SRM VALLIAMMAI ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF GENERAL ENGINEERING

LAB MANUAL

II SEMESTER

1901009 - PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY

Regulation – 2019

Academic Year 2022 – 2023 (EVEN Semester)

Dr.S.K.Saravanan, AP(Sel.G) / CSE	**Dr. S. Parthasarathy, AP(Sel.G) / CSE**
Ms. G. Sathya, AP(O.G) / CSE	

SYLLABUS

1901009 - PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY **L T P C**

0 0 4 2

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. How to create, slice, change, delete and index elements using Tuple.
8. Find First n prime numbers.
9. How to create, slice, change, add, delete and index elements using list.
10. Programs that take command line arguments (word count)
11. Write a program to reverse the string.
12. How to change, delete, add and remove elements in Dictionary.
13. Find the most frequent words in a text read from a file.
14. Simulate elliptical orbits in Pygame.
15. Simulate bouncing ball using Pygame.

PLATFORM NEEDED

Python 3 interpreter for Windows/Linux

TOTAL: 60 PERIODS

INDEX

S.No	Topic	Page No
1.	Compute the GCD of two numbers.	4
2.	Find the square root of a number (Newton's method)	6
3.	Exponentiation (power of a number)	8
4.	Find the maximum of a list of numbers	10
5.	Linear search and Binary search	12
6.	Selection sort, Insertion sort	17
7.	How to create, slice, change, delete and index elements using Tuple.	21
8.	First n prime numbers	23
9.	How to create, slice, change, add, delete and index elements using list.	25
10.	Program to calculate length of the string	28
11.	Write a program to reverse the string	29
12.	How to change, delete, add and remove elements in Dictionary	30
13.	Find the most frequent words in a text read from a file	32
14.	Simulate elliptical orbits in Pygame	34
15.	Simulate bouncing ball using Pygame	38
A.	Additional Exercises	41
B.	Viva Questions	54

Ex. No: 1

COMPUTE THE GCD OF TWO NUMBERS

AIM:

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

ALGORITHM :

Step1: Start

Step2:readtwonumberstofindtheGCDn1,n2.

Step3: rem=n1%n2

Step4: whilerem!=0

n1=n2

n2=rem

rem=n1%n2

Step5: print GCDisn2.

Step6: Stop

PROGRAM/SOURCE CODE :

```
n1=int(input("Enter a number:"))
n2=int(input("Enter another number"))
rem=n1%n2
while rem!=0 :
    n1=n2
    n2=rem
    rem=n1%n2
print ("gcd of given numbers is ",n2)
```

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: Go to 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 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: Go to STEP-3 upto n-1.

Step 6: Read the searching element from the user

Step 7: Assign to FALSE flag value

Step 8: Search the element with using for loop until length of list

Step 9: If value is found assign the flag value is true

Step10: Then print the output of founded value and position.

Step 11: If value is not found then go to next step

Step 12: Print the not found statement

PROGRAM/SOURCE CODE :

```
a=[ ]
n=int(input("Enter number of elements:"))
fori in range(1,n+1):
        b=int(input("Enter element:"))
        a.append(b)
```

```
x = int(input("Enter number to search: "))
found = False
for i in range(len(a)):
        if(a[i] = = x):
                found = True
                print("%d found at %dthposition"%(x,i))
                break
if (found==False):
        print("%d is not in list"%x)
```

OUTPUT 1:

```
Enter number of elements:5
Enter element:88
Enter element:11
Enter element:64
Enter element:23
Enter element:89
Enter number to search: 11
11 found at 1th position
```

OUTPUT 2:

```
Enter number of elements:5
Enter element:47
Enter element:99
Enter element:21
Enter element:35
Enter element:61
Enter number to search: 50
50 is not in list
```

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
        else:
            return -1
arr =[ ]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
    b=int(input("Enter element:")) # inputs must be in ascending order
    arr.append(b)
print(arr)
element = int(input("Enter the element to be searched"))
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", found)
```

OUTPUT 1:

Enter number of elements:8

Enter element:11

Enter element:33

Enter element:44

Enter element:56

Enter element:63

Enter element:77

Enter element:88

Enter element:90

[11, 33, 44, 56, 63, 77, 88, 90]

Enter the element to be searched 63

element is present at index 4

OUTPUT 2:

Enter number of elements:7

Enter element:11

Enter element:15

Enter element:20

Enter element:25

Enter element:30

Enter element:40

Enter element:50

[11, 15, 20, 25, 30, 40, 50]

Enter the element to be searched 22

element not present in array

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(lst, size):
        for i in range(size):
                min = i
                for j in range(i + 1, size):
                        iflst[j] <lst[min]:
                                min = j
                (lst[i], lst[min]) = (lst[min], lst[i])
data =[ ]
n=int(input("Enter number of elements:"))
fori in range(1,n+1):
        b=int(input("Enter element:")) # inputs must be in shuffled order
        data.append(b)
print("list before sorting:",data)
size = len(data)
selectionSort(data, size)
print('Sorted list in Ascending Order:')
print(data))
```

OUTPUT:

Enter number of elements:5

Enter element:77

Enter element:2

Enter element:53

Enter element:41

Enter element:25

list before sorting: [77, 2, 53, 41, 25]

Sorted list in Ascending Order:

[2, 25, 41, 53, 77]

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 = [ ]
n=int(input("Enter number of elements:"))
for i in range(1,n+1):
        b=int(input("Enter element:")) # inputs must be in ascending order
        alist.append(b)
print("list before sorting:",alist)
insertionSort(alist)
print("list after sorting:",alist)
```

OUTPUT :

```
Enter number of elements:6
Enter element:321
Enter element:123
Enter element:650
Enter element:189
Enter element:420
Enter element:250
list before sorting: [321, 123, 650, 189, 420, 250]
list after sorting: [123, 189, 250, 321, 420, 650]
```

RESULT:

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

Ex. No: 7

Tuple Operations and its Methods

AIM:

To write a python program to create, slice, change, delete and index elements using Tuple.

ALGORITHM :

Step 1: Create the tuple.

Step 2: Indexing the tuple using the index operator [ ].

Step 3: Silicingthetuple by using the slicing operator - colon ":"

Step 4: Changing the tuple.

Step 5: Deleting the tuple

PROGRAM/SOURCE CODE :

```
print("Tuple is created in the name: my_tuple")

my_tuple = ('p','e','r','m','i','t')

print("Tuple indexing",my_tuple[0])

print("Tuple negative indexing",my_tuple[-1])

print("Tuple slicing",my_tuple[1:4])

my_tuple = (4, 3, 2,5, [6, 5])

my_tuple[4][1] = 9

print("Tuple changing",my_tuple)

del my_tuple

print("Tuple Delete",my_tuple)
```

OUTPUT :

Tuple is created in the name: my_tuple

Tuple indexing p

Tuple negative indexing t

Tuple slicing ('e', 'r', 'm')

Tuple changing (4, 3, 2, 5, [6, 9])

RESULT:

Thus the program to create, slice, change, delete and index elements using Tuple and the output isobtained.

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 eachtime.

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 :

```
x = int (input ("Enter the number:"))
count = 1
n = 2
while ( count <= x ):
   for i in range (2, n):
      if ( n % i == 0):
         break
   else:
      print (n)
      count = count + 1
   n = n+1
```

OUTPUT :

Enter the number : 15

2
3
5
7
11
13
17
19
23
29
31
37
41
43
47

RESULT:

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

Ex. No: 9

List Operations and its Methods

AIM:

To write a python program to create, slice, change, delete and index elements using List.

ALGORITHM :

Step 1: Create the List.

Step 2: Indexing the List using the index operator [ ].

Step 3: Silicing an element from the List by using the slicing operator - colon ":"

Step 4: Changing an element from the List.

Step 5: Appending the List.

Step 6: Removing an element from the List.

Step 7: Deleting an element from the List

PROGRAM/SOURCE CODE :

```
print("List is created in the name: list")
list = ['p','e','r','m','i','t']
print("List Created",list)
print("List indexing",list[0])
print("List negative indexing",list[-1])
print("List slicing",list[1:4])
list = ['p','e','r','m','i','t']
print("Given list",list)
list[0]=2
print("List Changing",list)
list[1:4]=[1,2,3]
print("List Changing",list)
list = ['p','e','r','m','i','t']
```

```
print("Given list",list)

list.append(['add','sub'])

print("List appending",list)

list = ['p','e','r','m','i','t']

print("Given list",list)

list.remove('p')

print("List Removing",list)

list = ['p','e','r','m','i','t']

print("Given list",list)

list[2:5] = []

print("List Delete",list)
```

OUTPUT :

List is created in the name: list

List Created ['p', 'e', 'r', 'm', 'i', 't']

List indexing p

List negative indexing t

List slicing ['e', 'r', 'm']

Given list ['p', 'e', 'r', 'm', 'i', 't']

List Changing [2, 'e', 'r', 'm', 'i', 't']

List Changing [2, 1, 2, 3, 'i', 't']

Given list ['p', 'e', 'r', 'm', 'i', 't']

List appending ['p', 'e', 'r', 'm', 'i', 't', ['add', 'sub']]

Given list ['p', 'e', 'r', 'm', 'i', 't']

List Removing ['e', 'r', 'm', 'i', 't']

Given list ['p', 'e', 'r', 'm', 'i', 't']

List Delete ['p', 'e', 't']

RESULT:

Thus program to create, slice, change, delete and index elements using List is executed ant theoutput is obtained.

Ex. No: 10

PROGRAM TO CALCULATE LENGTH OF THE STRING

AIM:

To write a python program to calculate length of the string.

ALGORITHM :

Step 1: Start

Step 2: Read input string.

Step 3: Count number of characters in string including spaces.

Step 4: Display the answer.

Step 5: Stop

PROGRAM/SOURCE CODE :

```
defLenOfStr(s):
   count=0
   for i in s:
      count=count+1
   return count

s = input ("Enter a string:")
print ("The number of character in original string is : ",LenOfStr(s))
```

OUTPUT :

```
Enter a string: SRM VALLIAMMAI
The number of character in original string is : 14
```

RESULT:

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

Ex. No: 11

REVERSE THE STRING

AIM:

To write a python program to reverse a string.

ALGORITHM :

Step 1: Define a function reverse(s)

Step 2: Read a string

Step 3: Print the original string

Step 4: Print the reversed string

PROGRAM/SOURCE CODE :

```
def reverse(s):
  str = " "
  for i in s:
        str = i + str
  return str

s=input ("Enter a string:")
print ("The original string is : ",end="")
print (s)
print ("The reversed string(using loops) is : ",end="")
print (reverse(s))
```

OUTPUT :

```
Enter a string:PYTHON PROGRAMMING
The original string is : PYTHON PROGRAMMING
The reversed string(using loops) is : GNIMMARGORP NOHTYP
```

RESULT:

Thus the program to reverse a string is executed and the output is obtained.

Ex. No: 12

DICTIONARY Operations and its Methods

AIM:

To write a python program to to change, delete, add and remove elements in Dictionary

ALGORITHM :

Step 1: Create the Dictionary.

Step 2: Change an element to dictionary.

Step 3: Add an element to dictionary.

Step 4: Remove an element to dictionary

Step 5: Delete an element to dictionary

PROGRAM/SOURCE CODE :

```
my_dict = {1:1, 2:4, 3:9, 4:16, 5:20}

print("New Dictionary is created in the name: my_dict",my_dict)

my_dict[5] = 25

print("Change an element in Dictionary",my_dict)

my_dict[6] = 36

print("Add an element in Dictionary",my_dict)

print("Remove the arbitary element from the dictionary",my_dict.pop(5))

print("Remove the using .pop (5) element from the dictionary",my_dict)

delmy_dict[6]

print("Delete the  element from the dictionary",my_dict)
```

OUTPUT :

New Dictionary is created in the name: my_dict {1: 1, 2: 4, 3: 9, 4: 16, 5: 20}

Change an element in Dictionary {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

Add an element in Dictionary {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}

Remove the arbitary element from the dictionary 25

Remove the using .pop (5) element from the dictionary {1: 1, 2: 4, 3: 9, 4: 16, 6: 36}

Delete the element from the dictionary {1: 1, 2: 4, 3: 9, 4: 16}

RESULT:

Thus program to change, delete, add and remove elements in Dictionary is executed and the output is obtained.

Ex. No: 13

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 :

```
handle=open('sample.txt','w')
handle.write("hi hello hello how are you")
handle.close()
name=input ("Enter file name:")
handle = open(name, 'r')
text = handle.read()
words = text.split()
counts = dict()
for word in words:
        counts[word] = counts.get(word,0) + 1
 bigcount = None
 bigword = None
 for word,count in counts.items():
     if bigcount is None or count >bigcount:
         bigword = word
```

```
        bigcount = count
print(bigword, bigcount)
```

OUTPUT :

```
Enter file name: sample.txt
hello 2
```

RESULT:

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

Ex. No: 14

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 :

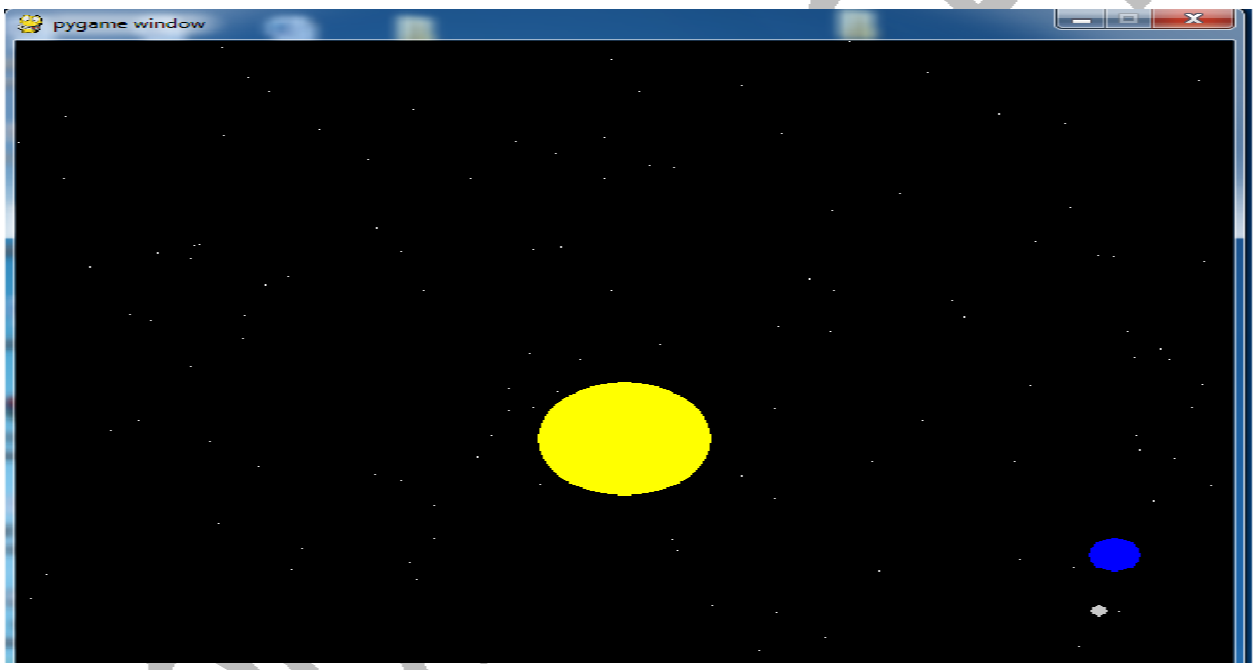
```
importpygame
import random
import math
pygame.init()
screen=pygame.display.set_mode((700,700))
white=(255,255,255)
blue=(0,0,255)
yellow=(255,255,0)
gray=(200,200,200)
black=(0,0,0)
sun_radius=50
center=(350,350)
earth_x=50
earth_y=350
earth_orbit=0
moon_orbit=0
clock=pygame.time.Clock()
running=True
stars=[(random.randint(0,699),random.randint(0,699)) for x in range(140)]
while running:
 for event in pygame.event.get():
    ifevent.type==pygame.QUIT:
       running=False
       earth_x=math.cos(earth_orbit)*300+350
       earth_y=-math.sin(earth_orbit)*300+350
```

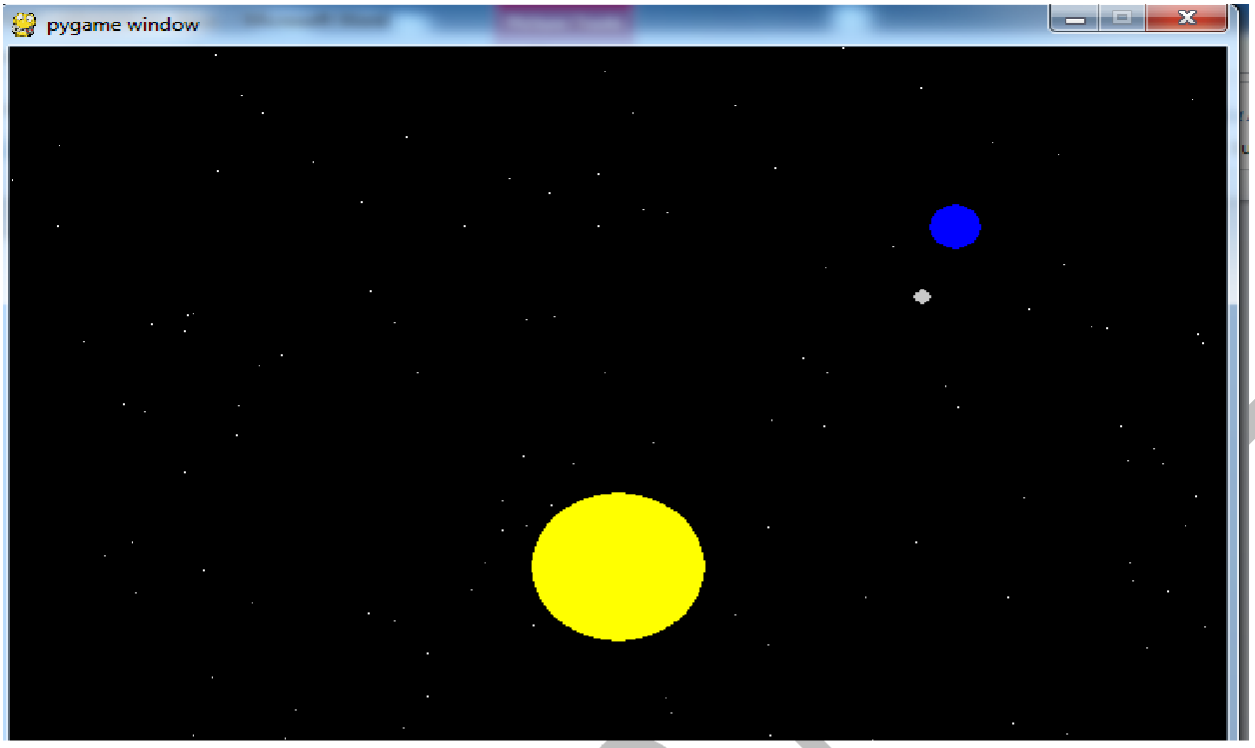
```
        moon_x=math.cos(moon_orbit)*50+earth_x
        moon_y=-math.sin(moon_orbit)*50+earth_y
        earth_orbit+=0.002
        moon_orbit+=0.01
        screen.fill(black)

    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 pygameis executed and the output is obtained.

Ex. No: 15

SIMULATE BOUNCING BALL USING PYGAME

AIM:

To write a python program to simulate bouncing ball usingpygame.

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
bouncepygame.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.

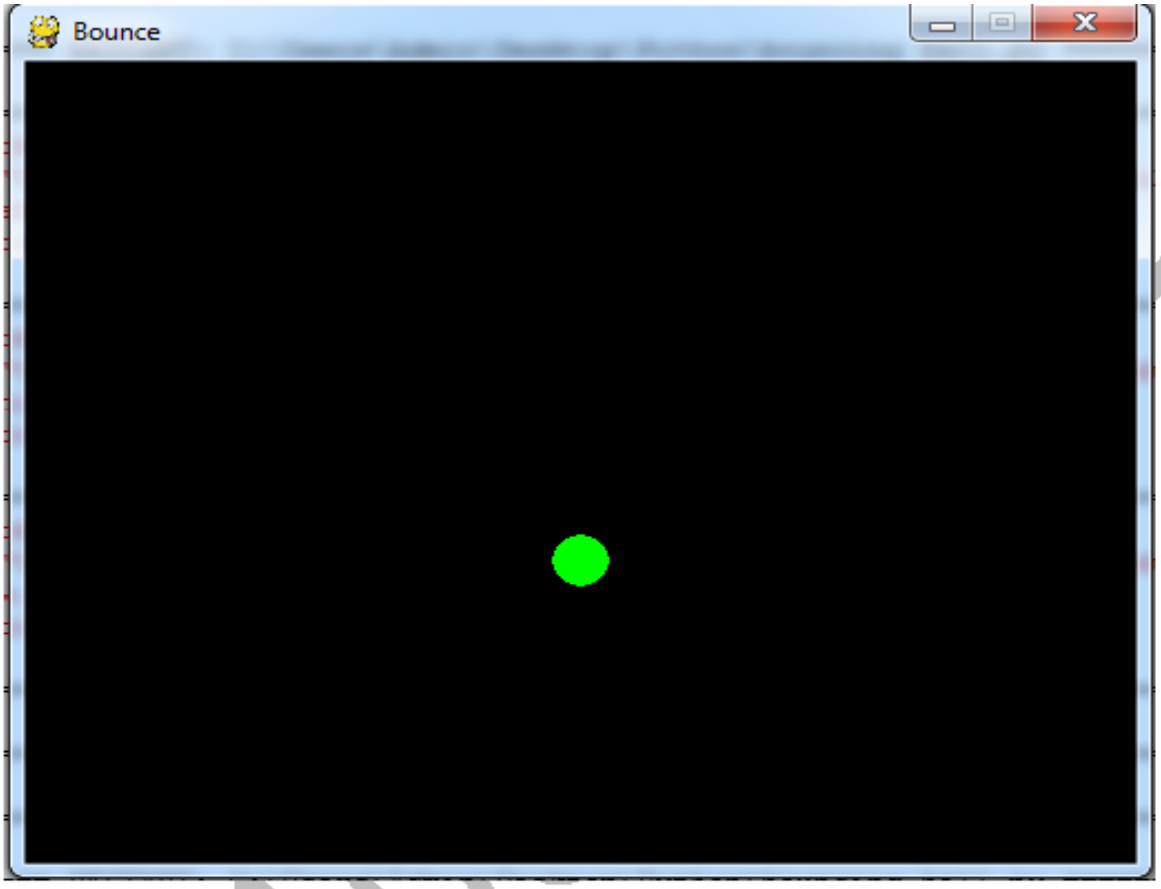
Step 10: Exit from the pygame.

PROGRAM/SOURCE CODE :

```
importpygame,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():
        ifevent.type==QUIT:
            pygame.quit()
    sys.exit()
```

OUTPUT :



RESULT:

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

A.Additional Exercises

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)
defmoveDisk(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

AIM:

To write a program to find given number is Armstrong 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
```

```
ifnum == sum:
    print(num,"is an Armstrong number")
else:
    print(num,"is not an Armstrong number")
```

OUTPUT

1634 is an Armstrong number

Result:

Thus the program to find the Armstrong number is executed successfully.

A3.BUBBLE SORT ALGORITHM

AIM:

To write a program on 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):
            ifalist[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]
```

Result:

Thus the bubble sort algorithm is being executed successfully.

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):
   if n <= 1:
      return n
   else:
      return n + recur_sum(n-1)

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.

Result:

Thus the python program to find the sum of natural number up to n using recursive function has been executed.

A5. Python program to merge two lists

AIM

To write a program merge two lists.

ALGORITHM

1. Start
2. Declare&Initializations of list.
3. Using + operator for computational processing of merge the two lists.
4. Return the output of the calculations.
5. Stop

SOURCE CODE

```
# Initializing lists
test_list3 = [1, 4, 5, 6, 5]

test_list4 = [3, 5, 7, 2, 5]

# using + operator to concat
test_list3 = test_list3 + test_list4

# Printing concatenated list print ("Merged list using + : "
                + str(test_list3))
```

Output

Merged list using + : [1, 4, 5, 6, 5, 3, 5, 7, 2, 5]

Result:

Thus the program to merge two list has been executed successfully.

A6.Python program to remove Duplicates elements from a List

ALGORITHM

Step 1: create a list.

Step 2: create a new list which is empty.

Step 3: traverse every element in list.

Step 4: if element is not present in the list return true.

Step 5: append in the new list.

Step 6: display new list.

SOURCE CODE

```
#  To remove duplicate elements
def removeduplicateele(A):
    newlist = []
    for n in A:
        if n not in newlist: newlist.append(n)
    return newlist
# Driver Code
A=list()
n=int(input("Enter the size of the List ::"))
print("Enter the number ::")
            fori in range(int(n)):
    k=int(input(""))
    A.append(int(k))
print("THE NEW LIST IS ::>",removeduplicateele(A))
```

Output

Enter the size of the List ::5

Enter the number ::

10

20

30

20

10

Result:

Thus the to remove duplicates elements from a List is obtained

A7.COUNT THE NUMBER OF WORDS IN A FILE

AIM:

To write a python program to count the number of words in a file.

ALGORITHM :

Step1. Create one variable to hold the file path. This is a constant variable. In the example we are showing here, you need to change this value with the file path in your own system. Also, initialize one more variable to hold the total count of words. Initialize this variable as zero.

Step2. Open the file in read-only mode. We are only reading the content of the file for this example. For counting the number of words in the file, read mode will be sufficient.

Step 3. Iterate through each line of the file using a loop. As this is a text file, we can iterate through the lines one by one.

Step4. Inside the loop, split the line into its words. Find out the total number of words and add them to the variable used to hold the total count of words. On each iteration of the loop, add the count of each line to this variable.

Step5. After the loop will complete, the word count variable will hold the total count of words in the text file. Print out the value of this variable to the user.

PROGRAM/SOURCE CODE :

```
word_count = 0

file_name = "D//in.txt"

with open(file_name,'r') as file:
        for line in file:
                word_count += len(line.split())

print ("number of words : ",word_count)
```

OUTPUT

```
Number of words : 9
```

RESULT:

Thus the finding the count of the number of words in file is obtained.

A8.GENERATE ALL PERMUTATIONS OF A GIVEN STRING

AIM:

To write a python program to generate all permutations of a given string.

ALGORITHM

Step1: Initially we will permutation function from the itertools module

Step 2: Take the string from the user and assign it in a variable s.

Step 3: Generate all permutation using the permutation function and assign it in a variable p.

Step 4: Since all elements of p are in tuple form. So, convert it in the list.

Step 5: At last, add all list elements and print it which is our possible permutations

Step 6: Print the output .

PROGRAM/SOURCE CODE :

```
# importing the module
fromitertools import permutations

# input the sting
s=input('Enter a string: ')

A=[]
b=[]
p=permutations(s)

for k in list(p):
        A.append(list(k))
        for j in A:
                r= ' '. join(str(l) for l in j)
b.append(r)

print('Number of all permutations: ',len(b))
print('All permutations are: ')
print(b)
```

OUTPUT:

```
Enter a string: ABC
Number of all permutations: 21
All permutations are:
['ABC', 'ABC', 'ACB', 'ABC', 'ACB', 'BAC', 'ABC', 'ACB', 'BAC', 'BCA', 'ABC',
'ACB', 'BAC', 'BCA', 'CAB', 'ABC', 'ACB', 'BAC', 'BCA', 'CAB', 'CBA']
```

RESULT:

Thus the program to generate all permutations of a string hasbeen executed successfully.

B. Viva Questions

1.What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

2. What are the uses of python?

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

3. Define Docstrings.

Python also has extended documentation capability, called docstrings. Docstrings can be one line, or multiline.

4.How to make Command lines in python?

Python has commenting capability for the purpose of in-code documentation.Comments start with a #, and Python will render the rest of the line as a comment.

5. Define variables in python.

Variables are containers for storing data values.

6.List Rules for Python variables.

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

7. List the numeric types in python.

There are three numeric types in Python:

- int
- float
- complex

8.What is the use of type() function?

To verify the type of any object in Python, use the type() function:

9. List Python Operators.

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

10. Define Python Lists or Python Collections (Arrays).

There are four collection data types in the Python programming language:

- List is a collection which is ordered and changeable. Allows duplicate members.
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
- Set is a collection which is unordered and unindexed. No duplicate members.
- Dictionary is a collection which is unordered, changeable and indexed. No duplicate members.

11. What are the various Python Conditions and If statements?

Python supports the usual logical conditions from mathematics:

- Equals: a == b
- Not Equals: a != b
- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

12.What is elif keyword in python?

The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition"

13. What is else keyword in python?

The else keyword catches anything which isn't caught by the preceding conditions.

14. List the types of Python Loops.

Python has two primitive loop commands:

- while loops
- for loops

15.What is while Loop?

With the while loop we can execute a set of statements as long as a condition is true.

16.Differentiate the break and continue Statement.

With the break statement we can stop the loop even if the while condition is true:

With the continue statement we can stop the current iteration, and continue with the next:

17. What is for loop?

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

18. Define range() function.

The range() Function

To loop through a set of code a specified number of times, we can use the range() function,

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

19. Define function.

A function is a block of code which only runs when it is called.You can pass data, known as parameters, into a function.A function can return data as a result.

20. How to create a Function in python?

In Python a function is defined using the def keyword:

21.How to Call a Function in python?

To call a function, use the function name followed by parenthesis:

22.What is Recursion?

Python also accepts function recursion, which means a defined function can call itself.

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.

23. Define Python Lambda.

A lambda function is a small anonymous function. A lambda function can take any number of arguments, but can only have one expression.

24.Why Use Lambda Functions?

The power of lambda is better shown when you use them as an anonymous function inside another function.Say you have a function definition that takes one argument, and that argument will be multiplied with an unknown number.

25. Define Python Classes/Objects.

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

26. What is the __init__() Function.

The examples above are classes and objects in their simplest form, and are not really useful in real life applications. To understand the meaning of classes we have to understand the built-in __init__() function. All classes have a function called __init__(), which is always executed when the class is being initiated. Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

27. Define Object Methods.

Objects can also contain methods. Methods in objects are functions that belongs to the object.

28.What is the self Parameter?

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

29. Define Python Inheritance

Inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from, also called base class.

Child class is the class that inherits from another class, also called derived class.

30. What are Python Iterators?

An iterator is an object that contains a countable number of values.An iterator is an object that can be iterated upon, meaning that you can traverse through all the values.Technically, in Python, an iterator is an object which implements the iterator protocol, which consist of the methods __iter__() and __next__().

31. Compare Iterator vsIterable

Lists, tuples, dictionaries, and sets are all iterable objects. They are iterable containers which you can get an iterator from.

All these objects have a iter() method which is used to get an iterator:

32. What is a Module?

Consider a module to be the same as a code library.

A file containing a set of functions you want to include in your application.

33. How to Create a Module?

To create a module just save the code you want in a file with the file extension .py:

34.What are Python Dates?

A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.

35. What is the use of JSON.

JSON is a syntax for storing and exchanging data. JSON is text, written with JavaScript object notation.

36. How to use JSON in Python?

Python has a built-in package called json, which can be used to work with JSON data.

37. What is RegEx ?

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern.

38. What is PIP?

PIP is a package manager for Python packages, or modules if you like.

39.What is a Package?

A package contains all the files you need for a module. Modules are Python code libraries you can include in your project.

40. Define Exception Handling.

When an error occurs, or exception as we call it, Python will normally stop and generate an error message. These exceptions can be handled using the try statement:

41. What is Python Try Except?

The try block lets you test a block of code for errors.

The except block lets you handle the error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

42. What are the File opening modes in python.

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

The open() function takes two parameters; filename, and mode.

There are four different methods (modes) for opening a file: "r" - Read - Default

value. Opens a file for reading, error if the file does not exist "a" - Append -

Opens a file for appending, creates the file if it does not exist "w" - Write -

Opens a file for writing, creates the file if it does not exist "x" - Create - Creates

the specified file, returns an error if the file exists