Slip 1

Que:1 Attempt any one the

A] Write Python program to obtained the approximate real root of x 3– 4x – 9 = 0 by using Regula-falsi method

```
def falseposition(f, x0, x1, e):
    x0 = float(x0)
    x1 = float(x1)
    e = float(e)


    if f(x0) * f(x1) > 0.0:
        print('The given values do not bracket a root. Try different values.')
    else:
        step = 1
        condition = True

        while condition:

            x2 = x0 - (x1 - x0) * f(x0) / (f(x1) - f(x0))
            print('Iteration %d, x2 = %.6f, f(x2) = %.6f' % (step, x2, f(x2)))


            if f(x0) * f(x2) < 0:
                x1 = x2
            else:
                x0 = x2
            if abs(f(x2)) <= e:
                condition = False
            else:
                step += 1

        print('\nRequired root is %.8f' % x2)



def f(x):
    return x**3-4*x-9


falseposition(f, 1.0, 3.0, 0.00001)
```

```
Iteration 1, x2 = 2.333333, f(x2) = -5.629630
Iteration 2, x2 = 2.656051, f(x2) = -0.886809
Iteration 3, x2 = 2.700341, f(x2) = -0.110905
Iteration 4, x2 = 2.705779, f(x2) = -0.013451
Iteration 5, x2 = 2.706438, f(x2) = -0.001625
Iteration 6, x2 = 2.706517, f(x2) = -0.000196
Iteration 7, x2 = 2.706527, f(x2) = -0.000024
Iteration 8, x2 = 2.706528, f(x2) = -0.000003

Required root is 2.70652780
```

B] **Write Python program to evaluate interpolate value f(3) of the given data by Lagranges method. x 0 1 2 5 Y = f(x) 5**

```
n=int(input("enter number of data points:"))
enter number of data points:4
x=np.zeros((n))
y=np.zeros((n))
```

```
print("enter the data for x and y:")
enter the data for x and y:
for i in range(n):
    x[i]=float(input('x['+str(i)+')='))
    x[i]=float(input('y['+str(i)+')='))


x[0)=0
y[0)=5
x[1)=1
y[1)=13
x[2)=2
y[2)=22
x[3)=5
y[3)=129
xp=float(input("enter interpolation point:"))
enter interpolation point:3
yp=0

for i in range(n):
    p=1
    for j in range(n):
        if i!=j:
            p=p*(xp-x[j])/(x[i]-x[j])
            yp=yp+p*y[i]
    print(f'interpolated value at {xp} is {yp}')
interpolated value at 3.0 is 0.0
interpolated value at 3.0 is 0.0
interpolated value at 3.0 is 0.0
interpolated value at 3.0 is 0.0
```

Que:2 Attempt the following: [10 Marks]

(a) Write Python program to estimate the value of the integral 0 �

```
def s13(a,b,n,f):
   h=float(b-a)/n
   I=f(a)+f(b)
   for i in range(1,n):
      k=a+i*h
      if i%2==0:
         I=I+2*f(k)
      else:
         I=I+4*f(k)
         I=(h/3)*I
         return I



def f(x):
   return math.sin(x)
s13(0,math.pi,6,f)
0.3490658503988659
```

Write a python function that checks whether a given string is pangram or not.

```
def is_pangram(s):
    alphabet = set(string.ascii_lowercase)
    return set(s.lower()) >= alphabet
import string
text ="The quick brown fox jumps over the lazy dog"
print(f"Is the given text a pangram? {is_pangram(text)}")
```

Is the given text a pangram? True

```
def longest_word_length(word_list):

   return max(len(word) for word in word_list) if word_list else 0


words = ["apple", "banana", "strawberry", "blueberry"]

print(f"Length of the longest word: {longest_word_length(words)}")

Length of the longest word: 10
```

(a) Write a Python function that takes a list of words and returns the length of the longest one.

```
def longest_word_length(word_list):

   return max(len(word) for word in word_list) if word_list else 0

words = ["apple", "banana", "strawberry", "blueberry"]

print(f"Length of the longest word: {longest_word_length(words)}")
```

Length of the longest word: 10

B]Write a Python program to add ‘ing’ at the end of

```
import math
import string
s1="play"
s2="ing"
print(s1+s2)
playing
```

Slip 2

Que:1 Attempt any one the following: [10 Marks]

(a) Write Python program to obtained a real root of f (x) = x 3– 8x − 4 = 0 by using Newton–Raphson method

```
def n_r(f,g,x0,e,N):

   x0=float(x0)

   e=float(e)

   N=int(N)

   step=1

   flag=1
```

```
    condition=True
    while condition:
        if g(x0)==0.0:
            print("Divide by zero error:")
            break
        x1=x0-f(x0)/g(x0)
        print('Iteration-%d,x1=%0.6f and f(x1)=%0.6f'%(step,x1,f(x1)))
        x0=x1
        step +=1
        if step>N:
            flag=0
            break
        condition=abs(f(x1))>e
        if flag==1:
            print('\n required root is %0.8f'%x1)
        else:
            print('\n not convergent')
def f(x):
    return x**3-5*x+1


def g(x):
    return 3*x**2-5
x0=0.1
e=0.00001
N=100
n_r(f,g,x0,e,N)
```

```
Iteration-1,x1=0.200805 and f(x1)=0.004073
```

```
  required root is 0.20080483
  Iteration-2,x1=0.201640 and f(x1)=0.000000
 required root is 0.20163959
```

Que:2 Attempt the following: [10 Marks]

(a) Write Python program to estimate the value of the integral 0 1 1 1+�

```
def s13(a,b,n,f):
    h=float(b-a)/n
    I=f(a)+f(b)
    for i in range(1,n):
        k=a+i*h
        if i%2==0:
            I=I+2*f(k)
        else:
            I=I+4*f(k)
            I=(h/3)*I
            return I
def f(x):
    return 1/(1+x**2)
s13(0,1,4,f)
0.4387254901960784
```

Write python code that takes number as parameter and checks whether number is prime or not

```
def is_prime(n):
    if n<=1:
        return False
    for i in range(2,int(n**0.5)+1):
        if n%i==0:
            return False
        return True
```

```
print(is_prime(11))

True

print(is_prime(13))

True

print(is_prime(7))

True

print(is_prime(6))

False
```

Que:3 Attempt any one of the following the following: [5Marks]

A]Use Python code to generate the square root of numbers from 21 t0 49

```
import math
for i in range(21,50):
   print(f'square root of {i} is {math}.sqrt(i)')
```

```
square root of 21 is <module 'math' (built-in)>.sqrt(i)
square root of 22 is <module 'math' (built-in)>.sqrt(i)
square root of 23 is <module 'math' (built-in)>.sqrt(i)
square root of 24 is <module 'math' (built-in)>.sqrt(i)
square root of 25 is <module 'math' (built-in)>.sqrt(i)
square root of 26 is <module 'math' (built-in)>.sqrt(i)
square root of 27 is <module 'math' (built-in)>.sqrt(i)
square root of 28 is <module 'math' (built-in)>.sqrt(i)
square root of 29 is <module 'math' (built-in)>.sqrt(i)
square root of 30 is <module 'math' (built-in)>.sqrt(i)
square root of 31 is <module 'math' (built-in)>.sqrt(i)
square root of 32 is <module 'math' (built-in)>.sqrt(i)
square root of 33 is <module 'math' (built-in)>.sqrt(i)
square root of 34 is <module 'math' (built-in)>.sqrt(i)
square root of 35 is <module 'math' (built-in)>.sqrt(i)
square root of 36 is <module 'math' (built-in)>.sqrt(i)
square root of 37 is <module 'math' (built-in)>.sqrt(i)
square root of 38 is <module 'math' (built-in)>.sqrt(i)
square root of 39 is <module 'math' (built-in)>.sqrt(i)
square root of 40 is <module 'math' (built-in)>.sqrt(i)
square root of 41 is <module 'math' (built-in)>.sqrt(i)
square root of 42 is <module 'math' (built-in)>.sqrt(i)
square root of 43 is <module 'math' (built-in)>.sqrt(i)
square root of 44 is <module 'math' (built-in)>.sqrt(i)
square root of 45 is <module 'math' (built-in)>.sqrt(i)
square root of 46 is <module 'math' (built-in)>.sqrt(i)
square root of 47 is <module 'math' (built-in)>.sqrt(i)
square root of 48 is <module 'math' (built-in)>.sqrt(i)
square root of 49 is <module 'math' (built-in)>.sqrt(i)
```

b) Write the Python code to print ‘Python is bad’ and ‘Python is wonderful’ , where Wonderful is global variable and bad is local v

```
def print_statements():
   bad = "bad"  # Local variable
   print(f"Python is {bad}")
wonderful = "wonderful"
def print_global():
   print(f"Python is {wonderful}")
```

print_statements()

```
Python is bad
```

print_global

```
<function print_global at 0x000001591886DA80>
```

Slip 3

Que:1 Attempt any one the following: [10 Marks]
a]Write Python program to estimate a root of an equation f (x) = 3x − cos(x) − 1 using Newton–Raphson method correct up to four decimal place

```
def n_r(f,g,x0,e,N):
   x0=float(x0)
   e=float(e)
   N=int(N)
   step=1
```

```
    flag=1
    condition=True
    while condition:
        if g(x0)==0.0:
            print("Divide by zero error:")
            break
        x1=x0-f(x0)/g(x0)
        print('Iteration-%d,x1=%0.6f and f(x1)=%0.6f'%(step,x1,f(x1)))
        x0=x1
        step +=1
        if step>N:
            flag=0
            break
        condition=abs(f(x1))>e
        if flag==1:
            print('\n required root is %0.8f'%x1)
        else:
            print('\n not convergent')
def f(x):
    return 3*x-math.cos(x)-1

def g(x):
    return 3+math.sin(x)

x0=0.5
e=0.00005
N=100
```

```
n_r(f,g,x0,e,N)
Iteration-1,x1=0.608519 and f(x1)=0.005060


 required root is 0.60851865
Iteration-2,x1=0.607102 and f(x1)=0.000001


 required root is 0.60710188
```

Que:2 Attempt the following: [10 Marks]

(a] Write Python program to estimate the value of the integral 2 10 1 1 + �

```
def t(a,b,n):
    h=(b-a)/n
    result=f(a)+f(b)
    for i in range(1,n):
        result +=2*f(a+i*h)
    result *=h/2
    return result
a=2
b=10
n=5
integral_estimate=t(a,b,n)
print("estimated value of the integral",integral_estimate)
estimated value of the integral 1.3206255135651455
```

(b**] Write Python code to find the square of odd numbers from 1 to 20 using while loop**

```
i=1
while i<=20:
    if i%2!=0:
        print(f"{i} squared is {i**2}")
```

```
  i +=1

1 squared is 1

3 squared is 9

5 squared is 25

7 squared is 49

9 squared is 81

11 squared is 121

13 squared is 169

15 squared is 225

17 squared is 289

19 squared is 361
```

Que:3 Attempt any one of the following the following: [5Marks]

(a)Write Python code to find check whether passed string is paliondrom

```
def is_palindrome(s):

   return s==s[::-1]

print(is_palindrome("Madam"))

False

print(is_palindrome("mam"))

True

print(is_palindrome("madam"))

True
```

B]Write Python program to find the product of n natural numbers using

```
def product_of_n_natural_numbers(n):
    """Calculate the product of the first n natural numbers."""
    if n < 1:
        return None
    product = 1
    for i in range(1, n + 1):
        product *= i
    return product
```

```
product_of_n_natural_numbers(8)
40320
product_of_n_natural_numbers(4)
```

Slip 4

Que:1 Attempt any one the following: [10 Marks]

(a) Write Python program to estimate a root of an equation $f(x) = 3x 2 + 4x-10$ using Regula- Falsi correct up to four decimal

```
import math
def falseposition(f, x0, x1, e):
    x0 = float(x0)
    x1 = float(x1)
    e = float(e)


    if f(x0) * f(x1) > 0.0:
        print('The given values do not bracket a root. Try different values.')
    else:
        step = 1
        condition = True

        while condition:

            x2 = x0 - (x1 - x0) * f(x0) / (f(x1) - f(x0))
            print('Iteration %d, x2 = %.6f, f(x2) = %.6f' % (step, x2, f(x2)))


            if f(x0) * f(x2) < 0:
                x1 = x2
            else:
                x0 = x2


            if abs(f(x2)) <= e:
                condition = False
            else:
                step += 1
        print('\nRequired root is %.8f' % x2)



def f(x):
    return 3*x**2 + 4*x - 10

falseposition(f, 1.0, 3.0, 0.00001)
Iteration 1, x2 = 1.187500, f(x2) = -1.019531
Iteration 2, x2 = 1.249057, f(x2) = -0.323346
Iteration 3, x2 = 1.268364, f(x2) = -0.100301
Iteration 4, x2 = 1.274333, f(x2) = -0.030899
Iteration 5, x2 = 1.276169, f(x2) = -0.009498
Iteration 6, x2 = 1.276734, f(x2) = -0.002918
Iteration 7, x2 = 1.276907, f(x2) = -0.000896
Iteration 8, x2 = 1.276960, f(x2) = -0.000275
Iteration 9, x2 = 1.276977, f(x2) = -0.000085
Iteration 10, x2 = 1.276982, f(x2) = -0.000026
Iteration 11, x2 = 1.276983, f(x2) = -0.000008

Required root is 1.27698328
```

Que:2 Attempt the following: [10 Marks]

(a) Write Python program to estimate the value of the integral 0 �

```
def s13(a,b,n,f):

  h=float(b-a)/n

  I=f(a)+f(b)

  for i in range(1,n):

    k=a+i*h

    if i%2==0:

      I=I+2*f(k)

    else:

      I=I+4*f(k)

      I=(h/3)*I

      return I
```

```
def f(x):

   return math.sin(x)

s13(0,math.pi,6,f)

0.3490658503988659
```

b) Generate all relatively prime numbers to 111 which are less than 150 using Python code.

```
lower=111
upper=150
print("Prime numbers between", lower, "and", upper, "are:")
Prime numbers between 111 and 150 are:
for num in range(lower, upper + 1):
  # all prime numbers are greater than 1
  if num > 1:
      for i in range(2, num):
          if (num % i) == 0:
              break
      else:
          print(num)



113
127
131
137
139
149
```

Que:3 Attempt any one of the following the following: [5Marks]

(a) Write a Python program to change a given string to a new string w

Slip 21

Write Python program to obtained the approximate real root of x 3− 4x − 9 = 0 by using Regula-falsi method

```
def falseposition(f, x0, x1, e):

   x0 = float(x0)

   x1 = float(x1)
```

```
e = float(e)


if f(x0) * f(x1) > 0.0:
    print('The given values do not bracket a root. Try different values.')
else:
    step = 1
    condition = True

    while condition:

        x2 = x0 - (x1 - x0) * f(x0) / (f(x1) - f(x0))
        print('Iteration %d, x2 = %.6f, f(x2) = %.6f' % (step, x2, f(x2)))


        if f(x0) * f(x2) < 0:
            x1 = x2
        else:
            x0 = x2


        if abs(f(x2)) <= e:
            condition = False
        else:
            step += 1

    print('\nRequired root is %.8f' % x2)
```

```
def f(x):
    return x**3-4*x-9


falseposition(f, 1.0, 3.0, 0.00001)
Iteration 1, x2 = 2.333333, f(x2) = -5.629630
Iteration 2, x2 = 2.656051, f(x2) = -0.886809
Iteration 3, x2 = 2.700341, f(x2) = -0.110905
Iteration 4, x2 = 2.705779, f(x2) = -0.013451
Iteration 5, x2 = 2.706438, f(x2) = -0.001625
Iteration 6, x2 = 2.706517, f(x2) = -0.000196
Iteration 7, x2 = 2.706527, f(x2) = -0.000024
Iteration 8, x2 = 2.706528, f(x2) = -0.000003


Required root is 2.70652780
```

Write Python program to evaluate interpolate value f(2.2) of the given data f(2) = 0.593, f(2.5)=0.816, f(3)=1.078 using Lagran

```
import numpy as np
n=int(input("enter number of data points:"))
enter number of data points:3
x=np.zeros((n))
y=np.zeros((n))
print("enter the data for x and y:")
enter the data for x and y:
for i in range(n):
    x[i]=float(input('x['+str(i)+')='))
    x[i]=float(input('y['+str(i)+')='))



x[0)=2
```

```
y[0)=0.593

x[1)=2.5

y[1)=0.816

x[2)=3

y[2)=1.078

xp=float(input("enter interpolation point:"))

enter interpolation point:2.2

yp=0

for i in range(n):

    p=1

    for j in range(n):

        if i!=j:

            p=p*(xp-x[j])/(x[i]-x[j])

            yp=yp+p*y[i]

    printf(f'interpolated value at {xp} is {yp}')
```

```
for i in range(n):

    p=1

    for j in range(n):

        if i!=j:

            p=p*(xp-x[j])/(x[i]-x[j])

            yp=yp+p*y[i]

    print(f'interpolated value at {xp} is {yp}')
```

```
interpolated value at 2.2 is 0.0

interpolated value at 2.2 is 0.0

interpolated value at 2.2 is 0.0
```

Write Python program to estimate the value of the integral 0 �

```
def s13(a,b,n,f):
    h=float(b-a)/n
    I=f(a)+f(b)
    for i in range(1,n):
        k=a+i*h
        if i%2==0:
            I=I+2*f(k)
        else:
            I=I+4*f(k)
            I=(h/3)*I
            return I



def f(x):
    return math.sin(x)
s13(0,math.pi,6,f)
0.3490658503988659
```

Write Python program to find absolute value of a given real number

```
import math
def absolute_value(n):
    return abs(n)
num = float(input("Enter a number: "))
Enter a number: -9.56473
print(f"The absolute value of {num} is {absolute_value(num)}")
```

The absolute value of **-9.56473** is **9.56473**

b) Write the Python code to print ‘Python is bad’ and ‘Python is wonderful’ , where Wonderful is global variable and bad is local v

```
def print_statements():
    bad = "bad"  # Local variable
    print(f"Python is {bad}")
wonderful = "wonderful"
def print_global():
    print(f"Python is {wonderful}")
```

print_statements()

```
Python is bad
```

print_global

```
<function print_global at 0x000001591886DA80>
```

Slip 20