Numerical Methods for Data Science - (SC602)

Assignment -1

1. Write a Program to find the Square Root of a given number N using Babylonian square root method (N=299,225)

Suppose you are given any positive number S. To find the square root of S, do the following:

Make an initial guess. Guess any positive number x0.

Improve the guess. Apply the formula x1 = (x0 + S / x0) / 2. The number x1 is a better approximation to sqrt(S).

Iterate until convergence. Apply the formula xn+1 = (xn + S / xn) / 2 until the process converges. Convergence is achieved when the digits of xn+1 and xn agree to as many decimal places as you desire.

```
import random
def babylonian(S):
  e=0.00001 #error
 # x1=x=S   # Taking random value as the S number
  x1=x= random.randint(1,S)
  y=1
  while x-y >e:
    x=x1
    y=(x+(S/x))/2
    x1=y
  return y
```

```
babylonian(299)
```

```
17.291616465790582
```

```
babylonian(225)
```

```
15.0
```

2. Take Two points from the user and plot the line using linear interpolation method.

We can use the Linear Interpolation method here.

1. Find the two adjacent (x1, y1) ,(x2,y2) from the x. i.e. (5,2.2360) and (6,2.4494).

Where x1 = 5, x2= 6, y1 = 2.2360, y2 = 2.4494, and we interpolate at point x = 5.5.

2. Using the formula y(x) = y1 + (x – x1) \frac{(y2 – y1) }{ (x2 – x1)}

3. After putting the values in the above equation.

y = 2.2360+(5.5-5)\frac{(2.4494-2.2360)}{(6 - 5)} y = 2.3427 At x = 5.5 the value of Y will be 2.3427. So by using linear interpolation we can easily determine the value of a function between two intervals.

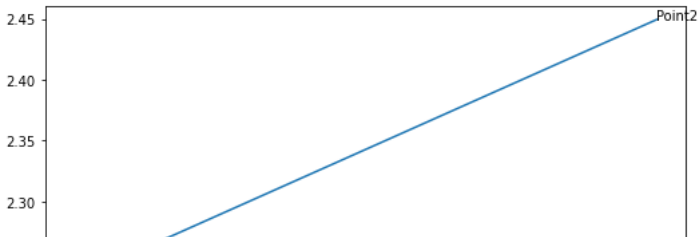
```
def interpolation(point1,point2,x):
  y=point1[1] +(x - point1[0])* ((point2[1] - point1[1])/(point2[0] - point1[0]))
  return y

point1 = list(map(float,input("Enter points x1 , y1 :").strip().split()))
point2 = list(map(float,input("Enter points x2 , y2 :").strip().split()))
#p1 =[5,2.2360]
#p2=[6,2.4494]
#x=5.5

x=float(input("Enter interpolating point : "))
print("Interpolation at point x is : ",interpolation(point1,point2,x) )

# Plotting graph
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = [7.50, 3.50]
plt.rcParams["figure.autolayout"] = True
x_values = [point1[0], point2[0]]
y_values = [point1[1], point2[1]]
plt.plot(x_values, y_values, linestyle="-")
plt.text(point1[0], point1[1], "Point1")
plt.text(point2[0], point2[1], "Point2")
```

```
Enter points x1 , y1 :5 2.2360
Enter points x2 , y2 :6 2.4494
Enter interpolating point : 5.5
Interpolation at point x is :  2.3427
Text(6.0, 2.4494, 'Point2')
```



3. Estimate the natural logarithm of 2 using linear interpolation. First, perform the computation by interpolating between ln 1 =0 and ln 6 = 1.791759. Then, repeat the procedure, but use a smaller interval from ln 1 to ln 4 (1.386294). Note that the true value of ln 2 is 0.6931472. Try for some different intervals and show errors for all intervals. Write your observation for the relationship between error and intervals.

```
#FINDING NATURAL LOAGARITHM OF 2 USING LINEAR INTERPOLATION
import math
def interpolate(interval,x):
      x1=interval[0]
      x2=interval[1]
      y1=math.log(interval[0])
      y2=math.log(interval[1])
      y= y1+(x - x1)*((y2 - y1)/(x2 - x1))
      return y
interval=list(map(int,input("Enter interval :").strip().split()))
x=2
approx=interpolate(interval,x)
print(approx)
#ORIGINAL VALUE OF LOG 2 IS 0.6931472
#ERROR % = ((ACTUAL VALUE - APPROX VALUE)/ACTUAL VALUE)*100
error = abs((math.log(2) - approx) / math.log(2)*100)
print("ERROR % = ",error)
```

```
Enter interval :1 6
0.358351893845611
ERROR % =  48.30074998557687
```

```
#FINDING NATURAL LOAGARITHM OF 2 USING LINEAR INTERPOLATION
#CHANGING INTERVAL VALUES
import math
def interpolate(interval,x):
      x1=interval[0]
      x2=interval[1]
      y1=math.log(interval[0])
      y2=math.log(interval[1])
      y= y1+(x - x1)*((y2 - y1)/(x2 - x1))
      return y
interval=list(map(int,input("Enter interval :").strip().split()))
x=2
approx=interpolate(interval,x)
print(approx)
#ORIGINAL VALUE OF LOG 2 IS 0.6931472
#ERROR % = ((ACTUAL VALUE - APPROX VALUE)/ACTUAL VALUE)*100
error = abs((math.log(2) - approx) / math.log(2)*100)
print("ERROR % = ",error)
```

```
Enter interval :1 4
0.46209812037329684
ERROR % =  33.333333333333336
```

```
1
```

```
1
```