print("List to Set:",T1)
my_set = {1, 3, 4, 5, 6}
print("Sum of elements :",sum(my_set))
my_set.discard(4)
print("Discard Set : ", my_set)
my_set.remove(6)
print("Removed Set : ", my_set)
my_set.pop()
print("Popped Set : ", my_set)
my_set.clear()
print("Cleared Set : ", my_set)
```

OUTPUT:

```
Empty Set :  {}
carSet1 :  {'Sensors', 'Gear', 'Wheels', 'Chassis', 'Engine', 'Brake'}
carSet2:  {'battery', 'Speaker', 'Accelator', 'mirrors', 'dashboard'}
Concatenated Set :  {'battery', 'Sensors', 'Speaker', 'mirrors', 'Accelator', 'Gear', 'Wheels',
'Chassis', 'Engine', 'dashboard', 'Brake'}
Length :  11
Max :  mirrors
Min :  Accelator
Membership (in) :  True
Membership (not in) :  False
Sorted Set :  ['Accelator', 'Brake', 'Chassis', 'Engine', 'Gear', 'Sensors', 'Speaker', 'Wheels',
'battery', 'dashboard', 'mirrors']
String to Set: {'r', 'a', 'e', 'w', 'J', 's', 'l', 'C'}
List to Set: {'r', 'a', 'e', 'w', 'J', 's', 'l', 'C'}
Sum of elements : 19
```

Discard Set : {1, 3, 5, 6}

Removed Set : {1, 3, 5}

Popped Set : {3, 5}

Cleared Set : set()

RESULT:

Thus the Python Program to perform operations on set was executed successfully.

Ex.No :5b	**OPERATIONS ON DICTIONARIES**
Date :	

AIM :

To implement Python program to perform the operations on Dictionaries with the components of a Car.

ALGORITHM :

1. Start the program.
2. Create dictionaries with the list of items present in the car.
3. Perform operations such as, fromkeys(), sorted(), finding length of the dictionaries.
4. Discard, remove and pop elements from the dictionaries and clear the dictionaries.
5. Stop the program.

PROGRAM:

```
my_dict = {'name': 'Wheels', 'no': 4}
print( "Dictionary : ", my_dict )
print("Name : ", my_dict['name'])
print("No : ", my_dict.get('no'))
print("Address : ", my_dict.get('address'))
my_dict['no'] = 5
print("Edited Dictionary : ", my_dict)
my_dict['wheels'] = 'alloy'
print("Updated Dictionary : ", my_dict)
car = {}.fromkeys(['Gear', 'Brake', 'Accelerator'], 0)
print("Dictionary : ", car)
for item in car.items():
   print(item)
print("Sortd Dictionary : ", list(sorted(car.keys())))
print("Mambership : ", 1 in car)
print("Mambership : ", 2 not in car)
```

```
squares = {0: 0, 1: 1, 3: 9, 4 : 16, 5: 25, 7: 49, 9: 81}
print( "Dictionary : ", squares )
print("All of Dictionary : ", all(squares))
print("Any of Dictionary : ", any(squares))
print("Length : ", len(squares))
print("Popped Dictionary : ", squares.pop(4))
print("Popped Item : ", squares.popitem())
print("Dictionary : ", squares)
squares.clear()
print(squares)
```

OUTPUT:

```
Dictionary :  {'name': 'Wheels', 'no': 4}
Name :  Wheels
No :  4
Address :  None
Edited Dictionary :  {'name': 'Wheels', 'no': 5}
Updated Dictionary :  {'name': 'Wheels', 'no': 5, 'wheels': 'alloy'}
Dictionary :  {'Gear': 0, 'Brake': 0, 'Accelerator': 0}
('Gear', 0)
('Brake', 0)
('Accelerator', 0)
Sortd Dictionary :  ['Accelerator', 'Brake', 'Gear']
Mambership :  False
Mambership :  True
Dictionary :  {0: 0, 1: 1, 3: 9, 4: 16, 5: 25, 7: 49, 9: 81}
All of Dictionary :  False
Any of Dictionary :  True
Length :  7
Popped Dictionary :  16
Popped Item :  (9, 81)
Dictionary :  {0: 0, 1: 1, 3: 9, 5: 25, 7: 49}
```

Name : Wheels

No : 4

Address : None

Edited Dictionary : {'name': 'Wheels', 'no': 5}

Updated Dictionary : {'name': 'Wheels', 'no': 5, 'wheels': 'alloy'}

Dictionary : {'Gear': 0, 'Brake': 0, 'Accelerator': 0}

('Gear', 0)

('Brake', 0)

('Accelerator', 0)

Sortd Dictionary : ['Accelerator', 'Brake', 'Gear']

Mambership : False

Mambership : True

Dictionary : {0: 0, 1: 1, 3: 9, 4: 16, 5: 25, 7: 49, 9: 81}

All of Dictionary : False

Any of Dictionary : True

Length : 7

Popped Dictionary : 16

Popped Item : (9, 81)

Dictionary : {0: 0, 1: 1, 3: 9, 5: 25, 7: 49}

{}

RESULT:

Thus the Python Program to perform operations on dictionaries was executed successfully.

Ex.No : 6	FUNCTIONS
Date :	

AIM :

To implement Python program to find factorial of a number, area of shapes and maximum and minimum element in a list.

i) FACTORIAL – USER DEFINED FUNCTION

ALGORITHM :

1. Start
2. Read n
3. Call the function fact(n)
4. Print value of the factorial
5. Stop

fact(n) function:

1. Start function
2. If n=0, return 1
3. If n>0 return n * fact(n-1)
4. Return

PROGRAM:

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)

n = int( input( "Enter a number : " ) )
fact = factorial(n)
print( "Factorial of ", n, " is ", fact )
```

OUTPUT:

Enter a number : 4

Factorial of 4 is 24

ii) AREA OF SHAPES – USER DEFINED FUNCTION

ALGORITHM

1. Start the program.
2. Create a function circleArea to find the area of the circle.
3. Create a function rectangleArea to find the area of the rectangle.
4. Call circleArea.
5. Call rectangleArea.
6. Stop the program.

PROGRAM

```
def circleArea( ):
    r = int( input( "Enter radius : " ) )
    area = 3.14 * r * r
    print("The area of circle is ", area)
def rectangleArea():
    l = int(input("Enter rectangle's length: "))
    b = int(input("Enter rectangle's breadth: "))
    area = l * b
    print("The area of rectangle is ", area)
circleArea()
rectangleArea( )
```

OUTPUT

Enter radius : 4

The area of circle is 50.24

Enter rectangle's length: 5

Enter rectangle's breadth: 4

The area of rectangle is 20

iii) MAX AND MIN IN LIST – BUILT-IN FUNCTION

ALGORITHM

1. Start the program.
2. Create an empty list named lst.
3. Read the value of num(total number of elements).
4. Read the elements of the list until num.
5. Print the value of maximum number using max() built-in function.
6. Print the value of maximum number using min() built-in function.
7. Stop

PROGRAM

```
lst = [ ]
num  = int(input("Enter the total number of elements:"))
for i in range(num):
        numbers = int(input("Enter the number:"))
        lst . append (numbers)
print("Maximum element in the list is:", max(lst))
print ("Minimum element in the list is:", min(lst))
```

OUTPUT:

```
Enter the total number of
elements:5
Enter the number:7
Enter the number:85
Enter the number:6
Enter the number: 90
Enter the number: 72
```

Maximum element in the list is: 90

Minimum element in the list is:6

RESULT:

Thus the Python Program to find factorial of a number, maximum and minimum of a list of numbers and area of shapes was executed andverified successfully.

EX.No: 7 **DATE:**	**STRINGS** **(**Reverse, Palindrome, Character count, Replacing Characters)

AIM:

To implement a program using Strings to reverse, check palindrome, character count and replace characters.

i)REVERSE OF A STRING:

AIM:

Step 1: Start the program.

Step 2: Hold the string in a variable.

Step 3: Reverse the string using a for loop.

Step 4: Print the result.

Step 5: Stop the program.

PROGRAM:

```
def rev(str):
    str1 = ""
    for i in str:
        str1 = i + str1
    return str1
str = "Python"
print("The original string is: ",str)
print("The reverse string is",rev(str))
```

OUTPUT:

('The original string is: ', 'Python')

('The reverse string is', 'nohtyP')

ii)PALINDROME, CHARACTER COUNT AND REPLACED CHARACTER:

ALGORITHM:

Step 1: Start the program.

Step 2: Hold the string in a variable.

Step 3: Reverse the string.

Step 4: Compare the string with reversed string. If both strings are same, print palindrome, else print not palindrome.

Step 5: Stop the program.

PROGRAM:

```
def palindrome(original):
        rev = original [::-1]
        print("Reversed " , original , " is " , rev)
        if(original == rev):  print("Palindrome")
        else: print("Not a Palindrome")
str1 = "arunachala"
str2 = "madam"
palindrome(str1)
palindrome(str2)
print("Character count ", str1 , " is ", len(str1))
print("Replaced ", str2, " is " , str2.replace('a','o'))
```

OUTPUT:

```
Reversed  arunachala  is  alahcanura
Not a Palindrome
Reversed  madam  is  madam
Palindrome
Character count  arunachala  is  10
Replaced  madam  is  modom
```

RESULT:

Thus the program to implement Strings reverse, check palindrome, character count and replace characters was executed successfully.

EX.No: 8 DATE:	PROGRAMS WITH WRITTEN MODULES FROM PYTHON STANDARD LIBRARIES

AIM:

To implement programs using written modules and Python Standard Libraries (Pandas, Numpy, Matplotlib, Scipy)

i)NUMPY:

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. NumPy is often used along with packages like SciPy (Scientific Python) and Matplotlib (Matrix plotting library). This combination is widely used as a replacement for MatLab.

Operations using NumPy

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

ALGORITHM:

Step 1: Start the program.

Step 2: Import the numpy package.

Step 3: Search for a number or a set of numbers with where function.

Step 4: Sort the array with sort function.

Step 5: Set a filter array that sets TRUE for the values that satisfy the condition.

Step 6: Filter and print the array.

Step 7: Stop the program.

PROGRAM:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 4, 4])
x = np.where(arr == 4)
print( "Number 4 is at :", x)
```

```
even = np.where(arr%2 == 0)
print( "Even number at : ", even)
sorted = np.sort(arr)
print("Sorted array is : ", sorted)
filter_arr = []
for element in arr:
  if element > 3:
    filter_arr.append(True)
  else:
    filter_arr.append(False)
newarr = arr[filter_arr]
print("Filtered array is : ", newarr)
```

OUTPUT:

Number 4 is at : (array([3, 5, 6]),)

Even number at : (array([1, 3, 5, 6]),)

Sorted array is : [1 2 3 4 4 4 5]

Filtered array is : [4 5 4 4]

ii)SCIPY:

SciPy is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, statistics and signal processing.

ALGORITHM:

Step 1: Start the program.

Step 2: Import the python program constants from scipy package.

Step 3: Calculate the area of circle using the constants.pi variable.

Step 4: Print different constants available in constants.

Step 5: Stop the program.

PROGRAM:

```
from scipy import constants
```

```
r = 30
area = constants.pi * r * r
print( "Area : ", area )
print(constants.nautical_mile)
print(constants.angstrom)
print(constants.speed_of_sound)
```

OUTPUT:

```
Area :  2827.4333882308138
1852.0
1e-10
340.5
```

iii)PANDAS:

Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including finance, economics, Statistics, analytics, etc.

ALGORITHM:

Step 1: Start the program.

Step 2: Import the python pandas package.

Step 3: Read the document of table format with read_csv function.

Step 4: Print different parts of csv document with head and tail functions.

Step 5: Drop duplicates and print the csv document.

Step 6: Stop the program.

PROGRAM:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.head(10))
```

```
print(df.tail())
df.drop_duplicates(inplace = True)
#inplace=True ensures no new DataFrame, but removes duplicates from original DataFrame.
print(df.head())
```

OUTPUT:

```
Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
2        60    117       145     479.0
…
7        45    104       134     253.3
8        30    109       133     195.1
9        60     98       124     269.0

Duration  Pulse  Maxpulse  Calories
164       60    105       140     290.8
165       60    110       145     300.4
166       60    115       145     310.2
167       75    120       150     320.4
168       75    125       150     330.4

Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
2        45    109       175     282.4
3        45    117       148     406.0
4        60    102       127     300.5
```

RESULT:

Thus the program to implement use written modules and Python Standard Libraries (Pandas, Numpy, Matplotlib, Scipy) was executed successfully.

EX.No: 9 **DATE:**	**WORD COUNT, LONGEST WORD AND FILE COPY OPERATIONS IN FILE**

AIM:

To find the most frequent words in a text read from a file using python.

ALGORITHM:

Step1: Start the program.

Step 2: Read the file name.

Step 3: Open the file.

Step 4: Read each line from the file to count the words.

Step 5: Split each line in to words and count them.

Step 6: Print the words and count.

Step 7: Stop the program.

PROGRAM:

```
filename=raw_input("Enter file name:")
inf=open(filename,'r')
wordcount={}
for word in inf.split( ):
    if word not in wordcount:
        wordcount[word]=1
    else:
        wordcount[word]+=1
for key in wordcount.keys():
    print("%s\n%s\n"%(key,wordcount[key]))
wordlist = inf.split( )
s = sorted(wordlist, key = len)
print(“Largest string : “, s[-1])
paragraphcount=1
linecount=1
for i in inf.split( ):
```

```
        if '\n' in i:
                linecount+=1

        if len(i)<2:
                paragraphcount=0
        elif len(i)>2:
                paragraphcount = paragraphcount + 1
print( "%d\n%d\n%s"%( paragraphcount, linecount, inf ) )
inf.close()
```

OUTPUT:

```
Enter file name:'hai.txt'
hai
2
k
1
how
1
r
1
u
1
Largest string : Hai
1
1
hai k how r u hai
```

COPY FILE WITH SHUTIL:

```
from shutil import copyfile
sourcefile=input("Enter the source file name:")
destfile=input("Enter the destination file name:")
```

```
copyfile(sourcefile,destfile)
print("File copied successfully!")
c=open(destfile,'r')
print(c.read())
c.close()
```

OUTPUT:

```
Enter the source file name: file1.txt
Enter the destination file name: file2.txt
File copied successfully!
Hai, how are you?
```

COPY FILE WITHOUT SHUTIL:

```
source = open("e:\\aruna\\file1.txt", "r")
dest = open("e:\\aruna\\file2.txt", "w")
line = source.readline()
while line:
   dest.write(line)
   line = source.readline()
print("File copied successfully!")
source.close()
dest.close()
```

OUTPUT:

```
File copied successfully!
```

LONGEST WORD IN A LINE:

ALGORITHM:

Step1: Start the program.

Step 2: Read the sentence.

Step 3: Split each line in to words to find the longest word in the sentence.

Step 6: Print the longest word and its length.

Step 7: Stop the program.

PROGRAM:

```
sentence=input("Enter Sentence:")
longest = max(sentence.split(),key=len)
print("Longest word is:",longest)
print("Its length is:",len(longest))
```

OUTPUT:

```
Enter Sentence: Tongue tied and twisted just an earth bound misfit I
Longest word is:twisted
Its length is:7
```

RESULT:

Thus the program to find the most frequent words in a text read from a file using python and file copy were executed successfully.

EX.No: 10 DATE:	REAL-TIME/TECHNICAL APPLICATION FOR EXCEPTION HANDLING

AIM:

To write a python program to implement a real time/technical application using exception handling

ALGORITHM:

1. Start
2. Read the values from the user.
3. Run the try block code.
4. If there is an exception then except block code is executed.
5. If no exception then the remaining code is executed.
6. Stop

PROGRAM- DIVIDE BY ZERO ERROR:

```
try:
        num1 = int(input("Enter First Number:"))
        num2 = int(input("Enter Second Number:"))
        result = num1/num2
        print(result)
except ValueError as e:
        print("Invalid Input")
except ZeroDivisionError as e:
        print(e)
```

OUTPUT:

```
Enter First Number:432.12
Invalid Input

Enter First Number:43
Enter Second Number:0
Division by zero
Enter First Number:2
```

Enter Second Number:2

1

PROGRAM – VOTER'S AGE VALIDATION:

```
validVoter = False
while not validVoter:
        try:
                age = int( input( "Please enter your age : " ) )
                if age >= 18 and age <= 100:  validVoter = True
                else:     print( "Not eligible to vote" )
        except:
                print("Enter a valid age between 1 and 100")
print("You are eligible to vote")
```

OUTPUT:

Please enter your age : -20

Not eligible to vote

Please enter your age : Twenty

Enter a valid age between 1 and 100

Please enter your age : 20

You are eligible to vote

PROGRAM – MARKS RANGE VALIDATION:

```
mark = -1
while (mark <0 or mark > 100):
   value = input("Enter a valid mark : ")
   if value.isdigit():
      data = int(value)
      break
print( "Valid mark is ", value)
```

OUTPUT:

Enter a valid mark : Ninety Eight

Enter a valid mark : 98.00

Enter a valid mark : 98

Valid mark is 98.

RESULT:

Thus the python program for exception handling was executed and verified successfully.

EX.No: 11 **DATE:**	**EXPLORING PYGAME TOOL**

AIM:

To explore Pygame tool.

PYGAME:

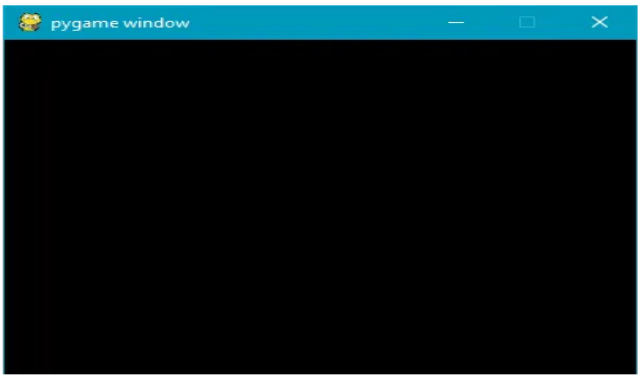
PyGame in a python library to develop games. Before starting with pygame we have to install pygame and have some programming knowledge. As we know, to play any game we first need a window. So we will first create a window in the PyGame.

PROGRAM:

(i) Create Window using Pygame

```
import pygame
pygame.init()
pygame.display.set_caption("pygame window")
window = pygame.display.set_mode((400, 300))
done = False
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    pygame.display.flip()
pygame.quit()
```

OUTPUT:



Functionality of above code:

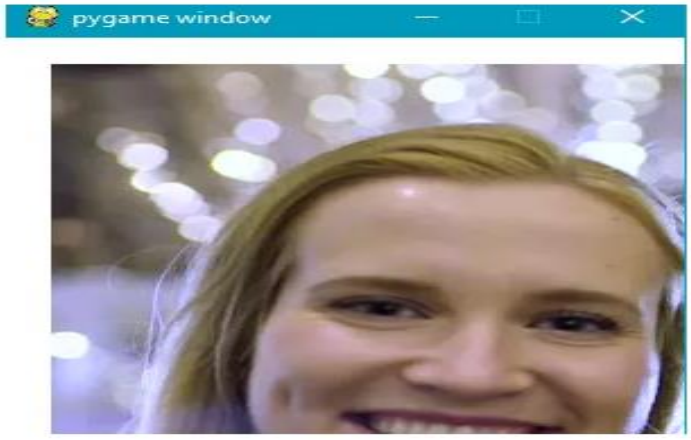
1. First we imported Pygame library

2. Initializing all modules of pygame (**pygame.init()**)
3. **pygame.display.set_caption ,** used to display name of the window.
4. After that **pygame.display.set_mode(),** is a function in pygame to create window.
5. The loop is placed to display our windows on the screen. (If we do not use loop, our window will be created but will not displayed for long time).
6. Inside the event, we handle all the actions that occur above the window.
7. If we quit the window **(event.type == pygame.QUIT**) means we close the window.
8. **pygame.display.flip(),** allows to update a portion of the screen, instead of the entire area.

(ii) Load image using Pygame

```
import pygame
pygame.init()
        window = pygame.display.set_mode((500, 500))
        done = False
        clock = pygame.time.Clock()
        while not done:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    done = True
            screen.fill((255, 255, 255))
            image = pygame.image.load(r'hhh.jpg')
            window.blit(image, (20, 20))
            pygame.display.flip()
            clock.tick(60)
        pygame.quit()
```

OUTPUT:



Functionality of above code:

1. What's new here is, **pygame.image.load() function,** which is used to load image.
2. And next is **window.blit(),** this function draw image onto the window at position (x,y).
3. As you can see from the output, our image is larger than our window size, this means that we should not take the image size smaller than the size of window.

(iii)Font Style and size in Pygame

```
import pygame
pygame.init()
window = pygame.display.set_mode((300, 300))
clock = pygame.time.Clock()
done = False
font = pygame.font.SysFont(None,50)
text = font.render("Goeduhub", True, (255, 0, 0))
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    window.fill((255, 255, 255))
    window.blit(text, (150 - text.get_width() // 2, 140 - text.get_height() // 2))
    pygame.display.flip()
    clock.tick(60)
pygame.quit()
```

OUTPUT:

functionality of above code:

1. **The pygame.font.SysFont(text_style,text_size) ,**function is used to describe the style of a text and the size of the text. None as style means, it will take default system text style.
2. **The font.render (text,True , color)-** Create a Text surface object to draw Text on it.
3. **window.fill(color),** it just fill our window with color which we defined in it , by default our window is black , here we filled it by white color

RESULT:

Thus the Pygame tool was executed successfully.

EX.No: 12 **DATE:**	**DEVELOP A GAME ACTIVITY USING PYGAME**

AIM:

To write a Python program to simulate bouncing ball in Pygame.

ALGORITHM:

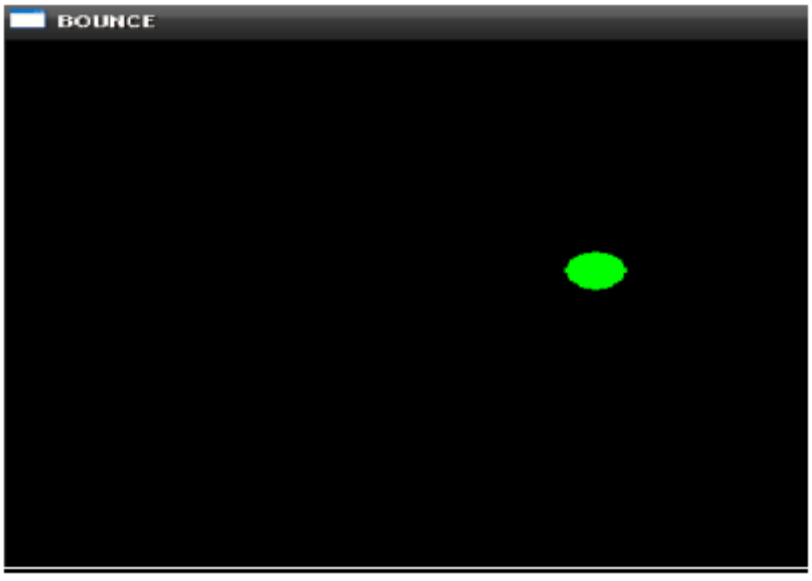
1. Start.
2. Import the required packages
3. Set up the colors for the bouncing ball.
4. Define the parameters to simulate bouncing ball.
5. Display the created simulation.
6. Stop.

PROGRAM:

```
import pygame, sys, time,random
from pygame.locals import*
from time import*
pygame.init()
windowSurface=pygame.display.set_mode((500,400),0,32)
pygame.display.set_caption("BOUNCE")
BLACK=(0,0,0)
WHITE=(255,255,255)
RED=(255,0,0)
GREEN=(0,255,0)
BLUE=(0,0,255)
info=pygame.display.Info()
sw=info.current_w
sh=info.current_h
y=0
direction=1
while True:
```

```
windowSurface.fill(BLACK)
pygame.draw.circle(windowSurface,GREEN,(250,y),13,0)
sleep(.006)
y+=direction
if y>=sh:
    direction=-1
elif y<=0:
    direction=1
pygame.display.update()
for event in pygame.event.get():
    if event.type==QUIT:
        pygame.quit()
        sys.exit()
```

OUTPUT:



RESULT:

Thus the Python program to simulate bouncing ball in Pygame was executed and verified successfully.