Lab File

**AS1105 Computational Mathematics**

PREPARED BY

**Rishi Sharma**

**2021BCA021**

FACULTY GUIDE

**Dr. Jaya Gupta**



**Department of Computer Science Engineering**

**Institute of Engineering and Technology (IET)**

**JK Lakshmipat University Jaipur**

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Name of Experiment** | **Date** | **Sign/Comment** |
| 1. | Matrix addition & subtraction | 3-02-2022 |  |
| 2. | Matrix multiplication | 10-02-2022 |  |
| 3. | Determinate of matrix | 17-02-2022 |  |
| 4. | Inversion of matrix | 24-02-2022 |  |
| 5. | Solution of equations using bisection method | 3-03-2022 |  |
| 6. | Solution of equations using Regula-falsi method | 10-03-2022 |  |
| 7. | Solutions of equations Newton Raphson method | 17-03-2022 |  |
| 8. | Forward interpolation method | 24-03-2022 |  |
| 9. | backward interpolation method | 31-03-2022 |  |
| 10. | Integration simpson's method | 7-04-2022 |  |
| 11. | Integration trapezoidal method | 21-04-2022 |  |

**1. Compute the matrix addition and subtraction using python?**

Code –

* **Matrix addition**

*# Python program to add two matrices user input*

r = int(input("Enter the rows: "))

c = int(input("Enter the columns: "))

print("Enter Matrix 1:")

m1 = [[int(input()) for i in range(c)] for i in range(r)]

print("Matrix 1 is: ")

for n in m1:

   print(n)

print("Enter Matrix 2:")

m2 = [[int(input()) for i in range(c)] for i in range(r)]

for n in m2:

   print(n)

r = [[0 for i in range(c)] for i in range(r)]

for i in range(len(r)):

   for j in range(c):

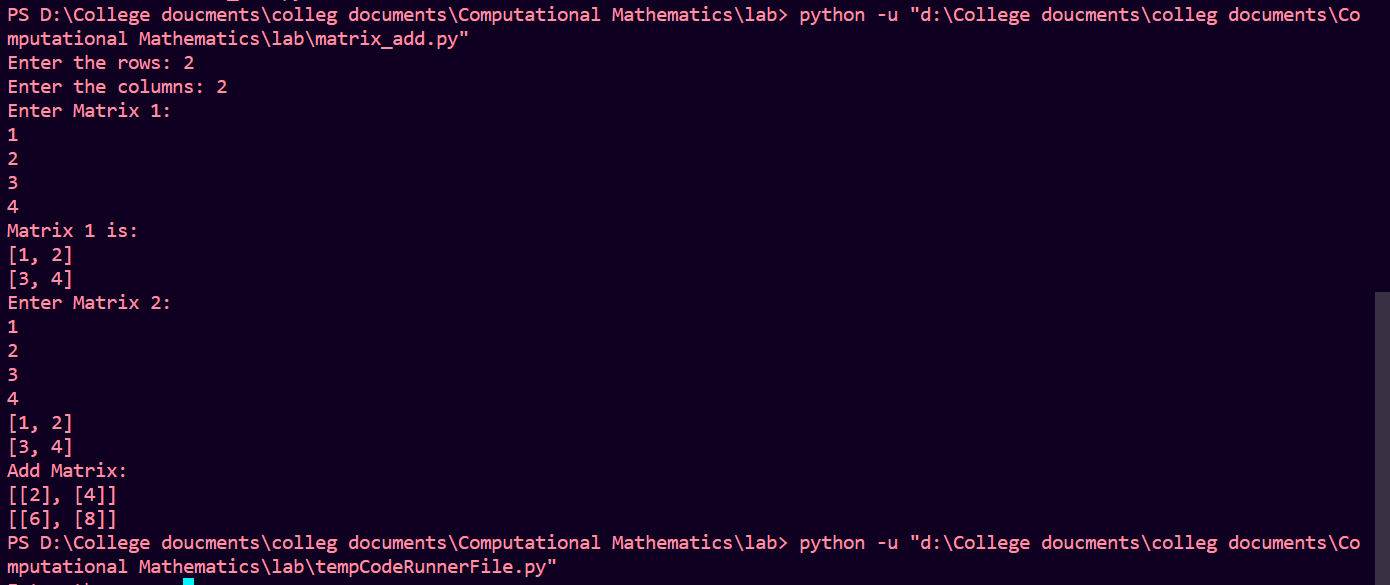
      r[i][j] = [m1[i][j] + m2[i][j]]

print("Add Matrix:")

for i in r:

   print(i)

**Output: -**

****

* **Matrix Subtraction**

*# Python program to add two matrices user input*

r = int(input("Enter the rows: "))

c = int(input("Enter the columns: "))

print("Enter Matrix 1:")

m1 = [[int(input()) for i in range(c)] for i in range(r)]

print("Matrix 1 is: ")

for n in m1:

   print(n)

print("Enter Matrix 2:")

m2 = [[int(input()) for i in range(c)] for i in range(r)]

for n in m2:

   print(n)

r = [[0 for i in range(c)] for i in range(r)]

for i in range(len(r)):

   for j in range(c):

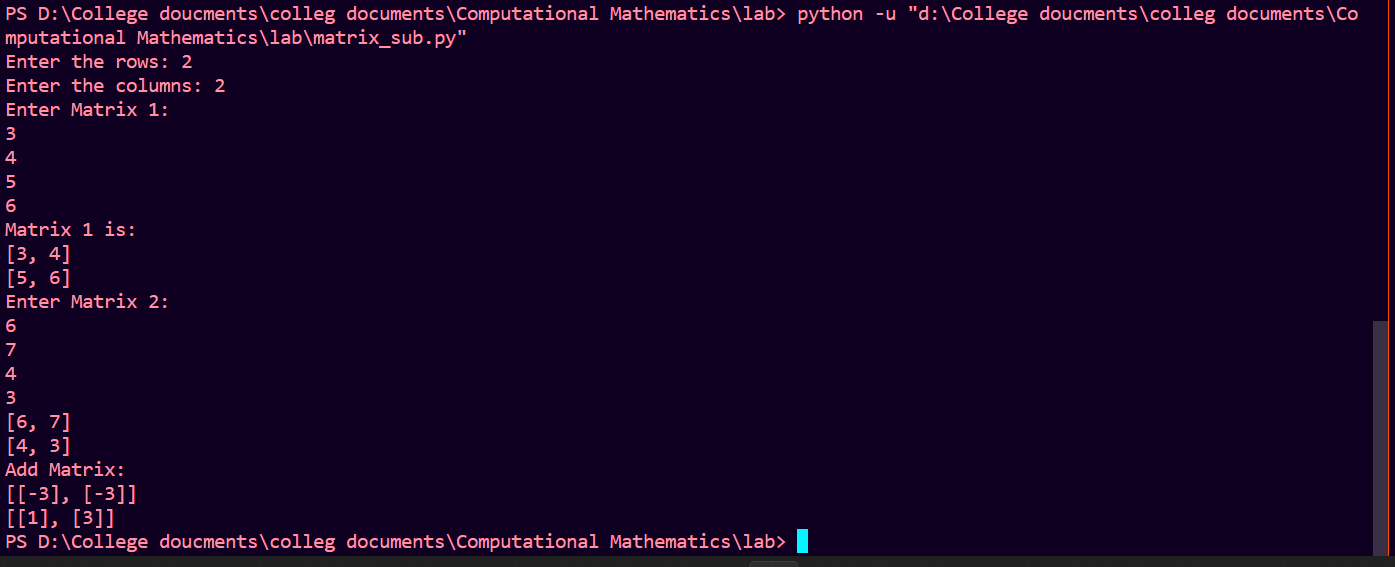
      r[i][j] = [m1[i][j] - m2[i][j]]

print("Add Matrix:")

for i in r:

   print(i)

**Output: -**



**2. Compute the Metrix multiplication using python?**

Code –

row1=int(input("Enter row for matrix1 "))

col1=int(input("Enter column for matrix1 "))

row2=int(input("Enter row for matrix2 "))

col2=int(input("Enter column for matrix2 "))

list1=[]

list2=[]

if(col1!=row2):print("Not possible")

else:

  for i in range (row1):

      a=[]

      for j in range(col1):

          a.append(int(input()))

      list1.append(a)

  for i in range (row2):

      a=[]

      for j in range(col2):

          a.append(int(input()))

      list2.append(a)

  matrix=[]

  for i in range(row1):

      a=[]

      for j in range(col2):

          sum=0

          for k in range(col1):

              sum+=(list1[i][k]\*list2[k][j])

          a.append(sum)

      matrix.append(a)

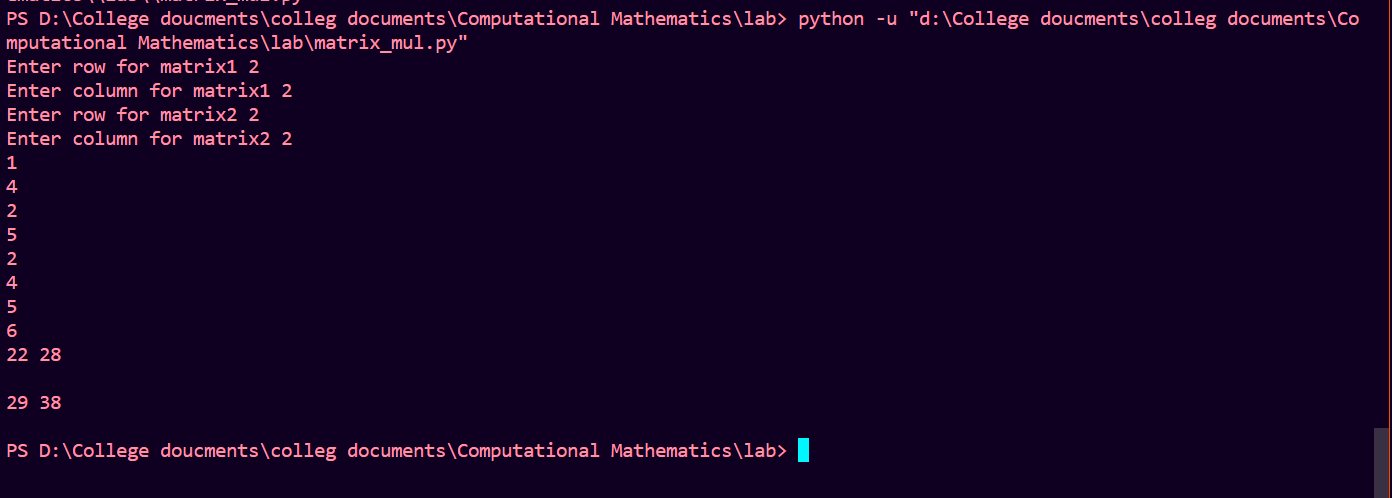
  for i in range(row1):

      for j in range(col2):

          print(matrix[i][j],end=" ")

      print("\n")

Output –



**3. Calculation of the Metrix determinant using python?**

Code –

*# python program to find*

*# determinant of matrix.*

*# defining a function to get the*

*# minor matrix after excluding*

*# i-th row and j-th column.*

def getcofactor(m, i, j):

    return [row[: j] + row[j+1:] for row in (m[: i] + m[i+1:])]

*# defining the function to*

*# calculate determinant value*

*# of given matrix a.*

def determinantOfMatrix(mat):

*# if given matrix is of order*

*# 2\*2 then simply return det*

*# value by cross multiplying*

*# elements of matrix.*

    if(len(mat) == 2):

        value = mat[0][0] \* mat[1][1] - mat[1][0] \* mat[0][1]

        return value

*# initialize Sum to zero*

    Sum = 0

*# loop to traverse each column*

*# of matrix a.*

    for current\_column in range(len(mat)):

*# calculating the sign corresponding*

*# to co-factor of that sub matrix.*

        sign = (-1) \*\* (current\_column)

*# calling the function recursily to*

*# get determinant value of*

*# sub matrix obtained.*

        sub\_det = determinantOfMatrix(getcofactor(mat, 0, current\_column))

*# adding the calculated determinant*

*# value of particular column*

*# matrix to total Sum.*

        Sum += (sign \* mat[0][current\_column] \* sub\_det)

*# returning the final Sum*

    return Sum

*# Driver code*

row1=int(input("Enter row for matrix1 "))

col1=int(input("Enter column for matrix1 "))

list1=[]

for i in range (row1):

      a=[]

      for j in range(col1):

          a.append(int(input()))

      list1.append(a)

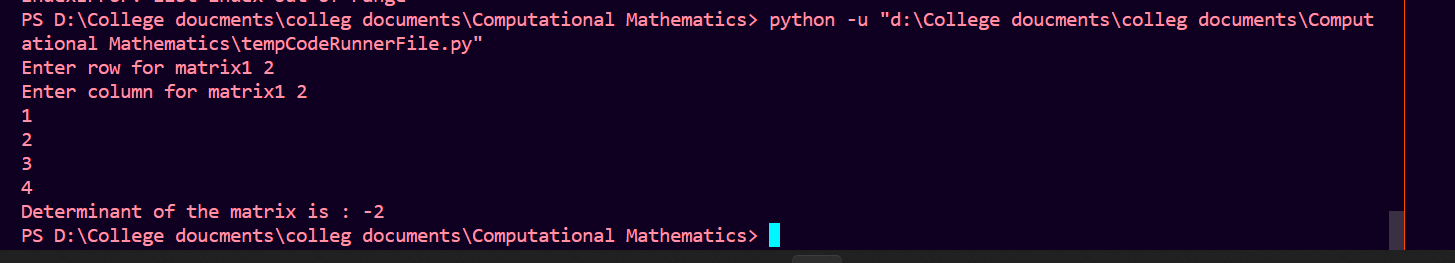
*# declaring the matrix.*

*# printing determinant value*

*# by function call*

print('Determinant of the matrix is :', determinantOfMatrix(list1))

Output –



**4. Compute the Matrix inversion using python?**

Code –

*# Importing Library*

import numpy as np

*# Finding an inverse of given array*

arr = np.array([[1, 2], [5, 6]])

inverse\_array = np.linalg.inv(arr)

print("Inverse array is ")

print(inverse\_array)

print()

*# inverse of 3X3 matrix*

arr = np.array([[1, 2, 3],

                [4, 9, 6],

                [7, 8, 9]])

inverse\_array = np.linalg.inv(arr)

print("Inverse array is ")

print(inverse\_array)

print()

*# inverse of 4X4 matrix*

arr = np.array([[1, 2, 3, 4],

                [10, 11, 14, 25],

                [20, 8, 7, 55],

                [40, 41, 42, 43]])

inverse\_array = np.linalg.inv(arr)

print("Inverse array is ")

print(inverse\_array)

print()

*# inverse of 1X1 matrix*

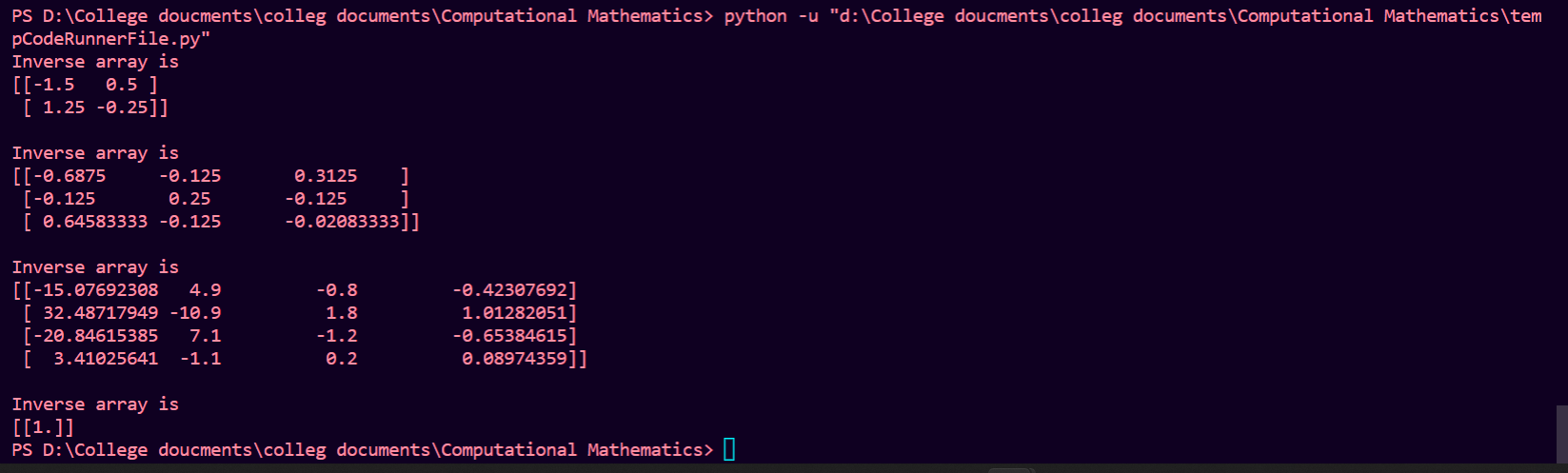
arr = np.array([[1]])

inverse\_array = np.linalg.inv(arr)

print("Inverse array is ")

print(inverse\_array)

Output –



**5. Solution of equations using bisection method in python?**

Code –

*# Python program for implementation of Bisection Method for solving equations*

*# An example function whose solution is determined using Bisection Method. The function is x^3 - x^2 + 2*

from re import X

def func(x):

    return x\*x\*x - 9\*x + 1

print ()

*# Prints root of func(x) with error of EPSILON*

def bisection(a,b):

    if (func(a) \* func(b) >= 0):

        print("You have not assumed right a and b\n")

        return

    c = a

    print (c)

    while ((b-a) >= 0.01):

*# Find middle point*

        c = (a+b)/2

        print (c)

*# Check if middle point is root*

        if (func(c) == 0.0):

            break

*# Decide the side to repeat the steps*

        if (func(c)\*func(a) < 0):

            b = c

        else:

            a = c

    print("The value of root is : ","%.4f"%c)

*# Driver code*

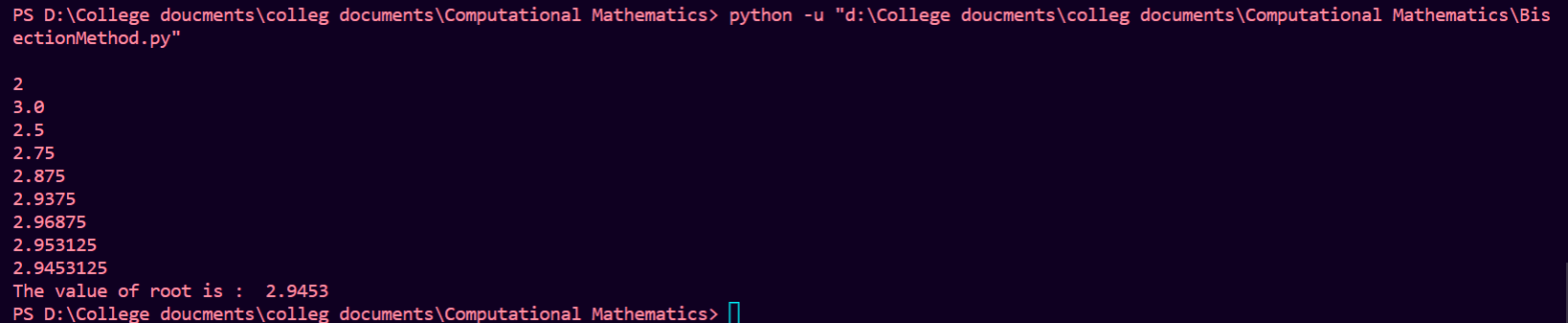
*# Initial values assumed*

a = 2

b = 4

bisection(a, b)

Output –



**6. Solution of equations using Regula-falsi method in python?**

Code –

*# Python3 implementation of Bisection*

*# Method for solving equations*

MAX\_ITER = 10

*# An example function whose solution*

*# is determined using Bisection Method.*

*# The function is x^3 - x^2 + 2*

def func( x ):

    return (x \* x \* x - 9\*x + 1)

*# Prints root of func(x) in interval [a, b]*

def regulaFalsi( a , b):

    if func(a) \* func(b) >= 0:

        print("You have not assumed right a and b")

        return -1

    c = a *# Initialize result*

    for i in range(MAX\_ITER):

*# Find the point that touches x axis*

        c = (a \* func(b) - b \* func(a))/ (func(b) - func(a))

        print (c)

*# Check if the above found point is root*

        if func(c) == 0:

            break

*# Decide the side to repeat the steps*

        elif func(c) \* func(a) < 0:

            b = c

        else:

            a = c

    print("The value of root is : " , '%.4f' %c)

*# Driver code to test above function*

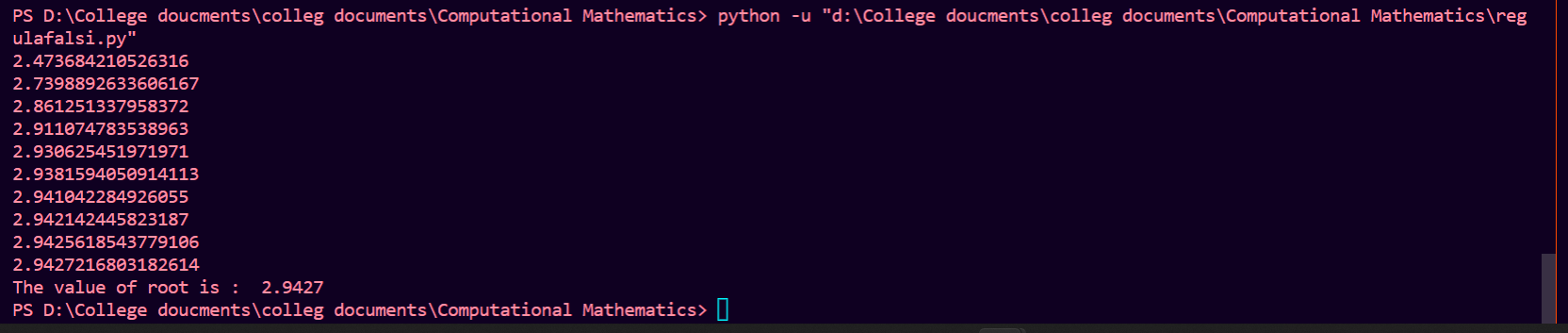
*# Initial values assumed*

a = 2

b = 4

regulaFalsi(a, b)

Output –



**7. Solutions of equations Newton Raphson method in python?**

Code –

*# Python3 code for implementation of Newton*

*# Raphson Method for solving equations*

*# The function is x^3 - x^2 + 2*

def func( x ):

    return x \* x \* x - 9 \* x + 1

*# Derivative of the above function*

*# which is 3\*x^x - 2\*x*

def derivFunc( x ):

    return 3 \* x \* x - 9

*# Function to find the root*

def newtonRaphson( x ):

    h = func(x) / derivFunc(x)

    while abs(h) >= 0.0001:

        h = func(x)/derivFunc(x)

*# x(i+1) = x(i) - f(x) / f'(x)*

        x = x - h

    print("The value of the root is : ",

                            "%.4f"% x)

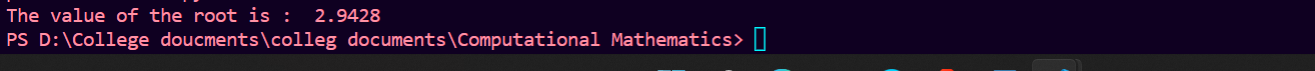
*# Driver program to test above*

x0 = 2 *# Initial values assumed*

*# print = (x0)*

newtonRaphson(x0)

Output –



**8. forward interpolation method in python?**

Code –

def u\_cal(u, n):

    temp = u

    for i in range(1, n):

        temp = temp \* (u - i)

    return temp

def fact(n):

    f = 1

    for i in range(2, n + 1):

        f \*= i

    return f

n = int(input("Enter number of entries: "))

x = []

y = [[0 for i in range(n)]

        for j in range(n)]

for i in range(n):

    a=int(input())

    x.append(a)

for i in range(n):

    y[i][0]=float(input())

for i in range(1, n):

    for j in range(n - i):

        y[j][i] = y[j + 1][i - 1] - y[j][i - 1]

value = int(input("Enter value: "))

for i in range(n):

    print(x[i], end = "\t")

    for j in range(n - i):

        print(y[i][j], end = "\t")

    print("")

sum = y[0][0]

u = (value - x[0]) / (x[1] - x[0])

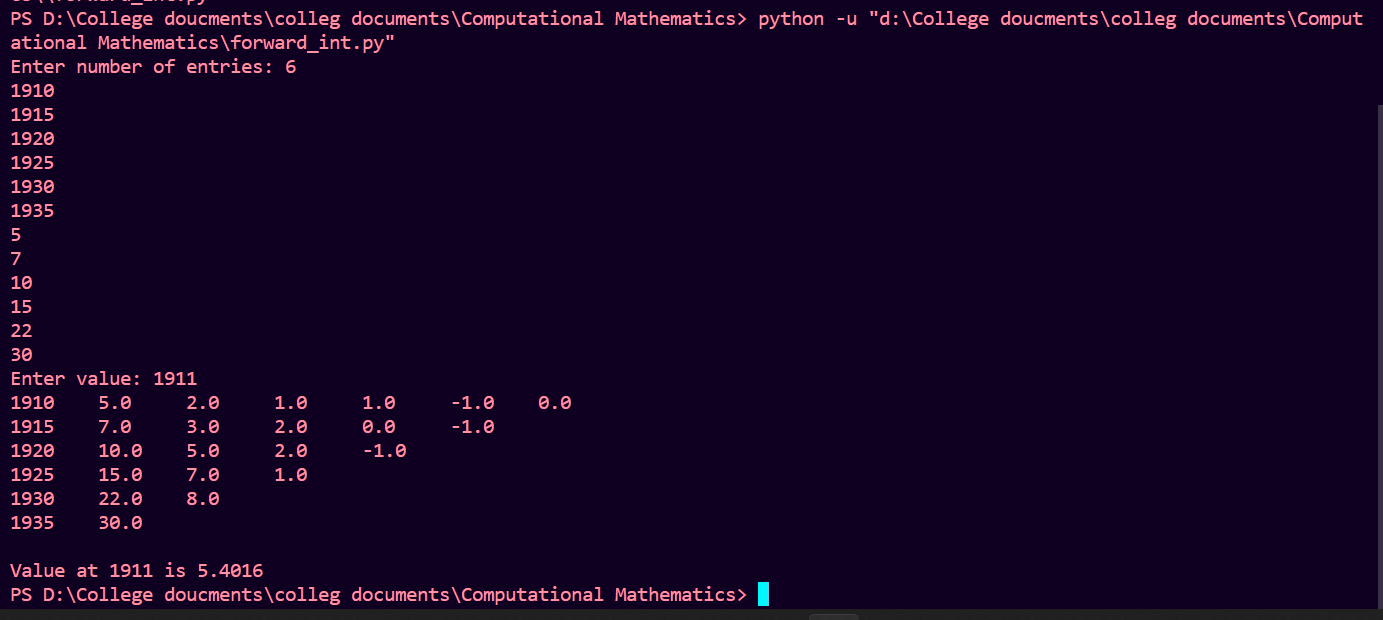
for i in range(1,n):

    sum = sum + (u\_cal(u, i) \* y[0][i]) / fact(i)

print("\nValue at", value,

    "is", round(sum, 6))

Output –



**9. backward interpolation method in python?**

Code –

def u\_cal(u, n):

    temp = u

    for i in range (1,n):

        temp = temp \* (u + i)

    return temp

def fact(n):

    f = 1

    for i in range(2, n + 1):

        f \*= i

    return f

n = int(input("Enter number of entries: "))

x = []

y = [[0 for i in range(n)]

        for j in range(n)]

for i in range(n):

    a=int(input())

    x.append(a)

for i in range(n):

    y[i][0]=float(input())

for i in range(1, n):

    for j in range(n - 1,i+1,-1):

        y[j][i] = y[j][i - 1] - y[j-1][i - 1]

for i in range(n):

    print(x[i], end = "\t")

    for j in range(i+1):

        print(y[i][j], end = "\t")

    print("")

value = int(input("Enter value "))

sum = y[n-1][0]

u = (value - x[n-1]) / (x[1] - x[0])

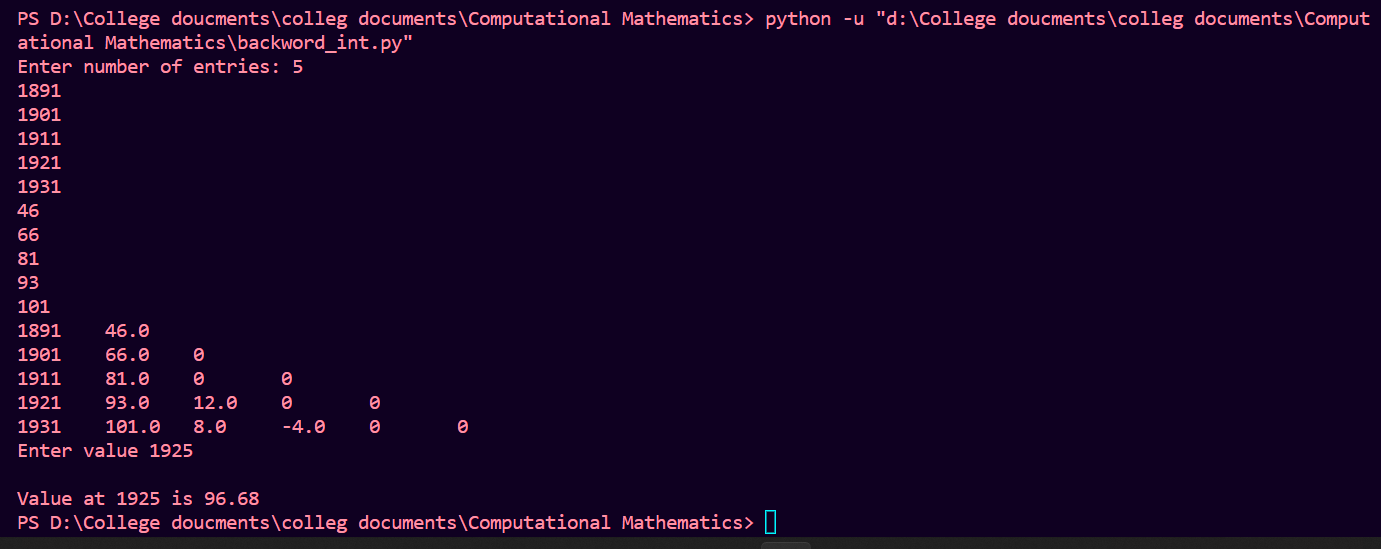
for i in range(1,n):

    sum = sum + (u\_cal(u, i) \* y[n-1][i]) / fact(i)

print("\nValue at", value,

    "is", round(sum, 6))

Output –



10. Integration simpson's method code in python?

Code –

1/3

from cmath import pi, sin

def function(x):

    return (sin(x)).real

n=int(input("enter no. of intervals "))

upper=pi

lower=0

h=(upper-lower)/n

arr=[]

for i in range(n):

    arr.append(function(lower))

    lower+=h

sum2=0

for i in range(2,n-1,2):

    sum2+=arr[i]

sum3=0

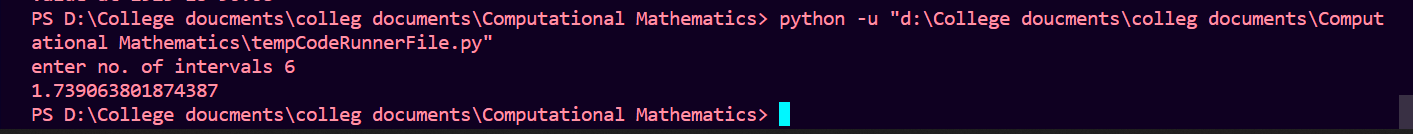
for i in range(1,n-1,2):

    sum3+=arr[i]

ans=h\*(arr[0]+arr[5]+2\*(sum2)+4\*(sum3))/3

print(ans)

Output –



3/8

from cmath import pi, sin

def function(x):

    return (sin(x)).real

n=int(input("enter no. of intervals "))

upper=pi

lower=0

h=(upper-lower)/n

arr=[]

for i in range(n):

    arr.append(function(lower))

    lower+=h

sum3=0

for i in range(3,n-1,3):

    sum3+=arr[i]

sum=0

for i in range(1,n-1):

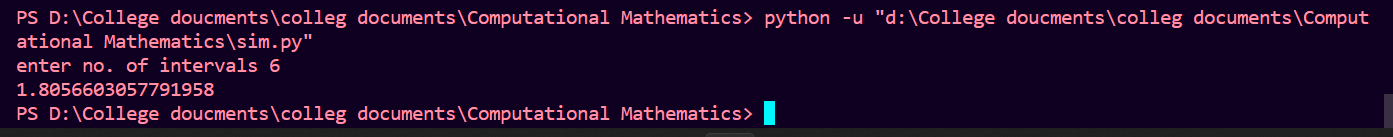
    if i%3!=0:

        sum+=arr[i]

ans=3\*h\*(arr[0]+arr[5]+2\*(sum3)+3\*(sum))/8

print(ans)

output –



11. Integration trapezoidal code in python?

Code –

from cmath import pi, sin

def function(x):

    return (sin(x)).real

n=int(input("enter the no of interval "))

upper=pi

lower=0

h=(upper-lower)/n

arr=[]

for i in range(n):

    arr.append(function(lower))

    lower+=h

sum=0

for i in range(1,n-1):

    sum+=arr[i]

ans=h\*((arr[0]+arr[n-1])+2\*(sum))/2

print(ans)

Output –

