SLIP 11

Q.1,

1

a)

```
m = [1,2,3,4,5,6,7]

print(len(m))
```

b).

```
l = "xyz" + "pqr"

print(l)
```

c).

```
s = "make in India"

print(s[:5])

print(s[:9])
```

2.

```
from sympy import *

A = Matrix([[1,1,1],[0,1,1],[0,0,1]])
```

a) `print(A.eigenvals())`

b) `print(A.det())`

c) `print(A.eigenvects())`

3.

```
S = [3,4,5,6,7,8,9,10,11,12,13]

S.reverse()

print(S)
```

Q2.

1.

```
#Distionary

college = {"Dr. Ansari sir":"DS","Pramod Deore":"Math", "Nisha Sharma":"Electronics", "Sunil Nikam":"C", "Sharad Bachhav":"Statistics"}

college
```

2.

```
from sympy import *

x,y,z = symbols('x,y,z')

A = Matrix([[1,2,-3],[-2,1,-2],[3,1,-2]])

b = Matrix([3,6,9])

linsolve((A,b),[x,y,z])
```

3.

```
for num in range(51,100):

    if num > 1:

        for i in range(2, num):

            if (num % i) == 0:
```

```
          break
    else:
        print(num)
```

Q 3.

a.

1.

```
import math
def simson3by8Rule(f,a,b,n):
    h = 0.5
    sm = f(a) + f(b)
    print("X0 = %.4f"%sm)

    for i in range(1,n):
        k = a + i*h
        y = f(k)
        if i%3 == 0:
            sm = sm + 2 *y
        else:
            sm = sm + 3 * y
        print("X%d = %.4f"%(i,y))
    y = f(b)
    print("X%d = %.4f"%(n,f(b)))
    sm = sm + h    * (3/8)
    print("Value of intergral    is : %.5f"%sm)
```

```
f = lambda x : math.e**x
a = float(input("Enter lower limit : "))
b = float(input("Enter upper limit : "))
n = int(input("Enter Total subintaraval : "))
simson3by8Rule(f,a,b,n)
```

2.

```
def newtonRaphson(f,df,x,n):
    h = f(x) / df(x)
    for n in range(1,n+1):
        h = f(x) / df(x)
        x = x -h

    print("The value newton Raphson method : ","%.4f"%x)

f = lambda x : x**5+ 3*x +1
df = lambda x : 5*x**4+3
x0 = 0
newtonRaphson(f,df,x0,5)
```

b.

1.

```
def falsepositionmethod(a,b,e):
    if f(a) * f(b) >=0:
        print("False positionmethod false ")

        print("enterval cheak ")

    else:

        step    = 1

        print("** false position method emplimention ** ")

        condition = True

        while condition:

            c = a-(b-a)* f(a)/(f(a) - f(b))

            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))

            if f(a) * f(c) <0:

                b = c

            else:

                a = c

            step = step +1

            condition = abs(f(c)) >e

            print("\n required root is : %.8f"%c)

f = lambda x : x**3-4*x-9

a = float(input("Enter lower limit : "))

b = float(input("Enter upper limit : "))

e =    float(input("Tolebrable error : "))


falsepositionmethod(a,b,e)
```

2.

```
from scipy.interpolate import interp1d
```

```
X = [150,152,154,155] # random x values

Y = [12.247,12.329,12.410,12.490] # random y values

interpolate_x = 153

y_interp = interp1d(X, Y)

print("Value of Y at x = {} is".format(interpolate_x), y_interp(interpolate_x))
```

SLIP 12

Q.1

1.

a.

```
(22 %2 )+ 9 - (3 +7) × 10 ÷ 2
```

b.

```
35 * 10 // 3 + 15 %    3
```

C.

```
2*5 - 2*5 + 4 // 7
```

2.

```
for i in range(25, 50):
    if i % 2 != 0:
        print(i)
```

3.

a)

```
from sympy import *
A = Matrix([[1, 0 , 5, 4], [2, 1, 6, -1], [3, 4, 0, 2]])
A.row_del(1)
A
```

b)

```
from sympy import *
```

```
A = Matrix([[1, 0 , 5, 4], [2, 1, 6, -1], [3, 4, 0, 2]])

A.col_del(1)

A
```

C.

Q2

1

```
from sympy import *

A = Matrix([[1,3,3],[2,3,3],[4,2,1]])

print(A.eigenvals())

print(A.det())

print(A.eigenvects())
```

2.

3.

```
for num in range(1,200):

    if num > 1:

        for i in range(2, num):

            if (num % i) == 0:

                break

            else:

                print(num)
```

Q.3

a)

1.

```
import math
def simsonthirdMethod(f,a,b,n):
    h = (b-a)/2
    sm = f(a) + f(b)
    print('x0 = %f'%f(a))

    for i in range(1,n):
        k = a + i*h
        y = f(k)
        if i%2 == 0:
            sm = sm + 2 * y
        else:
            sm = sm + 4 * y
        print('x%d = %f'%(i,y))

    y = f(b)
    print('x%d = %f'%(n,y))
    sm = sm * h /3
    print('value of integral is %.8f'%sm)
    return
```

```
f = lambda x : math.sin(x)

a = float(input("Enter lower limit : "))

b = float(input("Enter upper limit : "))

n = int(input("Enter Total no of subintraval : "))


simsonthirdMethod(f,a,b,n)
```

2.

```
from sympy import *

A = Matrix([[3,-2],[6,-4]])

print(A.is_diagonalizable())

P, D = A.diagonalize()

P

D
```

b.

1.

```
def falsepositionmethod(a,b,e):

    if f(a) * f(b) >=0:

        print("False positionmethod false ")

        print("enterval cheak ")

    else:

        step    = 1
        print("** false position method emplimention ** ")
        condition = True
        while condition:
            c = a-(b-a)* f(a)/(f(a) - f(b))
            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))
            if f(a) * f(c) <0:
                b = c
            else:
                a = c
            step = step +1
            condition = abs(f(c)) >e
            print("\n required root is : %.8f"%c)
f = lambda x : x**3-2*x-5
a = float(input("Enter lower limit : "))
b = float(input("Enter upper limit : "))
e =    float(input("Tolebrable error : "))
```

2.

```
def trap(f,a,b,n):
    h = (b-a)/2
    sm = f(a) + f(b)

    for i in range(1,n):
        k = a+ i*h
        y = f(k)
        sm = sm + 2*y

        print('x %d = %f'%(i,y))


    sm = sm * h/2

    print('\n value of integral is %0.8f'%sm)

    return


f = lambda x : 1/(1+x)

a = float(input("Enter lower limit : "))

b = float(input("Enter upper limit : "))

n = int(input("Enter subintaral num    : "))


trap(f,a,b,n)
```

SLIP 13

Q.1

1.

```
z1 = 3+2j

z2 = -4 + 1j

print(z1 + z2)

print(z1 - z2)

print(z1 * z2)
```

2.

3.

```
for i   in range(1,11):

        print(i*i)
```

Q.2

1.

```
S = [1,2,3,4,5,6,7,8,9]

S.reverse()

S
```

2.

```
f = lambda x : x**2-3*x

for i in range(-1,4):
    y = f(i)

    print(y)
```

3.

```
sm = 0

for i in range(50,100):

    sm = sm + i

print(sm/50)
```

<u>Q.3</u>

1

```
import math

def simsonthirdMethod(f,a,b,n):

    h = (b-a)/2

    sm = f(a) + f(b)

    print('x0 = %f'%f(a))


    for i in range(1,n):

        k = a + i*h

        y = f(k)

        if i%2 == 0:

            sm = sm + 2 * y

        else:

            sm = sm + 4 * y

        print('x%d = %f'%(i,y))
    y = f(b)
    print('x%d = %f'%(n,y))
    sm = sm * h /3
    print('value of integral is %.8f'%sm)
    return

f = lambda x : (1+x**3)**(1/2)
a = float(input("Enter lower limit : "))
b = float(input("Enter upper limit : "))
n = int(input("Enter Total no of subintraval : "))

simsonthirdMethod(f,a,b,n)
```

2.

```
from scipy.interpolate import interp1d
X = [3,5,7,9]
Y = [5,7,27,64]
interpolate_x = 5.5
y_interp = interp1d(X, Y)
print("Value of Y at x = {} is".format(interpolate_x),

    y_interp(interpolate_x))
```

b)

1.

```
def falsepositionmethod(a,b,e):
    if f(a) * f(b) >=0:
        print("False positionmethod false ")
        print("enterval cheak ")
    else:
        step    = 1
        print("** false position method emplimention ** ")
        condition = True
        while condition:
            c = a-(b-a)* f(a)/(f(a) - f(b))
            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))
            if f(a) * f(c) <0:
                b = c
            else:
                a = c
            step = step +1
            condition = abs(f(c)) >e
            print("\n required root is : %.8f"%c)
f = lambda x : x**3-4*x-9
a = float(input("Enter lower limit : "))
b = float(input("Enter upper limit : "))
e =    float(input("Tolebrable error : "))
```

```
falsepositionmethod(a,b,e)
```

2.

```
import numpy as np

import sys


n = int(input('Enter number of data points: '))

x = np.zeros((n))

y = np.zeros((n,n))


print('Enter data for x and y: ')

for i in range(n):

        x[i] = float(input( 'x['+str(i)+']='))

        y[i][0] = float(input( 'y['+str(i)+']='))

for i in range(1,n):

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

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

print('\nBACKWARD DIFFERENCE TABLE\n');

for i in range(0,n):

        print('%0.2f' %(x[i]), end='')

        for j in range(0, i+1):

                print('\t%0.2f' %(y[i][j]), end='')

    print()
```

SLIP 14

Q.1

1.

```
a =4
b = 6
c = 8
d = 12
print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

2.

```
print(3+(9-2)/7*2**2)
```

3.

```
string = "Helow"
string1 = "Word!"
string2 ="Good"
string3 ="Moring"
```

```
print(string + string1)

print(string + string2)

print(string + string3)
```

Q.2

1.

```
from sympy import *

A = Matrix([[1,2],[4,8]])

B = Matrix([[4,7],[7,9]])

print(A+B)

print(B+A)

print("Both matrix is same")

print("A + B = B + A")

print(A-B)
```

2.

```
f = lambda x : x + 5

for i in range(-5,6):

        y = f(i)

        print(y)
```

3.

```
from sympy import *

A = Matrix([[4,2,2],[2,4,2],[2,2,4]])

print(A.eigenvals())

print(A.eigenvects())
```

Q.3

A)

1.

```
import math

def simsonthirdMethod(f,a,b,n):

    h = (b-a)/2

    sm = f(a) + f(b)

    print('x0 = %f'%f(a))


    for i in range(1,n):

        k = a + i*h

        y = f(k)

        if i%2 == 0:

            sm = sm + 2 * y

        else:

            sm = sm + 4 * y
        print('x%d = %f'%(i,y))

    y = f(b)
    print('x%d = %f'%(n,y))
    sm = sm * h /3
    print('value of integral is %.8f'%sm)
    return

f = lambda x : (1 /(1+x**2))
a = float(input("Enter lower limit : "))
b = float(input("Enter upper limit : "))
n = int(input("Enter Total no of subintraval : "))

simsonthirdMethod(f,a,b,n)
```

2.

```
def newtonRaphson(f,df,x,n):
    h = f(x) / df(x)
    for n in range(1,n+1):
        h = f(x) / df(x)
        x = x -h
    print("The value newton Raphson method : ","%.4f"%x)


f = lambda x : x**3 - 8*x - 4
df = lambda x : 2*x+5
x0 = 0.5
newtonRaphson(f,df,x0,5)
```

B)

1.

```
def falsepositionmethod(a,b,e):
    if f(a) * f(b) >=0:
        print("False positionmethod false ")
        print("enterval cheak ")
    else:
        step    = 1
        print("** false position method emplimention ** ")
        condition = True
        while condition:
            c = a-(b-a)* f(a)/(f(a) - f(b))
            print('Iteration-%d, c = %0.4f and f(c) = %0.4f' % (step, c, f(c)))
            if f(a) * f(c) <0:
                b = c
            else:
                a = c
            step = step +1
            condition = abs(f(c)) >e
            print("\n required root is : %.3f"%c)
f = lambda x : x**3 - 2 * x -4
a = float(input("Enter lower limit : "))
b = float(input("Enter upper limit : "))
e =    float(input("Tolebrable error : "))

falsepositionmethod(a,b,e)
```

2.

```
# Reading number of unknowns
n = int(input('Enter number of data points: '))

# Making numpy array of n & n x n size and initializing
# to zero for storing x and y value along with differences of y
x = np.zeros((n))
y = np.zeros((n,n))
```

```
# Reading data points
print('Enter data for x and y: ')
for i in range(n):
        x[i] = float(input( 'x['+str(i)+']='))
        y[i][0] = float(input( 'y['+str(i)+']='))

# Generating forward difference table
for i in range(1,n):
        for j in range(0,n-i):
                y[j][i] = y[j+1][i-1] - y[j][i-1]



print('\nFORWARD DIFFERENCE TABLE\n');


for i in range(0,n):
        print('%0.2f' %(x[i]), end='')
        for j in range(0, n-i):
                print('\t\t%0.2f' %(y[i][j]), end='')
    print()
```

SLIP 15

Q.1

1.

```
for i in range(1,12):
        print(i)
```

2.

```
a)      print(math.sin(75))
b)      print(math.sin(75))
c)      print(math.e)
d)      print(math.cos(56))
```

3.

```
r = 5
print("area : ",3.14*r*r)
print("circularense : ",2*3.14*r)
print("dimeter : ",2*r)
```

Q.2

1.

```
from sympy import *

A = Matrix([[10,20],[30.,40]])

B = Matrix([[40,50],[50.,60]])

C = Matrix([[70,80],[80.,90]])

print((A+B)+C)

print(A+(B+C))

print("Both matrix same ")

print("(A+B)+C = A+(B+C)")
```

2.

```
from sympy import *

A = Matrix([[3,-2],[6,-4]])

print(A.eigenvals())

print(A.eigenvects())
```

3.

```
for num in range(1000,2000):

    if num > 1:

        for i in range(2, num):

            if (num % i) == 0:

                break

        else:
```

```
      print(num)
```

Q.3

A)

1.

```
import math
def simsonthirdMethod(f,a,b,n):
    h = (b-a)/2
    sm = f(a) + f(b)
    print('x0 = %f'%f(a))

    for i in range(1,n):
        k = a + i*h
        y = f(k)
        if i%2 == 0:
            sm = sm + 2 * y
        else:
            sm = sm + 4 * y
        print('x%d = %f'%(i,y))

    y = f(b)
    print('x%d = %f'%(n,y))
    sm = sm * h /3
    print('value of integral is %.8f'%sm)
    return

f = lambda x : math.e**x
a = float(input("Enter lower limit : "))
b = float(input("Enter upper limit : "))
n = int(input("Enter Total no of subintraval : "))

simsonthirdMethod(f,a,b,n)
```

2.

```
def newtonRaphson(f,df,x,n):
    h = f(x) / df(x)
    for n in range(1,n+1):
        h = f(x) / df(x)
        x = x -h

    print("The value newton Raphson method : ","%.4f"%x)

f = lambda x : 3*x-math.cos(x)-1
df = lambda x :3 + math.sin(x)
```

```
x0 = 0.5

newtonRaphson(f,df,x0,5)
```

B)

1.

```
def falsepositionmethod(a,b,e):

    if f(a) * f(b) >=0:

        print("False positionmethod false ")

        print("enterval cheak ")

    else:

        step    = 1

        print("** false position method emplimention ** ")

        condition = True

        while condition:

            c = a-(b-a)* f(a)/(f(a) - f(b))

            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))

            if f(a) * f(c) <0:

                b = c

            else:

                a = c

            step = step +1

            condition = abs(f(c)) >e
```

```
                    print("\n required root is : %.8f"%c)


f = lambda x : x**3 - 4*x -9

a = float(input("Enter lower limit : "))

b = float(input("Enter upper limit : "))

e =    float(input("Tolebrable error : "))


falsepositionmethod(a,b,e)
```

2.

```
from scipy.interpolate import interp1d

X = [2,2.5,3]

Y = [0.593,0.816,1.078]

interpolate_x = 2.5

y_interp = interp1d(X, Y)

print("Value of Y at x = {} is".format(interpolate_x),


        y_interp(interpolate_x))
```

SLIP 16

Q.1

1.

```
def findAbsolute(N):
        if (N < 0):
                N = (-1) * N ;
        print(N);
if _name_ == '_main_':
        N = -12;
        findAbsolute(N);
```

2.

```
List1 = [5,10,15,20,25,30]
List2 = [7,14,21,28,35,42]
a)      print(List1 + List2)
b)      print(7*List1)
c)      print(11*List1)
```

3.

```
r = 5
print("area : ",3.14*r*r)
print("circularense : ",2*3.14*r)
```

Q.2

1.

```
marks = (70,80,55,78,65)
sm =0
for i in range(len(marks)):
        sm = sm + marks[i]
print(sm/5)
```

2.

```
from sympy import *
x =[1,-5,0]
y = [2,3,-1]
print(5*x)
print(x+y)
print(x-3*y)
```

3.

```
# Python program to inverse
# a matrix using numpy


# Import required package


import numpy as np
```

```
# Taking a 4 * 4 matrix

A = np.array([[6, 1, 1, 3],

              [4, -2, 5, 1],

              [2, 8, 7, 6],

              [3, 1, 9, 7]])


# Calculating the inverse of the matrix

print(np.linalg.inv(A))
```

Q.3

a)

1.

```
import math
def simsonthirdMethod(f,a,b,n):
        h = (b-a)/2
        sm = f(a) + f(b)
        print('x0 = %f'%f(a))
        for i in range(1,n):

                k = a + i*h

                y = f(k)

                if i%2 == 0:

                        sm = sm + 2 * y

                else:

                        sm = sm + 4 * y

                print('x%d = %f'%(i,y))


        y = f(b)

        print('x%d = %f'%(n,y))

        sm = sm * h /3

        print('value of integral is %.8f'%sm)

        return


f = lambda x : math.sin(x)

a = float(input("Enter lower limit : "))

b = float(input("Enter upper limit : "))

n = int(input("Enter Total no of subintraval : "))


simsonthirdMethod(f,a,b,n)
```

2.

```
def newtonRaphson(f,df,x,n):

        h = f(x) / df(x)
```

```
    for n in range(1,n+1):

        h = f(x) / df(x)

        x = x -h


    print("The value newton Raphson method : ","%.4f"%x)


f = lambda x : x**2+ 5*x +1

df = lambda x : 5*x**4+5*x+1

x0 = -1

newtonRaphson(f,df,x0,5)
```

b).

```
def falsepositionmethod(a,b,e):

    if f(a) * f(b) >=0:

        print("False positionmethod false ")

        print("enterval cheak ")

    else:

        step    = 1

        print("** false position method emplimention ** ")

        condition = True

        while condition:

            c = a-(b-a)* f(a)/(f(a) - f(b))

            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))
            if f(a) * f(c) <0:

                b = c

            else:

                a = c

            step = step +1

            condition = abs(f(c)) >e

            print("\n required root is : %.8f"%c)

f = lambda x : x**2 -2*x-1

a = float(input("Enter lower limit : "))

b = float(input("Enter upper limit : "))

e =    float(input("Tolebrable error : "))


falsepositionmethod(a,b,e)
```

2).

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

    h = (b-a)/2

    sm = f(a) + f(b)


    for n in range(1,n):

        k = a+ i*h
        y = f(k)
        sm = sm + 2*y
        print('x %d = %f'%(i,y))

    sm = sm * h/2
    print('\n value of integral is %0.8f'%sm)
    return

f = lambda x : 1/(1+x)
a = float(input("Enter lower limit : "))
b = float(input("Enter upper limit : "))
n = int(input("Enter subintaral num     : "))

trap(f,a,b,n)
```

slip 18

Q.1)

1.

```
num = (16,3,5,48,2,44,5,6,78,12,5,6,24)

min(num)
```

2.

```
from math import sqrt

print("Input lengths of shorter triangle sides:")

a = 12

b = 5

c = sqrt(a*2 + b*2)

print("The length of the hypotenuse is:",c)
```

3.

```
num = 125312.3142

print(int(num))
```

Q.2

1.

```
from sympy import *
```

```
A = Matrix([[2,4],[4,3]])
B = Matrix([[4,3],[5,3]])
print(A + B)
print(A.T)
print(A**-1)
```

2.

```
n = 20
sm = 0
while n > 0:
    sm = sm + n
    n = n-1
print(sm)
```

3.

```
from sympy import *
A = Matrix([[3,-2],[6,-4]])
P,D =A.diagonalize()
P
D
```

Q.3

a)

1.

```
import math
def simsonthirdMethod(f,a,b,n):
    h = (b-a)/2
    sm = f(a) + f(b)
    print('x0 = %f'%f(a))

    for i in range(1,n):
        k = a + i*h
        y = f(k)
        if i%2 == 0:
            sm = sm + 2 * y
        else:
            sm = sm + 4 * y
        print('x%d = %f'%(i,y))

    y = f(b)
    print('x%d = %f'%(n,y))
    sm = sm * h /3
    print('value of integral is %.8f'%sm)
    return
```

```
f = lambda x : 1/x


simsonthirdMethod(f,1,3,8)
```

2.

```
from scipy.interpolate import interp1d
X = [1,2,3,4]
Y = [11,9,27,64]
interpolate_x = 2.9
y_interp = interp1d(X, Y)
print("Value of Y at x = {} is".format(interpolate_x),

            y_interp(interpolate_x))
```

b)

1.

```
def falsepositionmethod(a,b,e):
          if f(a) * f(b) >=0:
                    print("False positionmethod false ")
                    print("enterval cheak ")
```

```
    else:
        step    = 1
        print("** false position method emplimention ** ")
        condition = True
        while condition:
            c = a-(b-a)* f(a)/(f(a) - f(b))
            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))
            if f(a) * f(c) <0:
                b = c
            else:
                a = c
            step = step +1
            condition = abs(f(c)) >e
            print("\n required root is : %.8f"%c)
f = lambda x : x**3 - 5*x -9
falsepositionmethod(2,3,0.00001)
```

2.

```
import math
def trap(f,a,b,n):
    h = (b-a)/2
    sm = f(a) + f(b)
    for i in range(1,n):
        k = a+ i*h
        y = f(k)
        sm = sm + 2*y
        print('x %d = %f'%(i,y))

    sm = sm * h/2
    print('\n value of integral is %0.8f'%sm)
    return

f = lambda x : math.cos(x)
trap(f,0,1,5)
```

SLIP 19

Q.1

1.

```
for i in range(2,11):

        for j in range(1,11):

                print(i*j)
```

2.

```
x=int(input(" enter a number "))

if x == 0:

        print("zero")

elif x % 2 == 0:

        print(" is even ")

else:

        print(" is odd ")
```

3.

```
num ={"swapni":29 , "chetan": 4,"dnyneswar": 24,"rush":13,"gaurav":19}

print(num)
```

Q.2.

1.

```
from sympy import *
import numpy as np
A = Matrix([[1,2,2],[2,1,2],[2,2,1]])
print(A.T)
print(np.linalg.inv(A))
```

2.

```
from sympy import *
A = Matrix([[5,2,5,4],[10,3,4,6],[2,0,-1,11]])
print(A.rank())
print(A.rref())
```

3.

```
import pprint
import scipy
import scipy.linalg      # SciPy Linear Algebra Library

A = scipy.array([ [7, 3, -1, 2], [3, 8, 1, -4], [-1, 1, 4, -1], [2, -4, -1, 6] ])
P, L, U = scipy.linalg.lu(A)
print("L:")
pprint.pprint(L)
```

```
print("U:")
pprint.pprint(U)
```

Q.3

A)

1.

```
import math
def simsonthirdMethod(f,a,b,n):
    h = (b-a)/2
    sm = f(a) + f(b)
    print('x0 = %f'%f(a))

    for i in range(1,n):
        k = a + i*h
        y = f(k)
        if i%2 == 0:
            sm = sm + 2 * y
        else:
            sm = sm + 4 * y
        print('x%d = %f'%(i,y))
    y = f(b)
    print('x%d = %f'%(n,y))
    sm = sm * h /3
    print('value of integral is %.8f'%sm)
    return

f = lambda x : 1/(1+x**2)
simsonthirdMethod(f,0,1,6)
```

2.

```
def falsepositionmethod(a,b,e):
    if f(a) * f(b) >=0:
        print("False positionmethod false ")
        print("enterval cheak ")
    else:
        step    = 1
        print("** false position method emplimention ** ")
        condition = True
        while condition:
            c = a-(b-a)* f(a)/(f(a) - f(b))
            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))
            if f(a) * f(c) <0:
                b = c
            else:
                a = c
            step = step +1
            condition = abs(f(c)) >e
            print("\n required root is : %.8f"%c)
f = lambda x : x**3 - 8*x - 4

falsepositionmethod(3,4,0.00001)
```

B)

1.

```
def trap(f,a,b,n):
    h = (b-a)/2
    sm = f(a) + f(b)

    for i in range(1,n):
        k = a+ i*h
        y = f(k)
        sm = sm + 2*y
        print('x %d = %f'%(i,y))
    sm = sm * h/2

    print('\n value of integral is %0.8f'%sm)

    return


f = lambda x : x**2

trap(f,0,1,5)
```

SLIP 20

Q.1

1.

```
n = int(input("Enter number : "))
for i in range(1,n):
        print(i**(1/2))
```

2.

```
sm =0
for i in range(1,21):
        sm = sm + (i*i)
print(sm)
```

3.

Q.3

1.

```
tuple =("I am Indain ")
print(tuple)
```

2.

3.

```
from sympy import *
A = Matrix([[3,-2],[6,-4]])
print(A.is_diagonalizable())
P,D = A.diagonalize()
P
D
```

Q.3

1.

```
import math
def simson3by8Rule(f,a,b,n):
    h = 0.5
    sm = f(a) + f(b)
    print("X0 = %.4f"%sm)

    for i in range(1,n):
        k = a + i*h
        y = f(k)
        if i%3 == 0:
            sm = sm + 2 *y
        else:
            sm = sm + 3 * y
        print("X%d = %.4f"%(i,y))
    y = f(b)
    print("X%d = %.4f"%(n,f(b)))
    sm = sm + h    * (3/8)
    print("Value of intergral    is : %.5f"%sm)

f = lambda x : math.cos(x)
simson3by8Rule(f,1,3,6)
```

2.

```
from scipy.interpolate import interp1d
X = [11,2,3,6]
Y = [2,6,12,42]
interpolate_x = 5
y_interp = interp1d(X, Y)
print("Value of Y at x = {} is".format(interpolate_x),
     y_interp(interpolate_x))
```

b).

1.

```
def falsepositionmethod(a,b,e):
    if f(a) * f(b) >=0:
        print("False positionmethod false ")
        print("enterval cheak ")
    else:
        step    = 1
        print("** false position method emplimention ** ")
        condition = True
        while condition:
            c = a-(b-a)* f(a)/(f(a) - f(b))
            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))
            if f(a) * f(c) <0:
                b = c
            else:
                a = c
            step = step +1
            condition = abs(f(c)) >e
            print("\n required root is : %.8f"%c)
f = lambda x : x**3 - 5*x -9
falsepositionmethod(2,3,0.00001)
```

2.

```
def trap(f,a,b,n):
    h = (b-a)/2
    sm = f(a) + f(b)

    for i in range(1,n):
        k = a+ i*h
        y = f(k)
        sm = sm + 2*y
        print('x %d = %f'%(i,y))

    sm = sm * h/2
    print('\n value of integral is %0.8f'%sm)
    return

f = lambda x : x**3 - 3*x +2
trap(f,1,3,5)
```

Slip 17

Q.1

1.

```
wonderful = "wonderful"

def fun():

    bad = "bad"

    print("Python is "+wonderful)

    print("Python is "+bad)


fun()
```

2.

```
from sympy import *

A = Matrix([[3,-2],[6,-4]])

print(A.is_diagonalizable())

P,D = A.diagonalize()

P

D
```

3.

```
import math

for a in range (1, 51):

    for b in range (1, 51):
```

```
    c = a*2 + b*2

    if math.sqrt(c) <= 50:

        print("a = %d, b = %d, c = %.4f"%(a,b,math.sqrt(c)))
```

Q.2

1.

```
Import numpy as np

A = np.array([[2,3],[1,-4]])

B = np.array([[1,-2],[-1,3]])

a = np.linalg.inv(A)

b = np.linalg.inv(B)

ab = np.linalg.inv(A*B)


print(ab == b*a)
```

2.

```
from sympy import *

x,y,z=symbols(‘x,y,z’)

A=Matrix([[1,-2,3],[2,1,1],[-3,2,-2]])

B=Matrix([7,4,10])
```

```
linsolve((A,B), [x,y,z])
```

3.

```
Import numpy as np

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

A.trace()

A.transpose()
```

Q.3

a)

1.

```
import numpy as np


# Reading number of unknowns

n = int(input('Enter number of data points: '))

x = np.zeros((n))

y = np.zeros((n))

print('Enter data for x and y: ')

for i in range(n):

        x[i] = float(input( 'x['+str(i)+']='))
        y[i] = float(input( 'y['+str(i)+']='))
xp = float(input('Enter interpolation point: '))
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('Interpolated value at %.3f is %.3f.' % (xp, yp))
```

2.

```
def newtonRaphson(f,df,x,n):
        h = f(x) / df(x)
        for n in range(1,n+1):
                h = f(x) / df(x)
                x = x -h
                print("x : ",(n,x))
        print("The value newton Raphson method : ","%.4f"%x)
```

```
f = lambda x : x**5+ 5*x + 6

df = lambda x : 5*x**4 - 5

x0 = 1

newtonRaphson(f,df,2,5)
```

b)

1.

```
def falsepositionmethod(a,b,e):

    if f(a) * f(b) >=0:

        print("False positionmethod false ")

        print("enterval cheak ")

    else:

        step    = 1

        print("** false position method emplimention ** ")

        condition = True

        while condition:

            c = a-(b-a)* f(a)/(f(a) - f(b))

            print('Iteration-%d, c = %0.6f and f(c) = %0.6f' % (step, c, f(c)))
            if f(a) * f(c) <0:

                b = c

            else:

                a = c

            step = step +1

            condition = abs(f(c)) >e

            print("\n required root is : %.8f"%c)

f = lambda x : x**2 - 2 * x -1


falsepositionmethod(1,3,0.0001)
```

2.

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

    h = (b-a)/2

    sm = f(a) + f(b)


    for i in range(1,n):

        k = a+ i*h

        y = f(k)

        sm = sm + 2*y
        print('x %d = %f'%(i,y))


    sm = sm * h/2

    print('\n value of integral is %0.8f'%sm)

    return


f = lambda x : x**2

trap(f,0,1,10)
```