SYLLABUS

GE8161 PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY **LT P C**

0042

COURSE OBJECTIVES:

- To write, test, and debug simple Python programs.
- To implement Python programs with conditionals and loops.
- Use functions for structuring Python programs.
- Represent compound data using Python lists, tuples, dictionaries.
- Read and write data from/to files in Python.

LIST OF PROGRAMS

1. Compute the GCD of two numbers.
2. Find the square root of a number (Newton's method)
3. Exponentiation (power of a number)
4. Find the maximum of a list of numbers
5. Linear search and Binary search
6. Selection sort, Insertion sort
7. Merge sort
8. First n prime numbers
9. Multiply matrices
10. Programs that take command line arguments (word count)
11. Find the most frequent words in a text read from a file
12. Simulate elliptical orbits in Pygame
13. Simulate bouncing ball using Pygame

PLATFORM NEEDED

Python 3 interpreter for Windows/Linux

TOTAL: 60 PERIODS

INDEX

Ex. No: 1

COMPUTE THE GCD OF TWO NUMBERS

AIM:

To write a python program to compute the GCD of two numbers.

ALGORITHM :

Step 1: Start

Step 2: read two numbers to find the GCD n1,n2.

Step 3: rem=d1%d2

Step 4: while rem!=0

d1=d2

d2=rem

Rem=d1%d2

Step 5: print GCD is d2.

Step 6: Stop

PROGRAM/SOURCE CODE :

```
d1=int(raw_input("Enter a number:"))
d2=int(raw_input("Enter another number"))
rem=d1%d2
while rem!=0 :
   d1=d2
   d2=rem
   rem=d1%d2
```

```
print "gcd of given numbers is : %d" %(d2)
```

OUTPUT :

```
Enter a number :54
Enter another number :24
GCD of given number is: 6
```

RESULT:

Thus the program to find the GCD of two numbers is executed and the output is obtained.

Ex. No: 2

FIND THE SQUARE ROOT OF A NUMBER (NEWTON'S METHOD)

AIM:

To write a python program to find the square root of a number (Newton's method)

ALGORITHM :

Step 1: Define a function for Newton square root with two arguments.

Step 2: Assign the approximate value = 0.5*n.

Step 3: In each iteration, decide the range.

Step 4: Then calculate the approximate value.

Step 5: Return the approximate value.

Step 6: Finally print the values.

PROGRAM/SOURCE CODE :

```
def newtonSqrt(n, howmany):
    approx = 0.5 * n
    for i in range(howmany):
        betterapprox = 0.5 * (approx + n/approx)
        approx = betterapprox
    return betterapprox

print("Newton Sqrt  Value is =".newtonSqrt(10, 3))
print("Newton Sqrt  Value is =".newtonSqrt(10, 5))
print("Newton Sqrt  Value is =".newtonSqrt(10, 10))
```

OUTPUT :

```
Newton Sqrt Value is =.3.16231942215
Newton Sqrt Value is .=3.16227766017
Newton Sqrt Value is .=3.16227766017
```

RESULT:

Thus the program to find the square root (Newton's method) is executed and the output is obtained.

Ex. No: 3

EXPONENTIATION (POWER OF A NUMBER)

AIM:

To write a python program to find the exponentiation of a number.

ALGORITHM :

Step 1: Start.

Step 2: read base value

Step 3: Read exponent value.

Step 4: if base value is equal to one return base

Step 5: if base value is not equal to one return .

return(base*power(base,exp-1))

Step 6: print the result of program.

Step 7: Stop.

PROGRAM/SOURCE CODE:

```
def power(base,exp):
   if(exp==1):
      return(base)
   if(exp!=1):
      return(base*power(base,exp-1))
base=int(input("Enter base: "))
exp=int(input("Enter exponential value: "))
print("Result:",power(base,exp))
```

OUTPUT :

Enter the base:3

Enter exponential value:2

Result: 9

RESULT:

Thus the program to find the exponentiation of a number is executed and the output is obtained.

Ex. No: 4

FIND THE MAXIMUM OF A LIST OF NUMBERS

AIM:

To write a python program to find the maximum of a list of numbers.

ALGORITHM :

Step 1: Start.

Step 2:Read the number of element in the list.

Step 3: Read the number until loop n-1.

Step 4: Then Append the all element in list

Step 5:Goto STEP-3 upto n-1.

Step 6: Sort the listed values.

Step 7:Print the a[n-1] value.

PROGRAM/SOURCE CODE :

```
a=[]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
   b=int(input("Enter element:"))
   a.append(b)
a.sort()
print("Largest element is:",a[n-1])
```

OUTPUT :

```
Enter number of elements:5
Enter element: 3
Enter element:2
Enter element:1
Enter element:5
```

Enter element:4

Largest element is:5

RESULT:

Thus the program to find the Maximum of a List of numbers is executed and the output is obtained.

Ex. No: 5a

LINEAR SEARCH

AIM:

To write a python program to perform the linear search.

ALGORITHM :

Step 1: Start

Step 2: Read the element in list.

Step 3: Read the searching element from the user

Step 4: Assign to FALSE flag value

Step 5: Search the element with using for loop until length of list

Step 6: if value is found assign the flag value is true

Step7:Then print the output of founded value and position.

Step8:if value is not found then go to next step

Step9:print the not found statement

PROGRAM/SOURCE CODE :

```
list_of_elements = [14, 20, 58, 90, 03, 17]
x = int(input("Enter number to search: "))
found = False
for i in range(len(list_of_elements)):
if(list_of_elements[i] == x):
found = True
print("%d found at %dth position"%(x,i))
break
if(found == False):
print("%d is not in list"%x)
```

OUTPUT :

Enter number to search: 90

found at 4th position

RESULT:

Thus the program to perform linear Search is executed and the output is obtained.

Ex. No: 5b

BINARY SEARCH

AIM:

To write a python program to perform the binary search.

ALGORITHM :

Binary_search [arr, starting index, last index, element]

Step:1- mid = (starting index + last index) / 2

Step:2- If starting index > last index

Then, Print "Element not found"

Exit

Else if element > arr[mid]

Then, starting index = mid + 1

Go to Step:1

Else if element < arr[mid]

Then, last index = mid - 1

Go to Step:2

Else:

{ means element == arr[mid] }

Print "Element Presented at position" + mid

Exit

PROGRAM/SOURCE CODE :

```
def Binary_search(arr,start_index,last_index,element):
   while (start_index<= last_index):
      mid =(int)(start_index+last_index)/2
      if (element>arr[mid]):
         start_index = mid+1
      elif (element<arr[mid]):
         last_index = mid-1
      elif (element == arr[mid]):
         return mid
   return -1

arr = [2,14,19,21,99,210,512,1028,4443,5110]
element = 4443
start_index = 0
last_index = len(arr)-1
found = Binary_search(arr,start_index,last_index,element)
if (found == -1):
   print "element not present in array"
else:
   print "element is present at index " + str(found)
```

OUTPUT:

element is present at index 8

RESULT:

Thus the program to perform Binary Search is executed and the output is obtained.

Ex. No: 6a

SELECTION SORT

AIM:

To write a python program to perform selection sort.

ALGORITHM:

Step 1: Read the number of elements for the list from the user.

Step 2: Using for loop insert the elements in the list.

Step 3: Initialize the minimum element as min=numbers[i].

Step 4: Using the swap method the elements are sorted accordingly.

Step 5: Print the sorted list.

PROGRAM/SOURCE CODE:

```
def selectionSort(nlist):
  for fillslot in range(len(nlist)-1,0,-1):
     maxpos=0
     for location in range(1,fillslot+1):
        if nlist[location]>nlist[maxpos]:
           maxpos = location
     temp = nlist[fillslot]
     nlist[fillslot] = nlist[maxpos]
     nlist[maxpos] = temp
nlist = [14,46,43,27,57,41,45,21,70]
selectionSort(nlist)
print(nlist)
```

OUTPUT:

```
[14, 21, 27, 41, 43, 45, 46, 57, 70]
```

RESULT:

Thus the program to perform Selection Sort is executed and the output is obtained.

Ex. No: 6b

INSERTION SORT

AIM:

To write a python program to perform insertion sort.

ALGORITHM:

Step 1: Read the number of elements for the list from the user.

Step 2: Define the function for insertion Sort

Step 3: Then initialize the loop as follows.

For i in range (1, len(alist)

Step 4: Using While loop check the condition

Position > 0 and alist[position-1]>currentvalue

Step 5: If the condition is true swap the values by changing the position.

Step 6: Print the sorted list.

PROGRAM/SOURCE CODE:

```
def insertionSort(alist):
  for index in range(1,len(alist)):
    currentvalue = alist[index]
    position = index
```

```
    while position>0 and alist[position-1]>currentvalue:
        alist[position]=alist[position-1]
        position = position-1

    alist[position]=currentvalue

alist = [54,26,93,17,77,31,44,55,20]
insertionSort(alist)
print(alist)
```

OUTPUT :

```
17, 20, 26, 31, 44, 54, 55, 77, 93
```

RESULT:

Thus the program to perform Insertion Sort is executed and the output is obtained.

Ex. No: 7

MERGE SORT

AIM:

To write a python program to perform Merge sort.

ALGORITHM:

Step 1: Compute the function def mergeSort(alist)

Step 2: With in the if condition

if len(alist)>1:

mid = len(alist)//2

lefthalf=alist[:mid]

righthalf=alist[mid;]

Step 3: Then merge the left half and right half .

Step 4: Then initialize the condition for merge(right half)

Step 5: print the OUTPUT : in step by step execution

PROGRAM/SOURCE CODE:

```
def mergeSort(alist):
   print("Splitting ",alist)
   if len(alist)>1:
```

```
    mid = len(alist)//2
    lefthalf = alist[:mid]
    righthalf = alist[mid:]
    mergeSort(lefthalf)
    mergeSort(righthalf)
    i=0
    j=0
    k=0
    while i < len(lefthalf) and j < len(righthalf):
        if lefthalf[i] < righthalf[j]:
            alist[k]=lefthalf[i]
            i=i+1
        else:
            alist[k]=righthalf[j]
            j=j+1
        k=k+1
    while i < len(lefthalf):
        alist[k]=lefthalf[i]
        i=i+1
        k=k+1
    while j < len(righthalf):
        alist[k]=righthalf[j]
        j=j+1
        k=k+1
print("Merging ",alist)
```

```
alist = []
n=int(input("Enter n:"))
for i in range(0,n):
    alist.insert(i,int(input("Enter numbers[%d]: " %i)))
mergeSort(alist)
print(alist)
```

OUTPUT :

```
Enter n:6
Enter numbers[0]: 45
Enter numbers[1]: 12
Enter numbers[2]: 34
Enter numbers[3]: 6
Enter numbers[4]: 8
Enter numbers[5]: 1
Splitting [45, 12, 34, 6, 8, 1]
Splitting [45, 12, 34]
Splitting [45]
Merging [45]
Splitting [12, 34]
Splitting [12]
Merging [12]
Splitting [34]
Merging [34]
Merging [12, 34]
Merging [12, 34, 45]
```

Splitting [6, 8, 1]

Splitting [6]

Merging [6]

Splitting [8, 1]

Splitting [8]

Merging [8]

Splitting [1]

Merging [1]

Merging [1, 8]

Merging [1, 6, 8]

Merging [1, 6, 8, 12, 34, 45]

[1, 6, 8, 12, 34, 45]

RESULT:

Thus the program to perform Merge Sort is executed and the output is obtained.

Ex. No: 8

FIRST N PRIME NUMBERS

AIM:

To write a python program to find first n prime numbers.

ALGORITHM:

Step1: Take in the upper limit for the range and store it in a variable.
Step 2: Let the first for loop range from 2 to the upper limit.
Step3: Initialize the count variable to 0.
Step4: Let the second for loop range from 2 to half of the number (excluding 1 and the number itself).
Step 5: Then find the number of divisors using the if statement and increment the count variable each time.
Step 6: If the number of divisors is lesser than or equal to 0, the number is prime.
Step 7: Print the final result.

PROGRAM/SOURCE CODE:

```
i=1
x = int(input("Enter the number:"))

for k in range (1, (x+1), 1):
        c=0;
        for j in range (1, (i+1), 1):
                a = i%j
                if (a==0):
                        c = c+1

        if (c==2):
                print (i)
        else:
                k = k-1
```

```
    i=i+1
```

OUTPUT:

```
Enter the number: 15
2
3
4
5
7
11
13
```

RESULT:

Thus the program to find first n prime numbers is executed and the output is obtained.

Ex. No: 9

MULTIPLY MATRICES

AIM:

To write a python program to perform matrix multiplication.

ALGORITHM :

Step 1: Assume the values for the first matrix as X
Step 2: Assume the values for the first matrix as Y
Step 3: Assume the resultant matrix = [[0,0,0],
[0,0,0],
[0,0,0]]
Step 4: Then iterate through rows of X
Step 5: Then iterate through rows of Y
Step 6: Multiply and then Print the output

PROGRAM/SOURCE CODE :

```
# 3x3 matrix
X = [[12,7,3],
   [4 ,5,6],
   [7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],
[6,7,3,0],
[4,5,9,1]]
# result is 3x4
result = [[0,0,0,0],
        [0,0,0,0],
        [0,0,0,0]]

# iterate through rows of X
```

```
for i in range(len(X)):
  # iterate through columns of
  Y for j in range(len(Y[0])):
     # iterate through rows of
     Y for k in range(len(Y)):
        result[i][j] += X[i][k] * Y[k][j]

for r in result:
  print(r)
```

OUTPUT :

[114, 160, 60]

[74, 97, 73]

[119, 157, 112]

RESULT:

Thus the program to perform matrix Multiplication is executed and the output is obtained.

Ex. No: 10

PROGRAMS THAT TAKE COMMAND LINE ARGUMENTS (WORD COUNT)

AIM:

To write a python program to perform command line arguments (word count).

ALGORITHM :

Step 1: Start

Step 2: Find the length of CLA.

Step 3: Store the argument list to a variable namely cmdargs.

Step 4: Print the total number of arguments passed to the script.

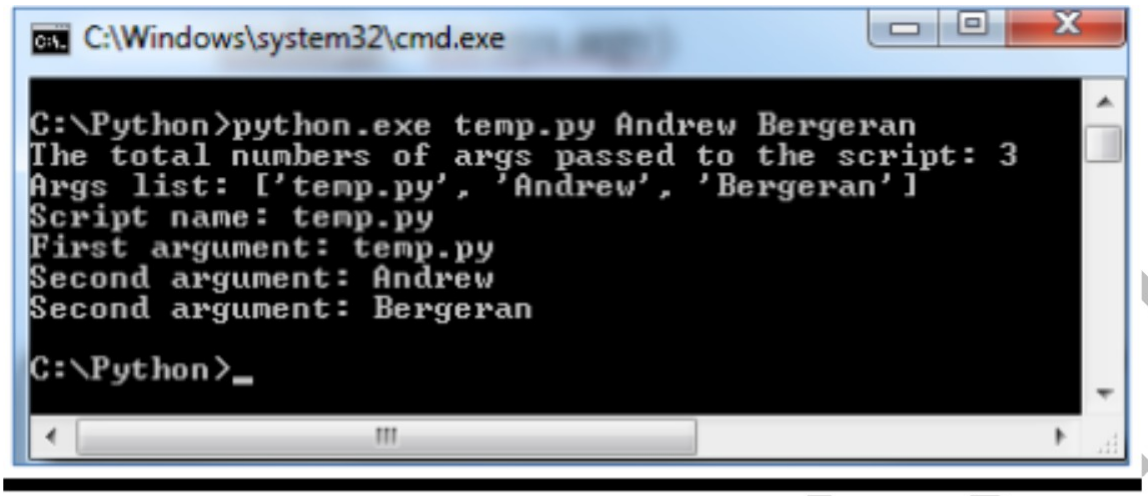
Step 5: Print the argument list separately.

Step 6: Stop

PROGRAM/SOURCE CODE :

```
import sys
total = len(sys.argv)
cmdargs = str(sys.argv)
print ("The total numbers of args passed to the script: %d " % total)
print ("Args list: %s " % cmdargs)
print ("Script name: %s" % str(sys.argv[0]))
print ("First argument: %s" % str(sys.argv[0]))
print ("Second argument: %s" % str(sys.argv[1]))
print ("Third argument: %s" % str(sys.argv[2]))
```

OUTPUT :



RESULT:

Thus the program to count the words is executed and the output is obtained.

Ex. No: 11

FIND THE MOST FREQUENT WORDS IN A TEXT READ FROM A FILE

AIM:

To write a python program to find the most frequent words from a file.

ALGORITHM :

Step 1: Create a file.

Step 2: Open the created file in read mode.

Step 3: Using for loop find the most frequent words.

Step 4: Assume the key for each of the words.

Step 5: Print the frequent words that are used in the file.

Step 6: Close the file and print the output .

PROGRAM/SOURCE CODE :

```
from string import punctuation
from operator import itemgetter
N=10
words = {}
words_gen = (word.strip(punctuation).lower() for line in open("test.txt")
for word in line.split())
for word in words_gen:
words [word] = words.get(word, 0) + 1
top_words = sorted(words.iteritems(), key=itemgetter(1), reverse=True)[:N]
for word, frequency in top_words:
print "%s: %d" % (word, frequency)
```

OUTPUT :

RESULT:

Thus the program to find the most frequent words in a text is executed and the output is obtained.

Ex. No: 12

SIMULATE ELLIPTICAL ORBITS IN PYGAME

AIM:

To write a python program to simulate elliptical orbits in pygame.

ALGORITHM :

Step 1: Import the necessary header files for the implementation of this pygame.

Step 2: Set the display mode for the screen using

screen=pygame.display.set_mode((700,700))

Step 3: Develop the balls with necessary colors.

white=(255,255,255)

blue=(0,0,255)

yellow=(255,255,0)

gray=(200,200,200)

black=(0,0,0)

Step 4: Set the radius for sun, moon and their orbit.

Step 5: Set the time for the pygame orbit clock=pygame.time.Clock()

Step 6: Update the earth position.

Step 7: Again update moon position based on earth position.

Step 8: Update the moon and earth angles.

Step 9: Reset the screen and draw the stars, sun, moon and the earth.

Step 10: Set the clock tick as (60) and exit.

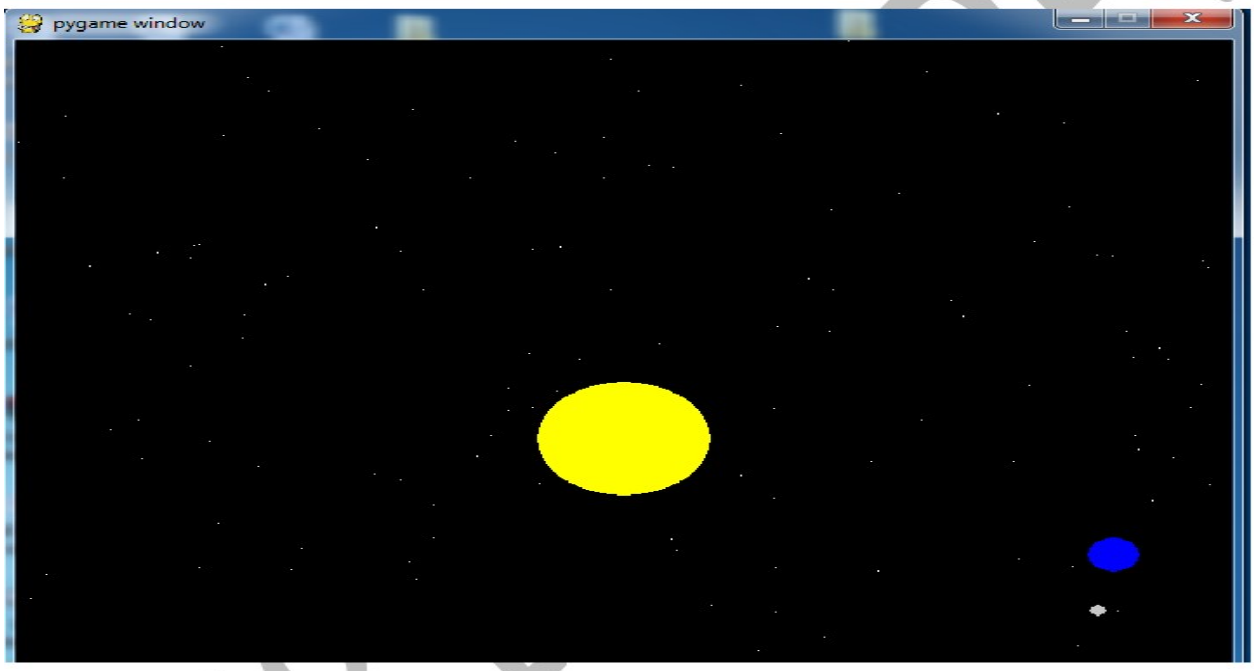
PROGRAM/SOURCE CODE :

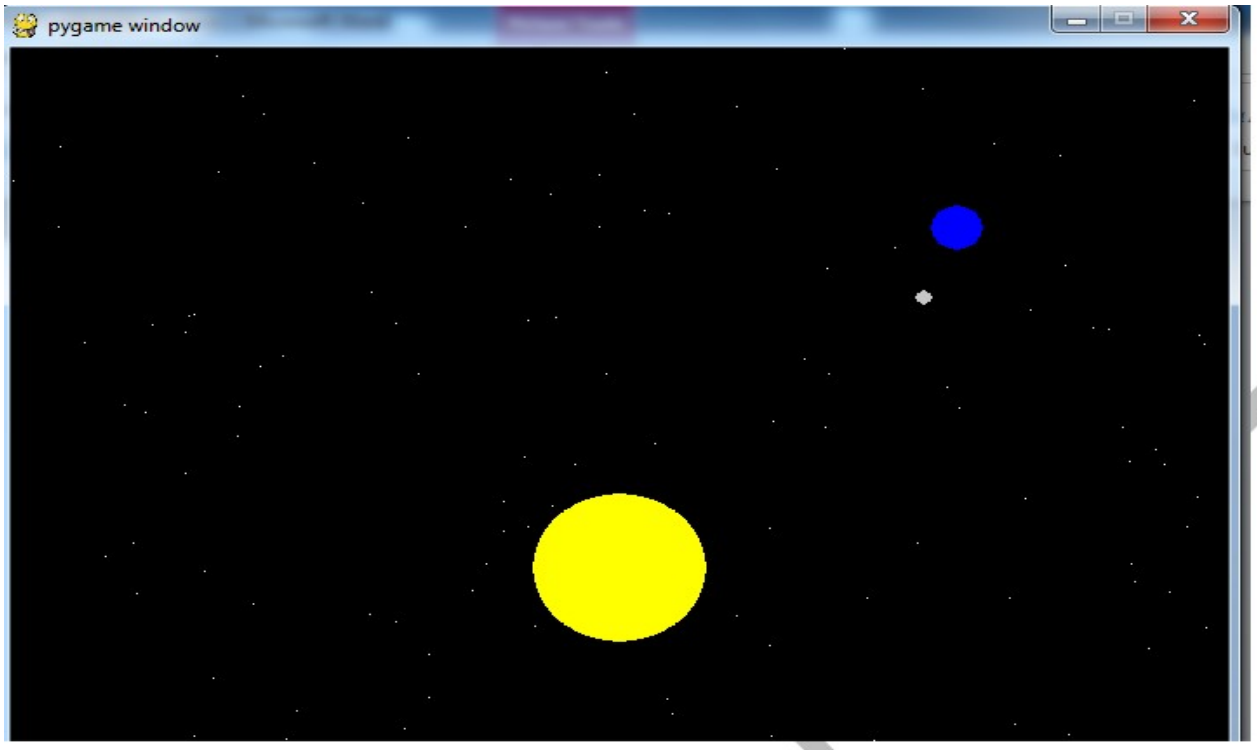
```
import pygame
import random
import math
pygame.init()
screen=pygame.display.set_mode((700,700))
white=(255,255,255)
```

```
blue=(0,0,255)
yellow=(255,255,0)
gray=(200,200,200)
black=(0,0,0)
sun_radius=50
center=(350,350)
earth_x=50
earth_y=350
earth_orbit=0
moon_orbit=0
clock=pygame.time.Clock()
running=True
stars=[(random.randint(0,699),random.randint(0,699)) for x in range(140)]
while running:
for event in pygame.event.get():
    if event.type==pygame.QUIT:
        running=False
        earth_x=math.cos(earth_orbit)*300+350
        earth_y=-math.sin(earth_orbit)*300+350
        moon_x=math.cos(moon_orbit)*50+earth_x
        moon_y=-math.sin(moon_orbit)*50+earth_y
        earth_orbit+=0.002
        moon_orbit+=0.01
        screen.fill(black)
for star in stars:
        x,y=star[0],star[1]
        pygame.draw.line(screen,white,(x,y),(x,y))
        pygame.draw.circle(screen,yellow,center,sun_radius)
        pygame.draw.circle(screen,blue,(int(earth_x),int(earth_y)),15)
        pygame.draw.circle(screen,gray,(int(moon_x),int(moon_y)),5)
        pygame.display.flip()
```

```
        clock.tick(60)
pygame.quit()
```

OUTPUT :





RESULT:

Thus the simulation of elliptical curve orbit using pygame is executed and the output is obtained.

Ex. No: 13

SIMULATE BOUNCING BALL USING PYGAME

AIM:

To write a python program to simulate bouncing ball using pygame.

ALGORITHM:

Step 1: Import the necessary files for the implementation of this Pygame.

Step 2: Set the display mode for the screen using

windowSurface=pygame.display.set_mode((500,400),0,32)

Step 3: Now set the display mode for the pygame to

bounce pygame.display.set_caption("Bounce")

Step 4: Develop the balls with necessary colors.

BLACK=(0,0,0)

WHITE=(255,255,255)

RED=(255,0,0)

GREEN=(0,255,0)

BLUE=(0,0,255)

Step 5: Set the display information info=pygame.display.Info()

Step 6: Set the initial direction as down.

Step 7: Change the direction from down to up.

Step 8: Then again change the direction from up to down.

Step 9: Set the condition for quit.

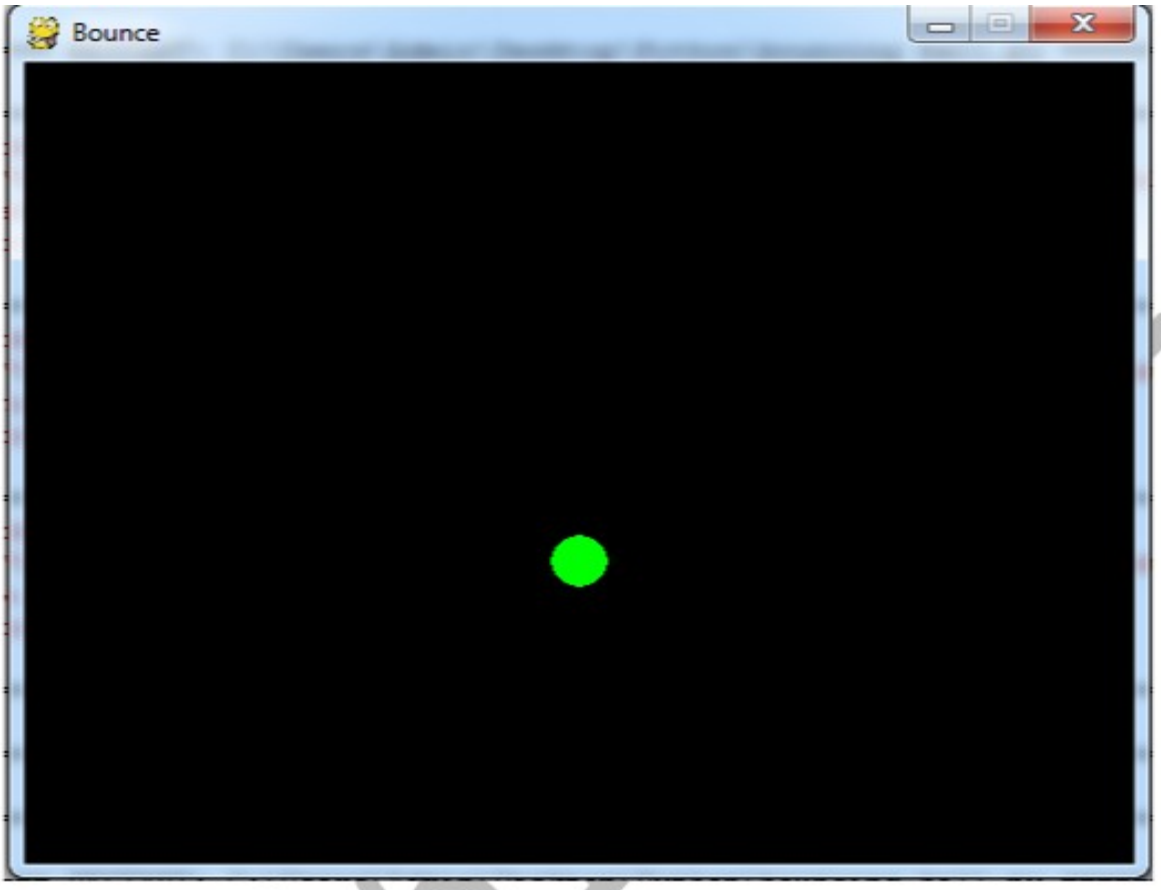
Step 10: Exit from the pygame.

PROGRAM/SOURCE CODE :

```
import pygame,sys,time
import random
frompygame.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(0.006)
   y+=direction
if y>=sh:
        direction=-1
elif y<=100:
        direction=1
        pygame.display.update()
for event in pygame.event.get():
        if event.type==QUIT:
        pygame.quit()
        sys.exit()
```

OUTPUT :



RESULT:

Thus the program to simulate bouncing ball using pygame is executed and the output is obtained.

TOPIC BEYOND SYLLABUS

A1.TOWER OF HANOI

AIM:

To write a python program for tower of Hanoi Scenario.

ALGORITHM:

Step 1: create a function as move tower and move disk.

Step 2: check the height and if it is greater than 1 do the following

Step 3: Move a tower of height-1 to an intermediate pole, using the final pole.

Step 4: Move the remaining disk to the final pole.

Step 5: Move the tower of height-1 from the intermediate pole to the final pole using the original pole.

Step 6: Display the result.

PROGRAM /SOURCE CODE:

```
def moveTower(height,fromPole, toPole, withPole):
   if height >= 1:
      moveTower(height-1,fromPole,withPole,toPole)
      moveDisk(fromPole,toPole)
      moveTower(height-1,withPole,toPole,fromPole)
def moveDisk(fp,tp):
   print("moving disk from",fp,"to",tp)
moveTower(3,"A","B","C")
```

OUTPUT:

```
moving disk from A to B
moving disk from A to C
moving disk from B to C
```

moving disk from A to B

moving disk from C to A

moving disk from C to B

moving disk from A to B

RESULT:

Thus the program for Tower of Hanoi scenario is executed and the output is obtained.

A2.PROGRAM TO FIND GIVEN NUMBER IS ARMSTRONG NUMBER OR NOT

ALGORITHM

1. Start
2. Declare variables
3. Read the Input number.
4. Calculate sum of cubic of individual digits of the input.
5. Match the result with input number.
6. If match, Display the given number is Armstrong otherwise not.
7. Stop

SOURCE CODE

```
num = 1634

# Changed num variable to string,

# and calculated the length (number of

digits) order = len(str(num))

# initialize sum

sum = 0

# find the sum of the cube of each digit

temp = num

while temp > 0: digit =

  temp % 10 sum +=

  digit ** order

  temp //= 10
```

```
# display the result
if num == sum:
   print(num,"is an Armstrong number")
else:
   print(num,"is not an Armstrong number")
```

OUTPUT

1634 is an Armstrong number

A3.BUBBLE SORT ALGORITHM

ALGORITHM

1. Start
2. Declare variables and create an array
3. Read the Input for number of elements and each element.
4. Develop a function bubblesort to sort the array
5. Compare the elements in each pass till all the elements are sorted.
6. Display the output of the sorted elements .
7. Stop

SOURCE CODE

```
def shortBubbleSort(alist):
    exchanges = True
    passnum = len(alist)-1
    while passnum > 0 and exchanges:
      exchanges = False
      for i in range(passnum):
          if alist[i]>alist[i+1]:
              exchanges = True
              temp = alist[i]
              alist[i] = alist[i+1]
              alist[i+1] = temp
      passnum = passnum-1

alist=[20,30,40,90,50,60,70,80,100,110]
```

```
shortBubbleSort(alist)

print(alist)
```

OUTPUT

[20,30,40,50,60,70,80,90,100,110]

A4.PYTHON PROGRAM TO FIND THE SUM OF NATURAL NUMBERS UP TO N USING RECURSIVE FUNCTION

ALGORITHM

1. Start
2. Declare variables and initializations
3. Read the Input variable.
4. Define recursive expression for computational processing.
5. Return the output of the calculations.
6. Stop

SOURCE CODE

```
def  recur_sum(n):
  """Function to return the sum
  of natural numbers using recursion"""
  if n <= 1:
     return n
  else:
     return n + recur_sum(n-1)


# change this value for a different
result num = 16


# uncomment to take input from the user
#num = int(input("Enter a number: "))


if num < 0:
  print("Enter a positive number")
```

```
else:
  print("The sum is",recur_sum(num))
```

OUTPUT

The sum is 136

VIVA QUESTIONS

1. What is Python?
2. What is the purpose of PYTHONPATH environment variable?
3. How Python is interpreted?
4. Is python a case sensitive language?
5. What are the supported data types in Python?
6. What are the built-in type does python provides?
7. What is namespace in Python?
8. What are Python's dictionaries?
9. How will you create a dictionary in python?
10. What is Expression?
11. What is module and package in Python?
12. Mention what are the rules for local and global variables in Python?
13. How will you convert a single character to its integer value in python?
14. What is the purpose of ** operator?
15. What is the purpose is of is operator?
16. How will you randomize the items of a list in place?
17. How will you capitalizes first letter of string?
18. What is function?
19. What are the types of function?
20. What is return type?
21. Mention the use of the split function in Python?
22. Explain what is Flash & its benefits?
23. What is File?
24. How to Read the file?
25. How to copy the data from one file to another file?