Question 1

Finding the root using the Bisection method

In [ ]:

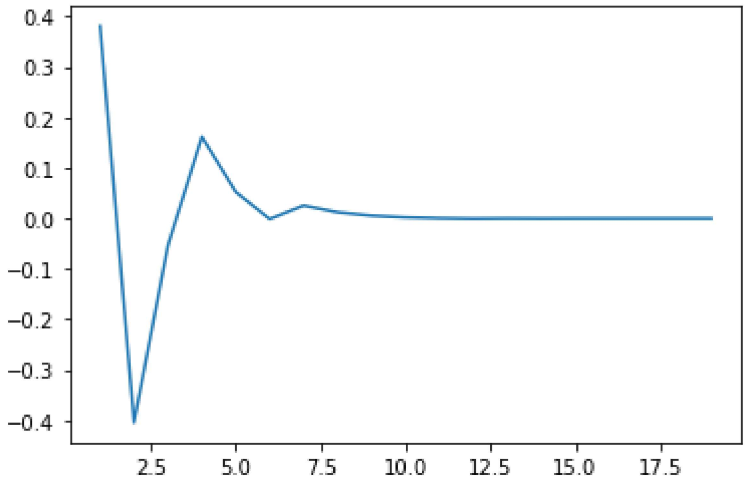
```
import math
from functionLibrary import rootBisec

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
```




```
i x_i
1 2.44
2 2.86
3 2.65
4 2.545
5 2.5975
6 2.6237500000000002
7 2.610625
8 2.6171875
9 2.62046875
10 2.622109375
11 2.6229296875
12 2.62333984375
13 2.623134765625
14 2.6232373046875
15 2.62318603515625
16 2.623160400390625
17 2.6231475830078126
18 2.623141174316406
19 2.623137969970703
```

Finding the root using the Regula Falsi method

In [ ]:

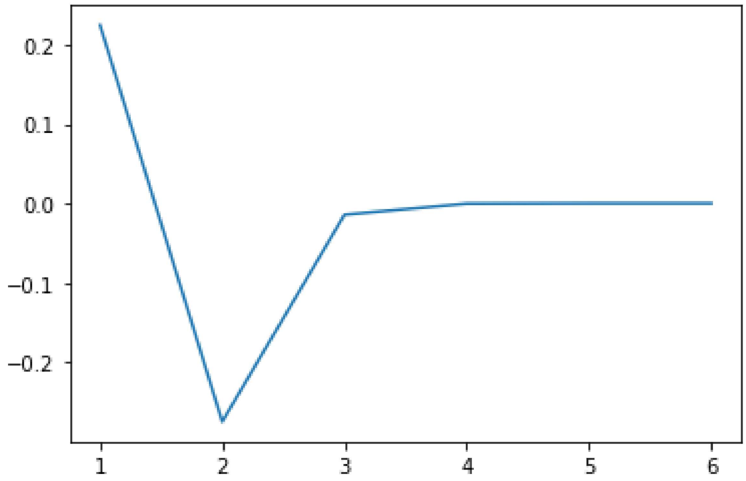
```
import math
from functionLibrary import rootRegFalsi

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 2.515130340871582
2 2.771242544732205
3 2.630253306650094
4 2.6233314385371953
5 2.6231452848955312
6 2.623140463495963
```

Question 2

Finding the root using the Bisection method

In [ ]:

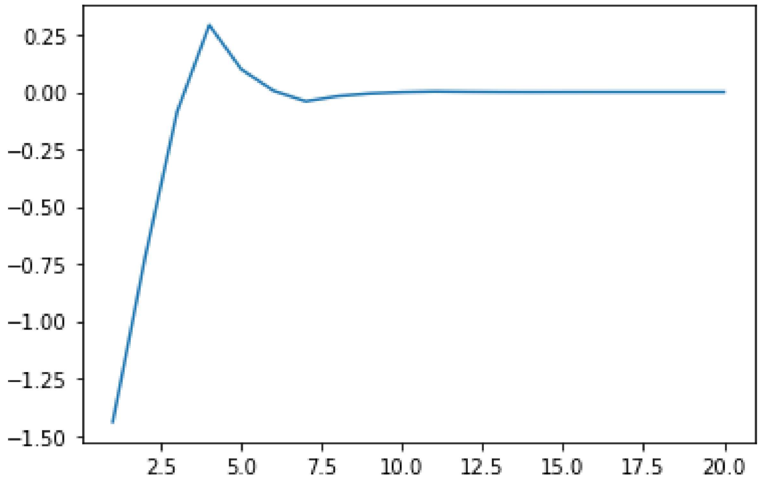
```
import math
from functionLibrary import rootBisec

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
```




```
i x_i
1 0.6360000000000001
2 -0.24599999999999977
3 -0.6869999999999997
```

```
4 -0.9074999999999998
5 -0.7972499999999998
6 -0.7421249999999997
7 -0.7145624999999998
8 -0.7283437499999997
9 -0.7352343749999997
10 -0.7386796874999997
11 -0.7404023437499997
12 -0.7395410156249997
13 -0.7391103515624997
14 -0.7388950195312497
15 -0.7390026855468748
16 -0.7390565185546872
17 -0.7390834350585935
18 -0.7390968933105466
19 -0.7390901641845701
20 -0.7390867996215817
```

Finding the root using the Regula Falsi method

In [ ]:

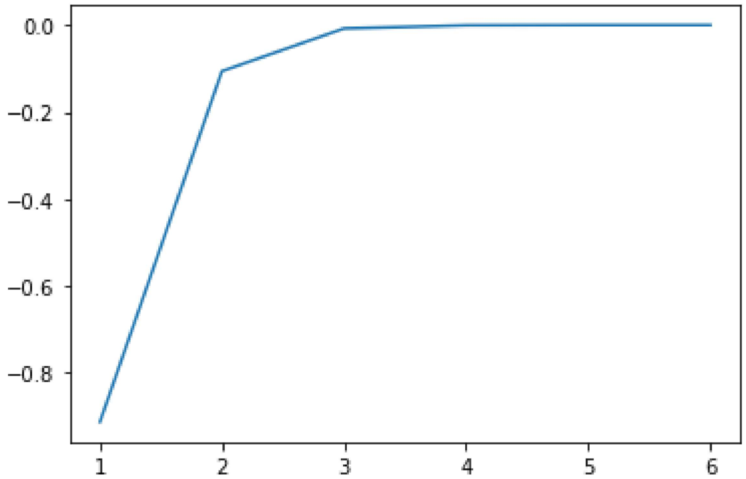
```
import math
from functionLibrary import rootRegFalsi

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 -0.08320534760978582
2 -0.6748520428807449
3 -0.7344560724626938
4 -0.7387624286320315
5 -0.7390626909641922
6 -0.7390835727489464
```

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

In [ ]:

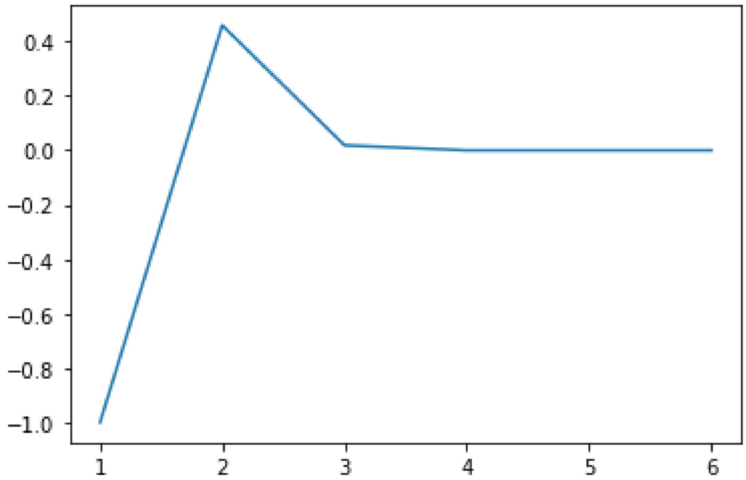
```
import math
import matplotlib.pyplot as plt
from functionLibrary import newRaph


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 -0.9999999999989999
2 -0.7503638678389067
3 -0.7391128909112653
4 -0.7390851333852837
5 -0.7390851332151607
The root is -0.739085 at 4 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]
```