Question 1

Finding the root using the Bisection method

In [ ]:

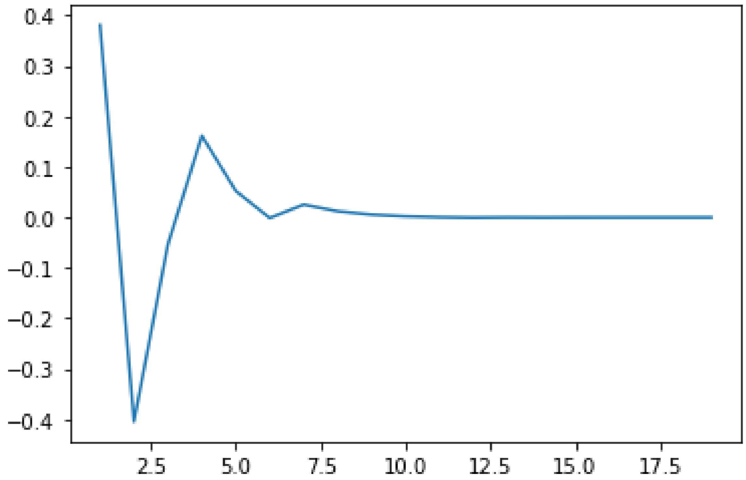
```
import math
from functionLibrary import rootBisec

y = lambda x: math.log(x/2) - math.sin(5*x/2) #defining the function

a = float(input("Enter first approx: "))
b = float(input("Enter second approx: "))

rootBisec(y, a, b)
```

```
Required root is:  2.623137969970703
```




```
x_i f(x_i)
1 0.3810133630172611
2 -0.4045966480895948
3 -0.05378489278189996
4 0.16175190955165575
5 0.052390109815500185
6 -0.001234323271522897
7 0.025460815432279432
8 0.01208182016791337
9 0.005415623412808823
10 0.0020885853075362504
11 0.00042661064337645715
12 -0.00040398693044757517
13 1.127926769850518e-05
14 -0.0001963619867327293
15 -9.254339733549832e-05
16 -4.063257414516075e-05
17 -1.4676780539124579e-05
18 -1.6987882464625237e-06
19 4.790231769524755e-06
```

Finding the root using the Regula Falsi method

In [ ]:

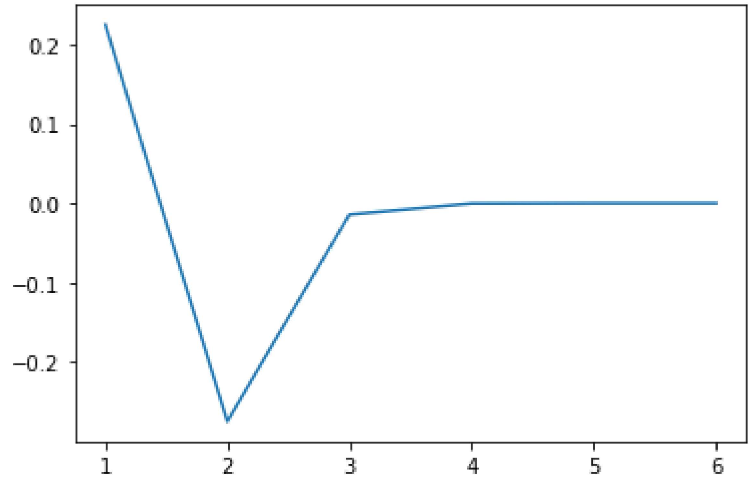
```
import math
from functionLibrary import rootRegFalsi

y = lambda x: math.log(x/2) - math.sin(5*x/2) #defining the function

a = float(input("Enter first approx: "))
b = float(input("Enter second approx: "))
```

```
rootRegFalsi(y, a, b)
```

```
Required root is:  2.623140463495963
```



```
x_i f(x_i)
1 0.22453691870116085
2 -0.2749867402642147
3 -0.01436413281618737
4 -0.0003869683540730362
5 -1.0022969852419017e-05
6 -2.59329427765298e-07
```

Question 2

Finding the root using the Bisection method

In [ ]:

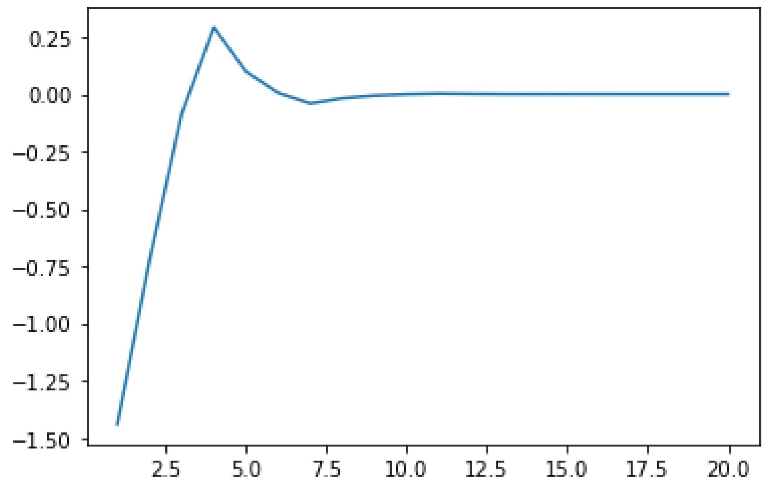
```
import math
from functionLibrary import rootBisec

y = lambda x: -x - math.cos(x) #defining the function

a = float(input("Enter first approx: "))
b = float(input("Enter second approx: "))

rootBisec(y, a, b)
```

```
Required root is:  -0.7390867996215817
```



```
x_i f(x_i)
1 -1.4404781165221552
2 -0.723894283619651
3 -0.08615215307489243
```

```
4 0.2917824111724301
5 0.09857319830990496
6 0.005090969327159245
7 -0.04081750155742625
8 -0.01793413260528698
9 -0.006439189164174963
10 -0.0006784980708837152
11 0.0022051403118625856
12 0.000763047075939105
13 4.2205964477126656e-05
14 -0.00031816319107869084
15 -0.00013798289734923141
16 -4.788953739565116e-05
17 -2.8420541925422427e-06
18 1.968188820977712e-05
19 8.419900275447034e-06
20 2.788918858076528e-06
```

Finding the root using the Regula Falsi method

In [ ]:

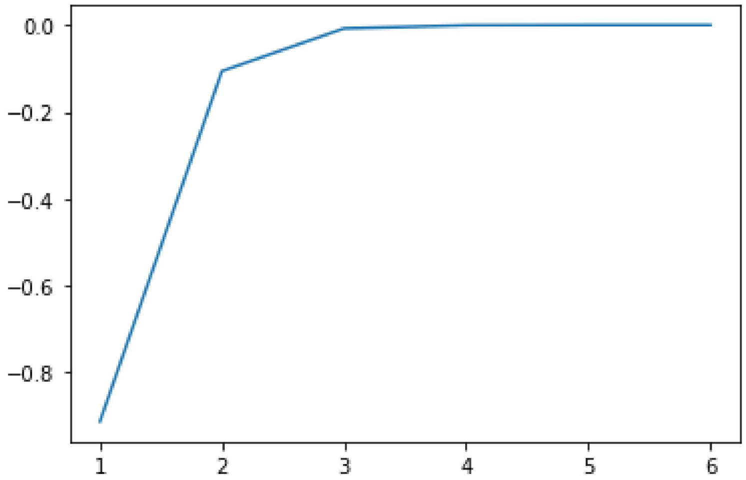
```python
import math
from functionLibrary import rootRegFalsi

y = lambda x: -x - math.cos(x) #defining the function

a = float(input("Enter first approx: "))
b = float(input("Enter second approx: "))

rootRegFalsi(y, a, b)
```

```
Required root is:  -0.7390835727489464
```




```
x_i f(x_i)
1 -0.9133350840659392
2 -0.10594735771282937
3 -0.007739322003794458
4 -0.0005400437849095718
5 -3.7559435059675295e-05
6 -2.611614127401296e-06
```

Finding the root using the Newton-Raphson method with initial guess x = 0.0

In [ ]:

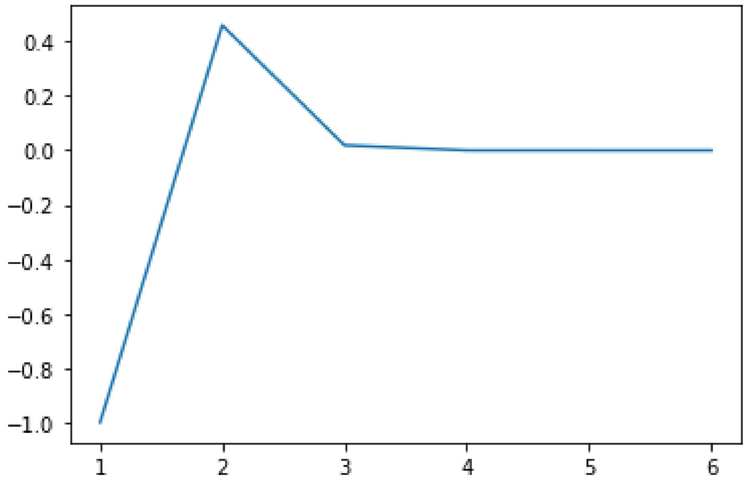
```python
import math
import matplotlib.pyplot as plt
from functionLibrary import newRaph


h = 0.00001 #defining the h that will be used to calculate first derivative of y usi
```

```
x0 = 0.0 #dummy variable for root


y = lambda x: -x - math.cos(x) #defining the function
dy = lambda x: (y(x+h)-y(x-h))/(2*h) #defining the first derivative of the function

x, n = newRaph(y, dy, x0, h, 100)
print("The root is %f at %d iterations." %(x, n)) #printing the results
```




```
x_i f(x_i)
1 -1.0
2 0.4596976941300186
3 0.018923073819868463
4 4.645589882956713e-05
5 2.8472002533419527e-10
6 0.0
The root is -0.739085 at 5 iterations.
```

Question 3

In [ ]:

```
import math
from functionLibrary import polyRoots

guess = float(input("Enter initial guess: "))
a = [4.0, 0.0, -5.0, 0.0, 1.0] #the co-efficients of the polynomial are fed to the f
                               #in this way. the lowest power (the constant) is writ

print("The required roots are: ") #printing the accompanying result statement
polyRoots(guess, a)
```

```
The required roots are:
```

Out[ ]:

```
[2.0, 1.0, -1.0, -2.0]
```