SRM VALLIAMMAI ENGINEERING COLLEGE

(AN AUTONOMOUS INSTITUTION)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF GENERAL ENGINEERING

LAB MANUAL

II SEMESTER

GE3232 - PROBLEM SOLVING AND PYTHON PROGRAMMING LABORATORY

Regulation – 2023

Academic Year 2024 – 2025 (EVEN Semester)

SYLLABUS

GE3232 - 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.

-Knowing about Object Oriented Concepts.

LIST OF PROGRAMS

1. Compute the GCD of two numbers.
2. Find the square root of a number. (Newton's method)
3. Find Exponentiation of a number. (power of a number)
4. Find the maximum of a list of numbers
5. Program for basic calculator operations using functions.
6. Generate Fibonacci series using function.
7. Program for Armstrong Number.
8. Program for check the number is Pallindrome or not.
9. Program for Sum of Array of numbers
10. How to create, slice, change, add, delete, and index elements using list.

11. Linear search and Binary search.

12. Find first n prime numbers.

13. Program to remove duplicate elements from the List.

14. Program for addition and transpose of a matrix.

15. How to create, slice, change, delete, and index elements using Tuple.

16. Write a program to reverse the string.

17. .How to change, delete, add and remove elements in Dictionary.

18. Create a dictionary of radius of circle and its circumference.

19. Program for count the number of words in a file.

20. Find the most frequent words in a text read from a file.

21. Program for student information system using class and objects.

22. Program for Employee payroll processing using class and objects.

PLATFORM NEEDED

Python 3 interpreter for Windows/Linux

TOTAL: 60 PERIODS

INDEX

S. No	Topic	Page No
1.	Compute the GCD of two numbers.	6
2.	Find the square root of a number (Newton's method)	8
3.	Find the Exponentiation of a number (power of a number)	10
4.	Find the maximum of a list of numbers	12
5.	Program for basic calculator operations using functions.	14
6.	Generate Fibonacci series using function.	18
7.	Program for Armstrong Number.	20
8.	Program for check the number is Pallindrome or not.	22
9.	Program for Sum of Array of numbers	24
10.	How to create, slice, change, add, delete, and index elements using list.	26
11.	Linear search and Binary search.	29
12.	Find first n prime numbers.	35
13.	Program to remove duplicate elements from the List.	37

14.	Program for addition and transpose of a matrix.	38
15.	How to create, slice, change, delete, and index elements using Tuple.	40
16.	Write a program to reverse the string.	42
17.	How to change, delete, add and remove elements in Dictionary.	43
18.	Create a dictionary of radius of circle and its circumference.	45
19.	Program for count the number of words in a file.	46
20.	Find the most frequent words in a text read from a file.	47
21.	Program for student information system using class and objects.	49
22.	Program for Employee payroll processing using class and objects.	52
A.	Additional Exercises	57
B.	Viva Questions	71

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:read two numbers to find the GCD n1,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

FIND THE 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 1Read the number of element in the list.

Step 2: Read the number until loop n -1.

Step 3: Then Append the all element in list

Step 4: Go to STEP -3 upto n-1.

Step 5: Sort the listed values.

Step 6: Print the a[n -1] value.

PROGRAM/SOURCE CODE:

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

print("Largest element is:",a[n -1])
a=[ ]
a.sort()
```

OUTPUT :

Enter number of elements:5

Enter element: 3

Enter element:2

Enter element:1

Enter element:5

Enter element:4

Largest element:5

RESULT:

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

EX.NO.5

PROGRAM FOR BASIC CALCULATOR USING FUNCTIONS

AIM:

Write a Python program to perform basic calculator using functions

ALGORITHM:

Step1:Start

Step2:Define the function

Step3:Check the given choice using if

Step4:If the choice matches then given arithmetic expression will be evaluated

Step5:If it not matches break the condition

Step5:End

PROGRAM/SOURCE CODE:

```
def add(x, y):
    return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
def divide(x, y):
    return x / y
print("Select operation.")
print("1.Add")
```

```
print("2.Subtract")

print("3.Multiply")

print("4.Divide")

while True:

    choice = input("Enter choice(1/2/3/4): ")

    if choice in ('1', '2', '3', '4'):

        try:

            num1 = float(input("Enter first number: "))

            num2 = float(input("Enter second number: "))

        except ValueError:

            print("Invalid input. Please enter a number.")

            continue



        if choice == '1':

            print(num1, "+", num2, "=", add(num1, num2))

        elif choice == '2':

            print(num1, "-", num2, "=", subtract(num1, num2))

        elif choice == '3':

            print(num1, "*", num2, "=", multiply(num1, num2))

        elif choice == '4':

            print(num1, "/", num2, "=", divide(num1, num2))

        next_calculation = input("Let's do next calculation? (yes/no): ")

        if next_calculation == "no":
```

```
        break
    else:
        print("Invalid Input")
```

OUTPUT:

```
Select operation.
1.Add
2.Subtract
3.Multiply
4.Divide
Enter choice(1/2/3/4): 1
Enter first number: 5
Enter second number: 6
5.0 + 6.0 = 11.0
Let's do next calculation? (yes/no): yes
Enter choice(1/2/3/4): 2
Enter first number: 7
Enter second number: 7
7.0 - 7.0 = 0.0
Let's do next calculation? (yes/no): yes
Enter choice(1/2/3/4): 3
Enter first number: 7
Enter second number: 7
7.0 * 7.0 = 49.0
```

Let's do next calculation? (yes/no):

Enter choice(1/2/3/4): 4

Enter first number: 4

Enter second number: 4

4.0 / 4.0 = 1.0

Let's do next calculation? (yes/no):

Enter choice(1/2/3/4): no

Invalid Input

RESULT:

Thus the program is executed for design the calculator to perform arithmetic operations using functions and the output is obtained.

EX.NO.6

GENERATE FIBONACCI SERIES USING FUNCTION

AIM:

Write a python program to generate Fibonacci series using function.

ALGORITHM:

Step1:Start

Step2:Get the number of terms

Step3: Check if the number of terms is valid

Step4:If there is only one term, return n1

Step5:If it not generate the fibonacci sequence upto n terms

Step5:End

PROGRAM/SOURCE CODE:

```
nterms = int(input("How many terms? "))
n1, n2 = 0, 1
count = 0
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence:")
   while count < nterms:
      print(n1)
      nth = n1 + n2
      # update values
      n1 = n2
```

```
    n2 = nth
    count += 1
```

OUTPUT:

```
How many terms? 5

Fibonacci sequence:

0

1

1

2

3
```

RESULT:

Thus the program is executed to find the Fibonacci series of a given number and the output is obtained.

EX.NO.7

PROGRAM FOR ARMSTRONG NUMBER

AIM:

Write a Python program to find the Armstrong number.Given a number x, determine whether the given number is Armstrong number or not. A positive integer of n digits is called an Armstrong number of order n (order is number of digits) if.

abcd... = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) + ...

ALGORITHM:

Step1:Start

Step2:Define Function to calculate x raised to the power y

Step3: Calculate order of the number

Step4:Then add the number with the sum

Step5:Define the function isArmstrong() to check the given is armstrong number or not.

Step6:If it the digit is a armstrong number

Step6:If it is not the digit is not a Armstrong number.

PROGRAM/SOURCE CODE:

```
def power(x, y):
    if y == 0:
        return 1
    if y % 2 == 0:
        return power(x, y // 2) * power(x, y // 2)
    return x * power(x, y // 2) * power(x, y // 2)
def order(x):
```

```
    n = 0
    while (x != 0):
        n = n + 1
        x = x // 10
    return n
def isArmstrong(x):
    n = order(x)
    temp = x
    sum1 = 0
    while (temp != 0):
        r = temp % 10
        sum1 = sum1 + power(r, n)
        temp = temp // 10
    return (sum1 == x)
x = 153
print(isArmstrong(x))
x = 1253
print(isArmstrong(x))
```

OUTPUT:

```
True
False
```

RESULT:

Thus the program is executed to find the given number is Armstrong number or not and the output is obtained.

EX.NO:8

PALLINDROME OF A DIGIT

AIM:

Write a Python program to reverse the digits of a given number and add them to the original. Repeat this procedure if the sum is not a palindrome.

Note: A palindrome is a word, number, or other sequence of characters which reads the same backward as forward, such as madam or race car.

ALGORITHM:

Step1:Start

Step2:Define the function

Step3:Check the position of the digits

Step4:If the end of the string Matches with the first string then given digit is a palindrome

Step5:If it not matches the digit is not a palindrome

Step5:End

PROGRAM/SOURCE CODE:

```
def rev_number(n):
 s = 0
 while True:
   k = str(n)
   if k == k[::-1]:
     break
   else:
     m = int(k[::-1])
     n += m
     s += 1
```

```
  return n
rev=int(input("enter num:"))
print(rev_number(rev))
```

OUTPUT:

```
Enter num: 145
Pallindrome number is
686
Enter num: 144
Pallindrome number is
585
```

RESULT:

Thus the program is executed to find the palindrome of a given number and the output is obtained

EX.NO.9

PROGRAM FOR SUM OF ARRAY OF NUMBERS

AIM:

Write a Python program to sum the array of numbers

ALGORITHM:

Step1:Start

Step2:Define the function

Step3:Initialize the variable

Step4:Check the condition while iterate through the array add each element to the sum variable

Step5:Input the values to the List

Step6:Calculate the length of the array using len()method.

Step7:Store the result in the variable sum and display the result.

Step5:End

PROGRAM/SOURCE CODE:

```
def _sum(arr):

sum = 0

for i in arr:

        sum = sum + i

return(sum)

if __name__ == "__main__":

arr = [12, 3, 4, 15]
```

```
n = len(arr)

ans = _sum(arr)

print('Sum of the array is ', ans)
```

OUTPUT:

```
Sum of the array is  34
```

RESULT:

Thus the program to calculate the sum of array is executed and the output is obtained.

Ex. No: 10

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 [ ].

Step3: Silicing an element from the List

Step4: 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[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 and the output is obtained**.**

Ex. No: 11a

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:")) for
i 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: 11b

BINARY SEARCH

AIM:

To write a python program to perform the binary search.

ALGORITHM:

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=input("enter no of elements:"))
for i in range(1,n+1):
        b=int(input("enter element"))
        arr.append(b)
print(arr)
element=int(input("enter 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.12

FIND FIRST 'N' PRIME NUMBERS

AIM:

To write a python program to find first 'N' prime numbers

ALGORITHM:

Step1:Define the function

Step2:If the number is 0 and 1 it rerurns false

Step3:Loop runs from 2 to n-1

Step4:If the number is divisible by i then n is not a prime number

Step5:If it is not then n is a prime number

Step6:End

PROGRAM/SOURCE CODE:

```
def isPrime(n):
    if(n==1 or n==0):
        return False
    for i in range(2,n):
            if(n%i==0):
            return False
    return True
N = 100;
for i in range(1,N+1):
```

```
    if(isPrime(i)):
        print(i,end=" ")
```

OUTPUT:

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

RESULT:

Thus the program was successfully executed to find the prime numbers and the output is obtained.

EX.NO.13

REMOVE DUPLICATE ELEMENTS FROM THE LIST

AIM:

To write a python program to remove duplicate elements from the List

ALGORITHM:

Step1:Define the function

Step2:Create an empty list to store the result

Step3:Check if the number is already presented in the created list

Step4:If it is not add the element in empty list

Step5:End

PROGRAM/SOURCE CODE:

```
def Remove(duplicate):
    final_list = []
    for num in duplicate:
        if num not in final_list:
            final_list.append(num)
    return final_list
duplicate = [2, 4, 10, 20, 5, 2, 20, 4]
print(Remove(duplicate))
```

OUTPUT:

```
[2, 4, 10, 20, 5]
```

RESULT:

Thus the program was executed to remove duplicate elements from the list and result is obtained.

EX.NO.14

PROGRAM FOR ADDITION AND TRANSPOSE OF A MATRIX

AIM:

To write a python program to add and transpose a matrix

ALGORITHM:

Step1:Define the function for addition and transpose of a matrix

Step2:Store the result in created result matrix

Step3:Rows and Colums are added until the length of the matrix A and B using for loop.

Step4: Rows and Colums are transposed until the length of the matrix X

Step5:End

PROGRAM/SOURCE CODE:

```
print("ADDITION OF TWO MATRICES")
def addM(A,B):
    result=[ [0,0,0],[0,0,0],[0,0,0] ]
    for i in range(len(A)):
        for j in range(len(A[0])):
            result[i][j]=A[i][j] + B[i][j]
    for k in result:
        print(k)
    return 0
A=[ [1, 2, 3], [3, 4, 5], [6, 7, 8] ]
B=[ [5, 6, 7], [1, 2, 3], [5, 3, 8] ]

addM(A,B)
print("TRANSPOSING A MATRIX")
def transpose(X):
    result = [[0,0,0],
              [0,0,0]]
    for i in range(len(A)):
```

```
        for j in range(len(X[0])):
                result[j][i] = X[i][j]
    for r in result:
      print(r)
    return 0
X = [[12,7],[4 ,5],[3 ,8]]
transpose(X)
```

OUTPUT:

```
ADDITION OF TWO MATRICES
[6, 8, 10]
[4, 6, 8]
[11, 10, 16]
TRANSPOSING A MATRIX
[12, 4, 3]
[7, 5, 8]
```

RESULT:

Thus the program was executed to add and transpose a given and result is obtained.

Ex. No: 15

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: Silicingthe tuple 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)

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])
Error

RESULT:

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

Ex. No: 16

REVERSE THE STRING

AIM:

To write a python program to reverse a string.

ALGORITHM:

Step1:Define a function reverse(s)

Step2:Read a string

Step3:Print the original string

Step4: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("original string is:",end=" ")
print(s)
print("The reversed string is:",end=" ")
print(reverse(s))
```

OUTPUT:

Enter a string:PYTHON PROGRAMMING

The original string is: PYTHON PROGRAMMING

The reversed string is:GNIMMARGORP NOHTYP

RESULT:

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

Ex. No: 17

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: 18

CREATE A DICTIONARY OF RADIUS OF CIRCLE AND ITS CIRCUMFERENCE.

AIM:

To write a python program to create a dictionary to access radius of circle to calculate area and its circumference.

ALGORITHM:

Step1:Define a dictionary with the radius values

Step2:Access the value

Step3:Calculate area and circumference of a circle

Step4: Access the result using items() method

Step4:Print the values using values() method

Step5:End

PROGRAM/SOURCE CODE:

```
radius={'rad1': 1.02, 'rad2': 2.5, 'rad3': 2.5}
pie = 3.14
print("Area of the circle")
area = {item: value*value*pie for (item, value) in radius.items()}
print (" The area of the given circle is: ",area.values())
print("Circumference of Circle: ")
circumference = {item: value*pie*2 for (item, value) in radius.items()}
print ("\nThe circumference of the given circle is: ",circumference.values())
```

OUTPUT:

```
Area of the circle
 The area of the given circle is:  dict_values([3.266856, 19.625, 19.625])
Circumference of Circle:
The circumference of the given circle is:  dict_values([6.405600000000001, 15.700000000000001,
15.700000000000001])
```

RESULT:

Thus the program to find the area and circumference of a circle using dictionary is executed and the output is obtained

Ex.No.19

PROGRAM TO COUNT THE NUMBER OF WORDS IN A FILE

AIM:

To write a python program to count the words from a file.

ALGORITHM:

Step1:Create a file

Step2:Open the created file in a read mode

Step3:Using split() split the content in a file

Step4:Count the number of words using len().

Step5:Print the words that are used in a file

Step6:Close the file and print the word

Step7:End

PROGRAM/SOURCE CODE:

```
file = open("data.txt", "r")
for line in file:
    words = line.split(" ");
        count = count + len(words);
   print("Number of words present in given file: " + str(count));
file.close();
```

OUTPUT:

Number of words present in given file: 63

RESULT:

Thus the program to find the number of words in a file is executed and the output is obtained

Ex. No: 20

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: 21

PROGRAM FOR STUDENT INFORMATION SYSTEM USING CLASS AND OBJECTS

AIM:

To write a python program to program for student information system using class and objects

ALGORITHM:

Step1:Create a class student

Step2:Define a function to create and append student

Step3:Define a function to search the given data in alist

Step4:Define a function to update the data in a list

Step5:Define the function to delete the data from the list

Step6:Create an object to access the data from the list

Step7:End

PROGRAM/SOURCE CODE :

```
class Student:

def __init__(self, name, rollno, m1, m2):
        self.name = name
        self.rollno = rollno
        self.m1 = m1
        self.m2 = m2

def accept(self, Name, Rollno, marks1, marks2):

        ob = Student(Name, Rollno, marks1, marks2)
        ls.append(ob)

def display(self, ob):
        print("Name : ", ob.name)
        print("RollNo : ", ob.rollno)
        print("Marks1 : ", ob.m1)
        print("Marks2 : ", ob.m2)
        print("\n")

def search(self, rn):
        for i in range(ls.__len__()):
```

```
                if(ls[i].rollno == rn):
                        return i

def delete(self, rn):
        i = obj.search(rn)
        del ls[i]

def update(self, rn, No):
        i = obj.search(rn)
        roll = No
        ls[i].rollno = roll

ls = []
obj = Student('', 0, 0, 0)

print("\nOperations used, ")
print("\n1.Accept Student details\n2.Display Student Details\n3.Search Details of a Student\n4.Delete
Details of Student\n5.Update Student Details\n6.Exit")
obj.accept("A", 1, 100, 100)
obj.accept("B", 2, 90, 90)
obj.accept("C", 3, 80, 80)
print("\n")
print("\nList of Students\n")
for i in range(ls.__len__()):
obj.display(ls[i])
print("\n Student Found, ")
s = obj.search(2)
obj.display(ls[s])
obj.delete(2)
print(ls.__len__())
print("List after deletion")
for i in range(ls.__len__()):
obj.display(ls[i])
obj.update(3, 2)
print(ls.__len__())
print("List after updation")
for i in range(ls.__len__()):
obj.display(ls[i])
print("Thank You !")
```

OUTPUT:

```
Operations used,

1.Accept Student details
2.Display Student Details
3.Search Details of a Student
4.Delete Details of Student
5.Update Student Details
6.Exit
```

```
List of Students
Name :  A
RollNo :  1
Marks1 :  100
Marks2 :  100

Name :  B
RollNo :  2
Marks1 :  90
Marks2 :  90

Name :  C
RollNo :  3
Marks1 :  80
Marks2 :  80

 Student Found,
Name :  B
RollNo :  2
Marks1 :  90
Marks2 :  90

2
List after deletion
Name :  A
RollNo :  1
Marks1 :  100
Marks2 :  100

Name :  C
RollNo :  3
Marks1 :  80
Marks2 :  80
2
List after updation
Name :  A
RollNo :  1
Marks1 :  100
Marks2 :  100
Name :  C
RollNo :  2
Marks1 :  80
Marks2 :  80
Thank You !
```

RESULT:

Thus the python program is executed using classes and object to display student information system and output is obtained

Ex. No: 22

PROGRAM FOR EMPLOYEE PAYROLL PROCESSING USING CLASS AND OBJECTS

AIM:

To write a python program for employee payroll processing using class and objects.

Write a Python class Employee with attributes like emp_id, emp_name, emp_salary, and emp_department and methods like calculate_emp_salary, emp_assign_department, and print_employee_details.

Sample Employee Data:
"ADAMS", "E7876", 50000, "ACCOUNTING"
"JONES", "E7499", 45000, "RESEARCH"
"MARTIN", "E7900", 50000, "SALES"
"SMITH", "E7698", 55000, "OPERATIONS"

- Use 'assign_department' method to change the department of an employee.
- Use 'print_employee_details' method to print the details of an employee.
- Use 'calculate_emp_salary' method takes two arguments: salary and hours_worked, which is the number of hours worked by the employee. If the number of hours worked is more than 50, the method computes overtime and adds it to the salary. Overtime is calculated as following formula:

overtime = hours_worked – 50
Overtime amount = (overtime * (salary / 50))

ALGORITHM:

Step1:Create a class employee

Step2:Define a function to calculate the salary

Step3:Define a function to assign the department of the employee

Step4:Define a function to print employee details

Step5:Create an object to access the data from the list

Step6:End

PROGRAM/SOURCE CODE :

```
class Employee:
    def __init__(self, name, emp_id, salary, department):
        self.name = name
        self.id = emp_id
        self.salary = salary
        self.department = department
    def calculate_salary(self, salary, hours_worked):
        overtime = 0
        if hours_worked > 50:
            overtime = hours_worked - 50
        self.salary = self.salary + (overtime * (self.salary / 50))
    def assign_department(self, emp_department):
        self.department = emp_department
    def print_employee_details(self):
        print("\nName: ", self.name)
        print("ID: ", self.id)
        print("Salary: ", self.salary)
        print("Department: ", self.department)
        print("----------------------")
employee1 = Employee("ADAMS", "E7876", 50000, "ACCOUNTING")
employee2 = Employee("JONES", "E7499", 45000, "RESEARCH")
employee3 = Employee("MARTIN", "E7900", 50000, "SALES")
employee4 = Employee("SMITH", "E7698", 55000, "OPERATIONS")
print("Original Employee Details:")
employee1.print_employee_details()
employee2.print_employee_details()
```

```
employee3.print_employee_details()

employee4.print_employee_details()

employee1.assign_department("OPERATIONS")

employee4.assign_department("SALES")

employee2.calculate_salary(45000, 52)

employee4.calculate_salary(45000, 60)

print("Updated Employee Details:")

employee1.print_employee_details()

employee2.print_employee_details()

employee3.print_employee_details()

employee4.print_employee_details()
```

OUTPUT:

```
Original Employee Details:

Name:  ADAMS

ID:  E7876

Salary:  50000

Department:  ACCOUNTING

----------

Name:  JONES

ID:  E7499

Salary:  45000

Department:  RESEARCH

----------

Name:  MARTIN

ID:  E7900

Salary:  50000

Department:  SALES
```

Name: SMITH

ID: E7698

Salary: 55000

Department: OPERATIONS

Updated Employee Details:

Name: ADAMS

ID: E7876

Salary: 50000

Department: OPERATIONS

Name: JONES

ID: E7499

Salary: 46800.0

Department: RESEARCH

Name: MARTIN

ID: E7900

Salary: 50000

Department: SALES

Name: SMITH

ID: E7698

Salary: 66000.0

Department: SALES

RESULT:

Thus the Program was executed to create employee payroll processing using classes and objects 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. Declare variables
2. Read the Input number.
3. Calculate sum of cubic of individual digits of the input.
4. Match the result with input number.
5. If match, Display the given number is Armstrong otherwise not.
6. Stop

SOURCE CODE

```
num = 1634

# Changed num variable to string, and calculated the length (number of digits)

order = len(str(num))

# initialize sum


# find the sum of the cube of each

digit temp = num



    sum = 0

    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

.

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

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 list has been executed successfully

A6.Python program to remove Duplicate elements from a List

AIM

To write a python program to remove duplicate elements from the list

ALGORITHM

Step1: Create a List

Step2: Create a new list which is empty.

Step3: 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:

```
def Remove(duplicate):
    final_list = []
    for num in duplicate:
        if num not in final_list:
            final_list.append(num)
    return final_list


# Driver Code
duplicate = [2, 4, 10, 20, 5, 2, 20, 4]
print(Remove(duplicate))
```

Output

Enter the size of the List ::5

Enter the number ::

10

20

30

20

10

RESULT:

Thus to remove the duplicate elements from the 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.

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 has been 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:

- Equals: a == b
- Not Equals: a != b

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:

- 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.

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

38.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.

39.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:

40. 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.

41. 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