Computational and Numerical Methods Lab - 2


```
Abhimanyu Karia - 202201435
Devarshi Patel - 202201447
```


Bisection Method

Algorithm of bisection method:

1. Select 2 initial points a,b such that f(a)*f(b)<0.
2. c=(a+b)/2, that is the midpoint of a and b.
3. if f(c)<e for some suitable number e, then we will stop the iteration and output c. In our case e=0.00001.
4. else, if f(a)*f(c)<0, we will change value of b to equal c.
5. else, if f(b)*f(c)<0, we will change value of a to equal c. Continue to iterate till we do not get the output.

In [ ]:
```
import math as mt
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:
```
err = 0.0001
```

Q - 1

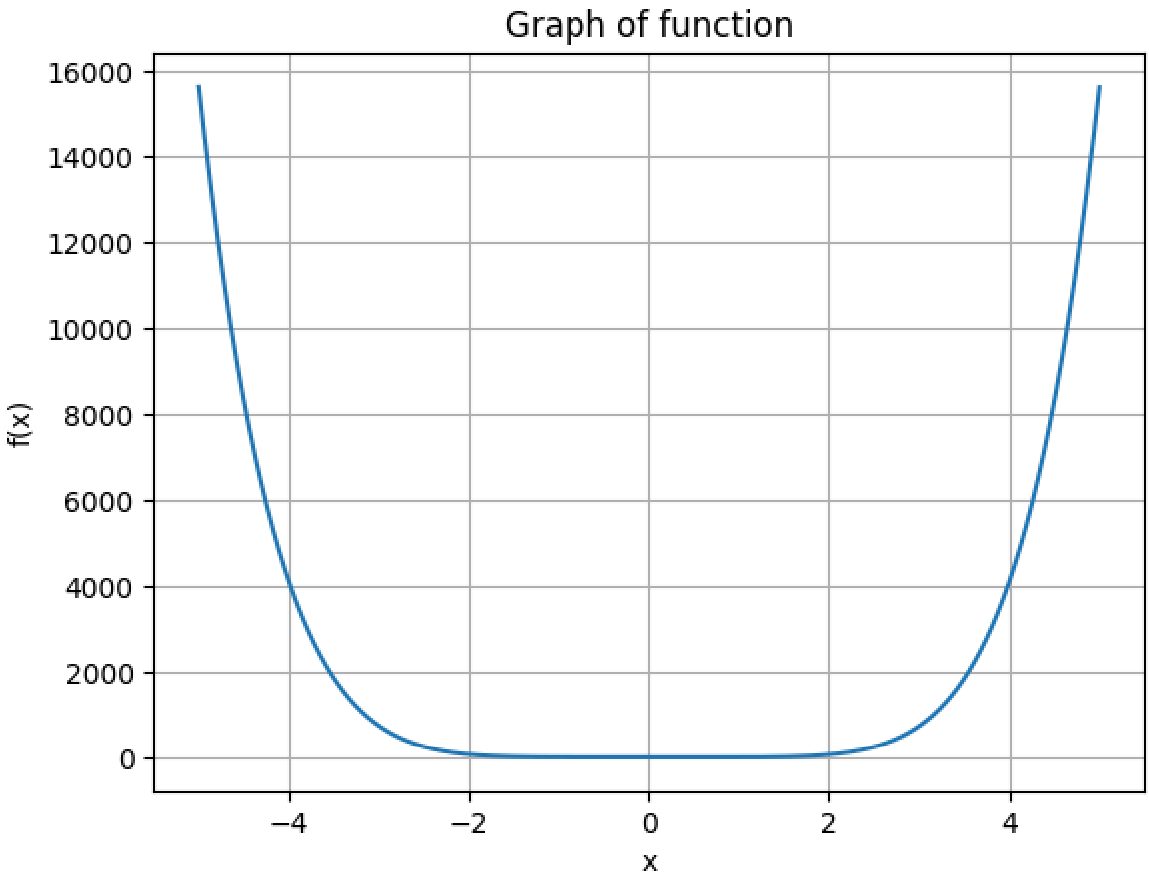
In [ ]:
```
def maxiter(a0,b0,err):
    return mt.ceil(np.log2((b0-a0)/err) - 1)

def fun(x):
    return x**6 - x - 1

def bisection_get_roots(a,b,err):
    n = maxiter(a,b,err)
    roots = []
    data = []
    error = []
    x_range = np.arange(-5,5+err,err)
    for i in range(n):
        x = (a+b)/2
        roots.append(x)
        fa = fun(a)
        fb = fun(b)
        fx = fun(x)
        temp = [i+1,a,b,x,b-x,fx]
        data.append(temp)
```

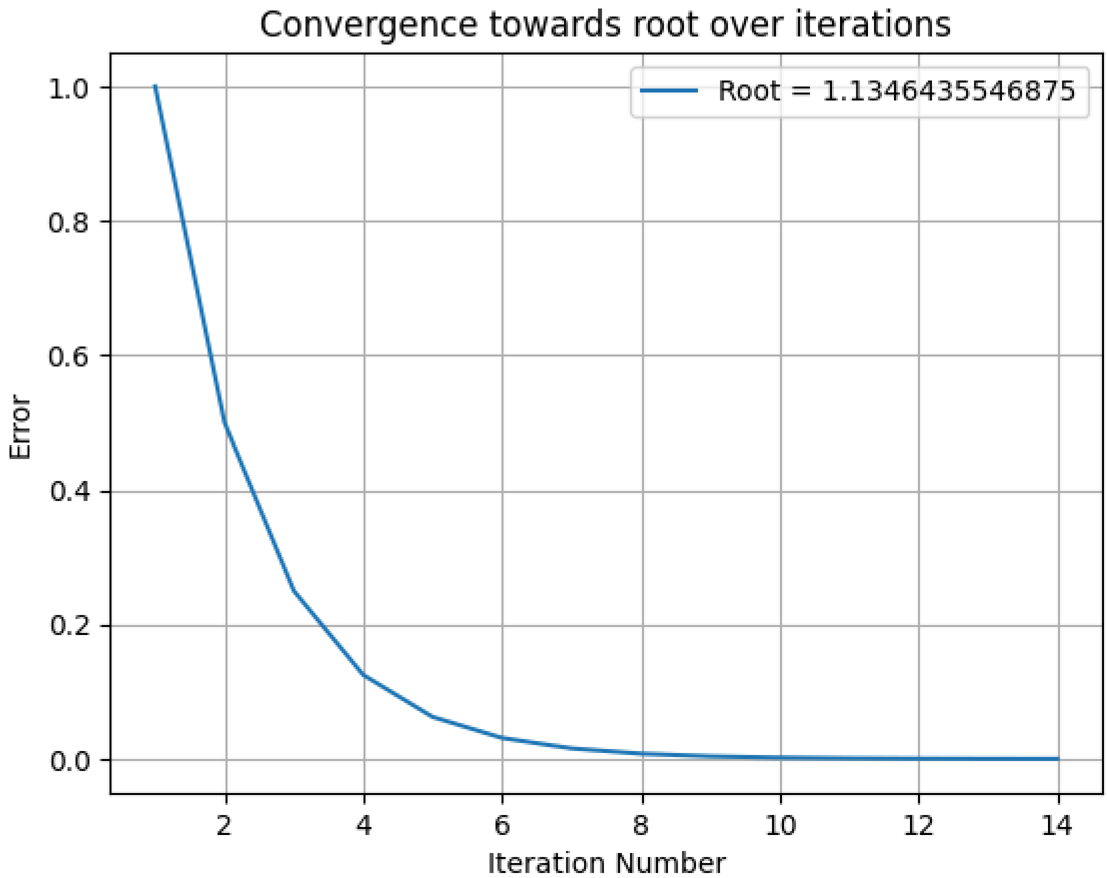
```python
        error.append(b-x)
        if fa*fx == 0 or fb*fx == 0:
            break
        elif fa*fx < 0 :
            b = x
        else:
            a = x
    df = pd.DataFrame(data,columns=['Iteration','an','bn','c','b-c','f(c)'])
    print('root is :',roots[-1])
    roots = np.array(roots)
    iter = np.arange(1,n+1,1)
    plt.figure(1)
    plt.plot(x_range,fun(x_range))
    plt.title("Graph of function")
    plt.xlabel('x')
    plt.ylabel('f(x)')
    plt.grid(True)
    plt.show()
    plt.figure(2)
    plt.plot(iter,error,label = 'Root = ' + str(roots[-1]))
    plt.title("Convergence towards root over iterations")
    plt.xlabel('Iteration Number')
    plt.legend()
    plt.ylabel('Error')
    plt.grid(True)
    plt.show()
    return df
```

$$x^6 - x - 1$$

In [ ]:
```python
df = bisection_get_roots(0,2,err)
df
```

```
root is : 1.1346435546875
```


Graph of function
16000
14000
12000
10000
8000
6000
4000
2000
0
f(x)
−4
−2
0
2
4
x


Convergence towards root over iterations
Root = 1.1346435546875
1.0
0.8
0.6
0.4
0.2
0.0
Error
2
4
6
8
10
12
14
Iteration Number

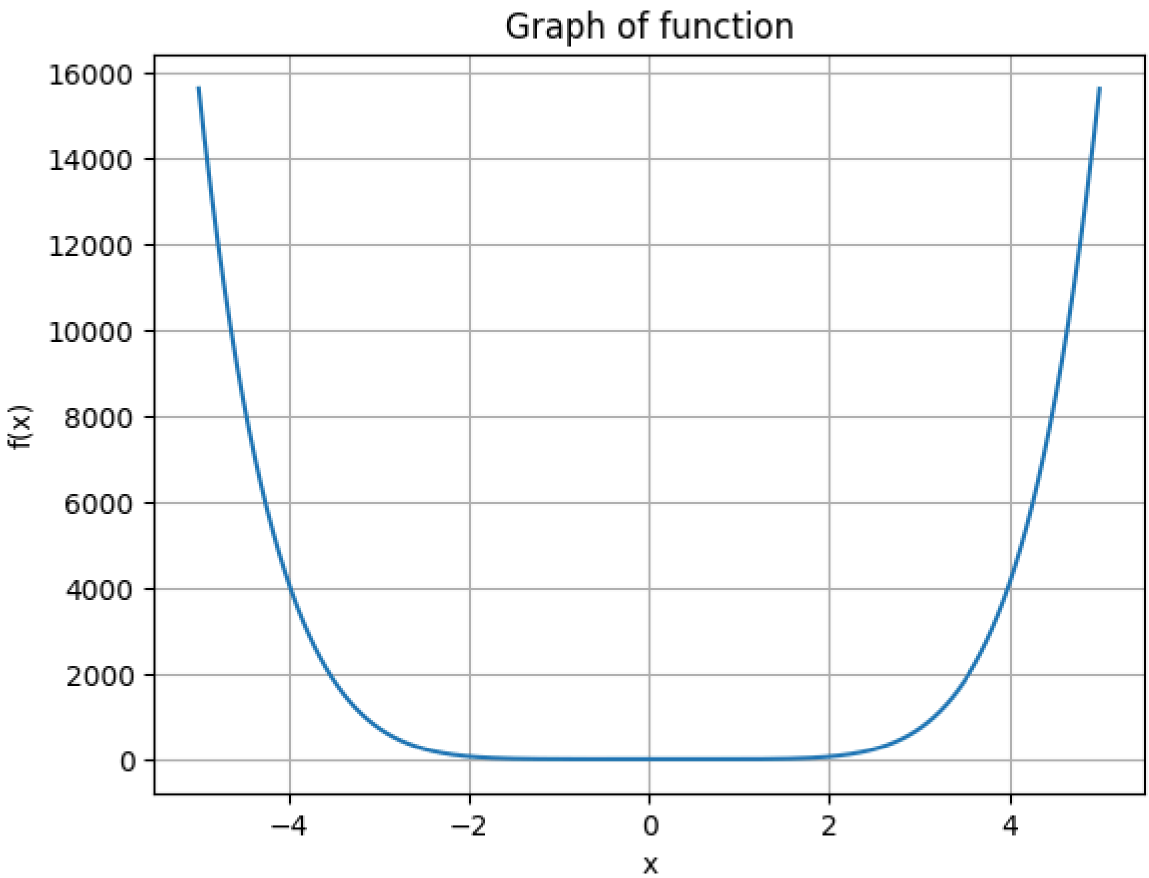
Out[ ]:

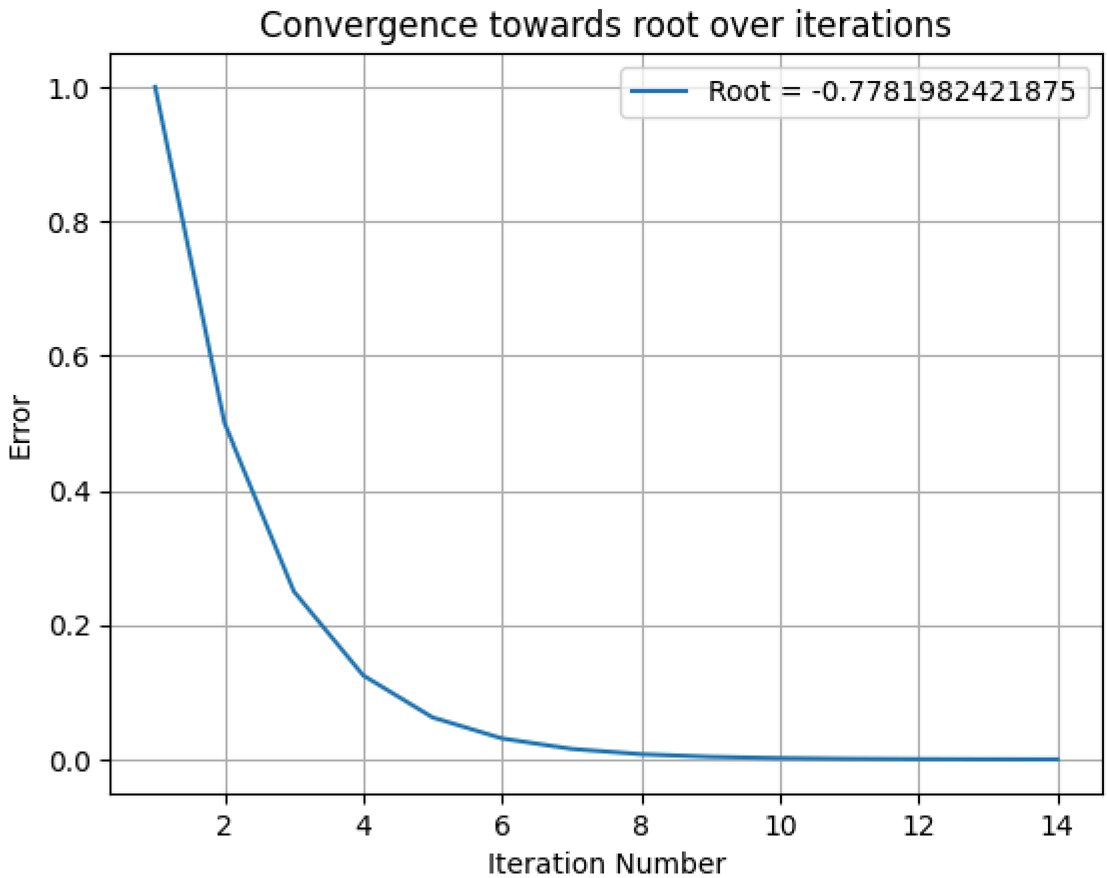
	Iteration	an	bn	c	b-c	f(c)
0	1	0.000000	2.000000	1.000000	1.000000	-1.000000
1	2	1.000000	2.000000	1.500000	0.500000	8.890625
2	3	1.000000	1.500000	1.250000	0.250000	1.564697
3	4	1.000000	1.250000	1.125000	0.125000	-0.097713
4	5	1.125000	1.250000	1.187500	0.062500	0.616653
5	6	1.125000	1.187500	1.156250	0.031250	0.233269
6	7	1.125000	1.156250	1.140625	0.015625	0.061578
7	8	1.125000	1.140625	1.132812	0.007812	-0.019576
8	9	1.132812	1.140625	1.136719	0.003906	0.020619
9	10	1.132812	1.136719	1.134766	0.001953	0.000427
10	11	1.132812	1.134766	1.133789	0.000977	-0.009598
11	12	1.133789	1.134766	1.134277	0.000488	-0.004591
12	13	1.134277	1.134766	1.134521	0.000244	-0.002084
13	14	1.134521	1.134766	1.134644	0.000122	-0.000829

In [ ]:

```
df = bisection_get_roots(-2,0,err)
df
```

```
root is : -0.7781982421875
```





Out[ ]:

	Iteration	an	bn	c	b-c	f(c)
0	1	-2.000000	0.000000	-1.000000	1.000000	1.000000
1	2	-1.000000	0.000000	-0.500000	0.500000	-0.484375
2	3	-1.000000	-0.500000	-0.750000	0.250000	-0.072021
3	4	-1.000000	-0.750000	-0.875000	0.125000	0.323795
4	5	-0.875000	-0.750000	-0.812500	0.062500	0.100200
5	6	-0.812500	-0.750000	-0.781250	0.031250	0.008624
6	7	-0.781250	-0.750000	-0.765625	0.015625	-0.032958
7	8	-0.781250	-0.765625	-0.773438	0.007812	-0.012495
8	9	-0.781250	-0.773438	-0.777344	0.003906	-0.002019
9	10	-0.781250	-0.777344	-0.779297	0.001953	0.003281
10	11	-0.779297	-0.777344	-0.778320	0.000977	0.000626
11	12	-0.778320	-0.777344	-0.777832	0.000488	-0.000698
12	13	-0.778320	-0.777832	-0.778076	0.000244	-0.000036
13	14	-0.778320	-0.778076	-0.778198	0.000122	0.000295

Results:

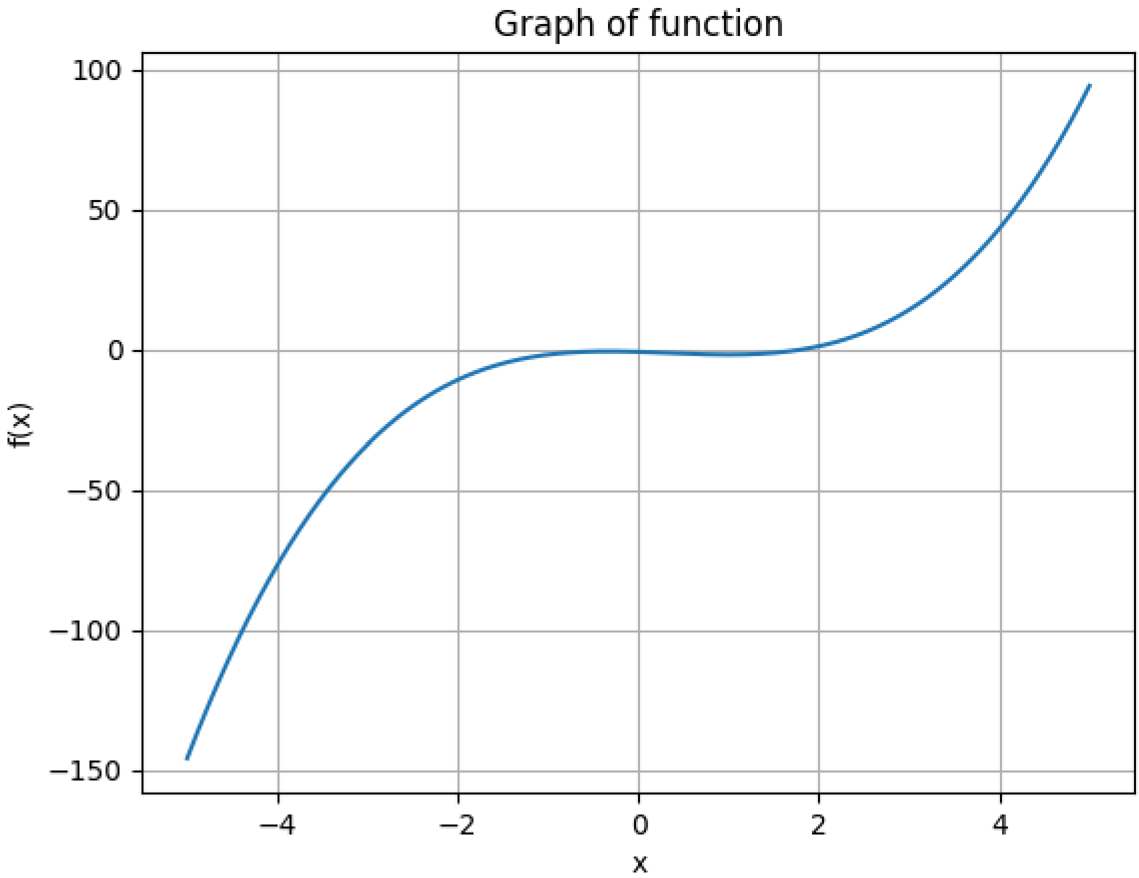
1. The root of function for the initial points 0 and 2 is 1.1346435546875
2. The root of function for the initial points -2 and 0 is -0.7781982421875

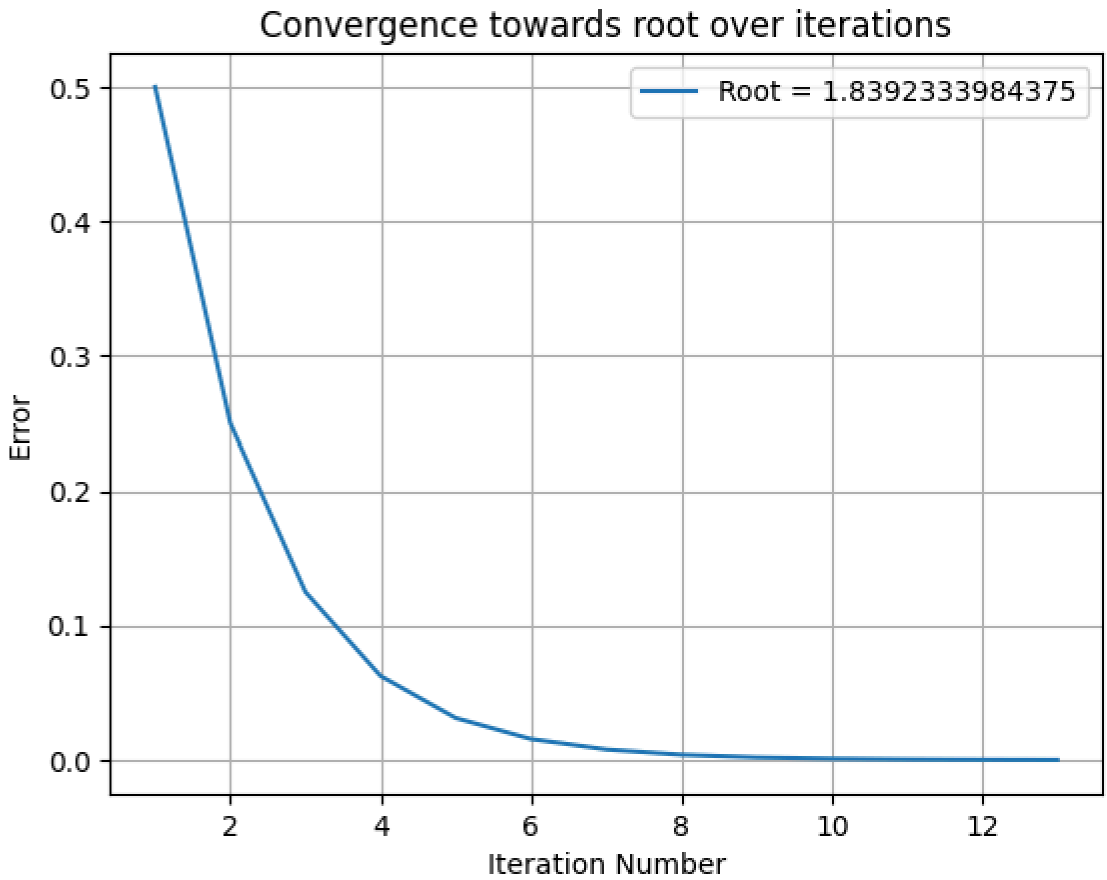
Q - 2

$$x^3 - x^2 - x - 1$$

In [ ]:
```
def fun(x):
    return x**3 - x**2 - x - 1
df = bisection_get_roots(1,2,err)
df
```

```
root is : 1.8392333984375
```





Out[ ]:

	Iteration	an	bn	c	b-c	f(c)
0	1	1.000000	2.000000	1.500000	0.500000	-1.375000
1	2	1.500000	2.000000	1.750000	0.250000	-0.453125
2	3	1.750000	2.000000	1.875000	0.125000	0.201172
3	4	1.750000	1.875000	1.812500	0.062500	-0.143311
4	5	1.812500	1.875000	1.843750	0.031250	0.024506
5	6	1.812500	1.843750	1.828125	0.015625	-0.060497
6	7	1.828125	1.843750	1.835938	0.007812	-0.018271
7	8	1.835938	1.843750	1.839844	0.003906	0.003048
8	9	1.835938	1.839844	1.837891	0.001953	-0.007629
9	10	1.837891	1.839844	1.838867	0.000977	-0.002294
10	11	1.838867	1.839844	1.839355	0.000488	0.000376
11	12	1.838867	1.839355	1.839111	0.000244	-0.000960
12	13	1.839111	1.839355	1.839233	0.000122	-0.000292

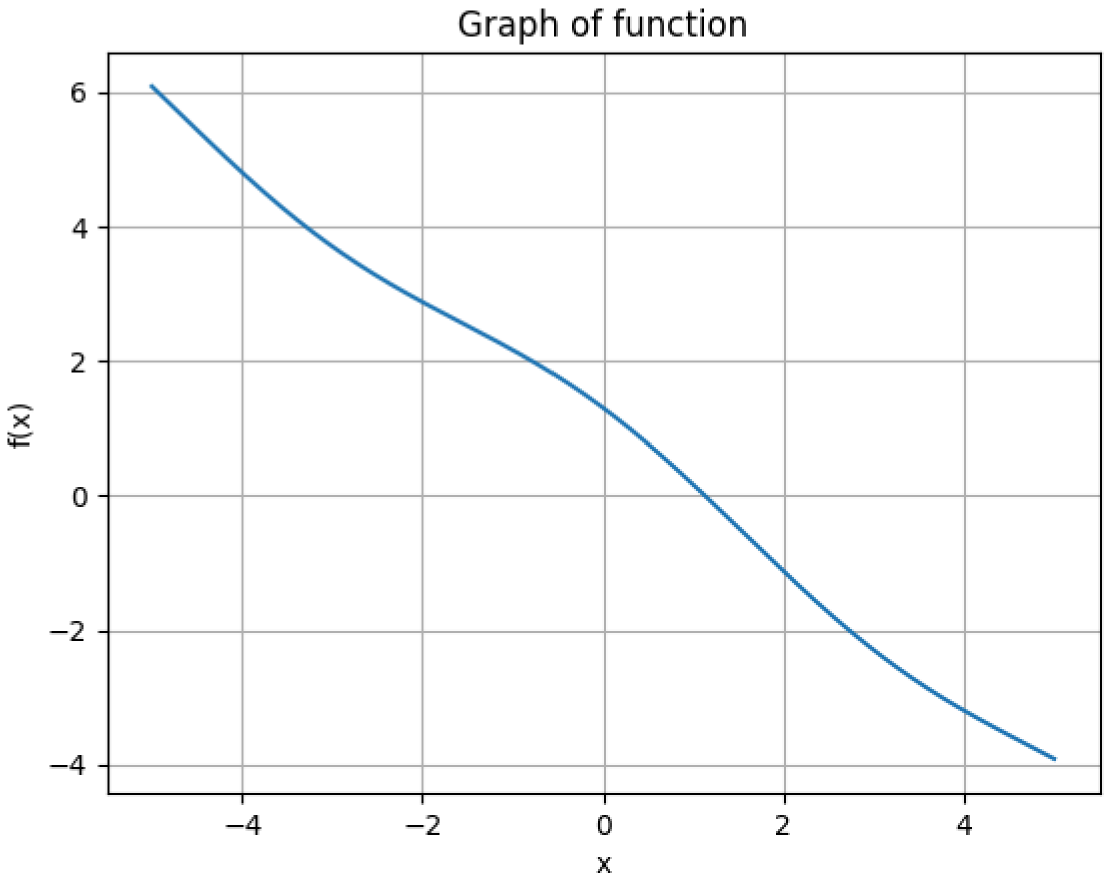
Result:

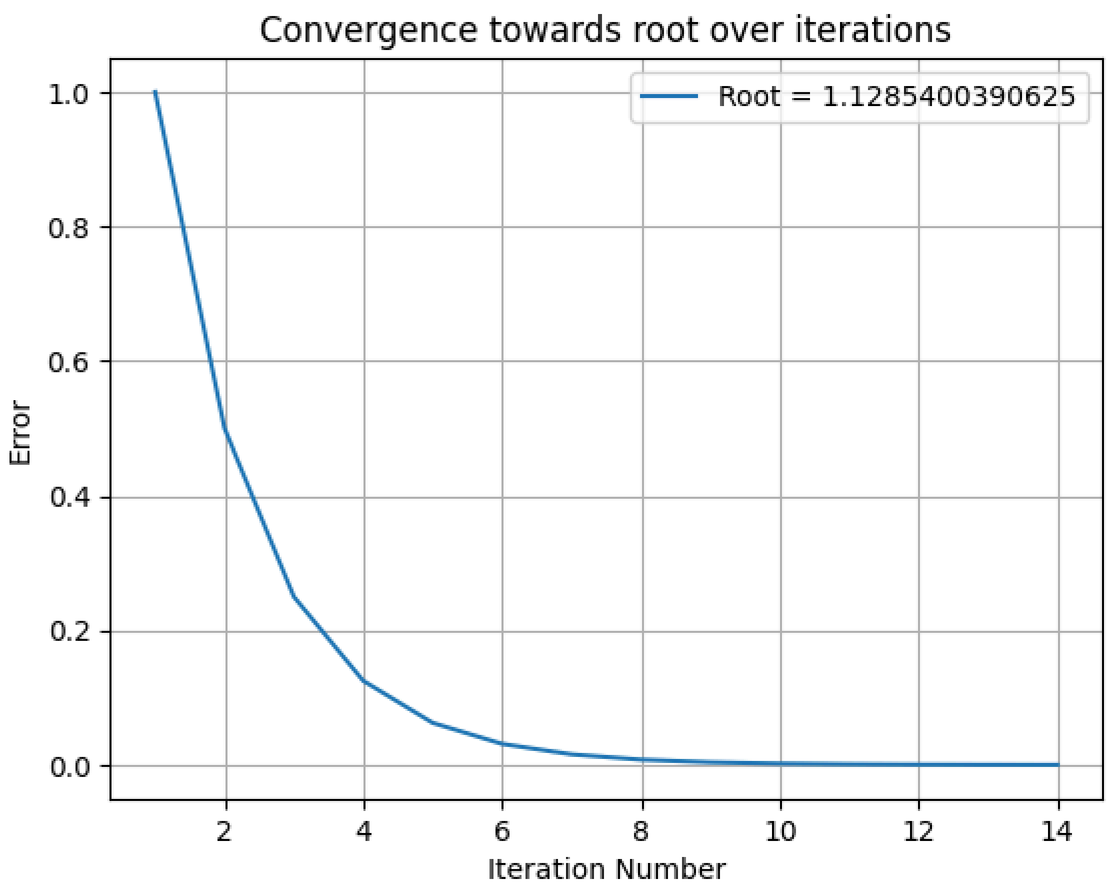
1. The root of the function is 1.1346435546875 for the initial points 1 and 2.
2. We can see the decrease in error with every iteration which shows convergence towards the root.

$$x = 1 + 0.3 * cos(x)$$

In [ ]:
```
def fun(x):
    return 1 + 0.3*np.cos(x) - x
df = bisection_get_roots(0,2,err)
df
```

```
root is : 1.1285400390625
```





Out[ ]:

	Iteration	an	bn	c	b-c	f(c)
0	1	0.000000	2.000000	1.000000	1.000000	0.162091
1	2	1.000000	2.000000	1.500000	0.500000	-0.478779
2	3	1.000000	1.500000	1.250000	0.250000	-0.155403
3	4	1.000000	1.250000	1.125000	0.125000	0.004353
4	5	1.125000	1.250000	1.187500	0.062500	-0.075306
5	6	1.125000	1.187500	1.156250	0.031250	-0.035418
6	7	1.125000	1.156250	1.140625	0.015625	-0.015517
7	8	1.125000	1.140625	1.132812	0.007812	-0.005578
8	9	1.125000	1.132812	1.128906	0.003906	-0.000612
9	10	1.125000	1.128906	1.126953	0.001953	0.001871
10	11	1.126953	1.128906	1.127930	0.000977	0.000630
11	12	1.127930	1.128906	1.128418	0.000488	0.000009
12	13	1.128418	1.128906	1.128662	0.000244	-0.000301
13	14	1.128418	1.128662	1.128540	0.000122	-0.000146

Result:

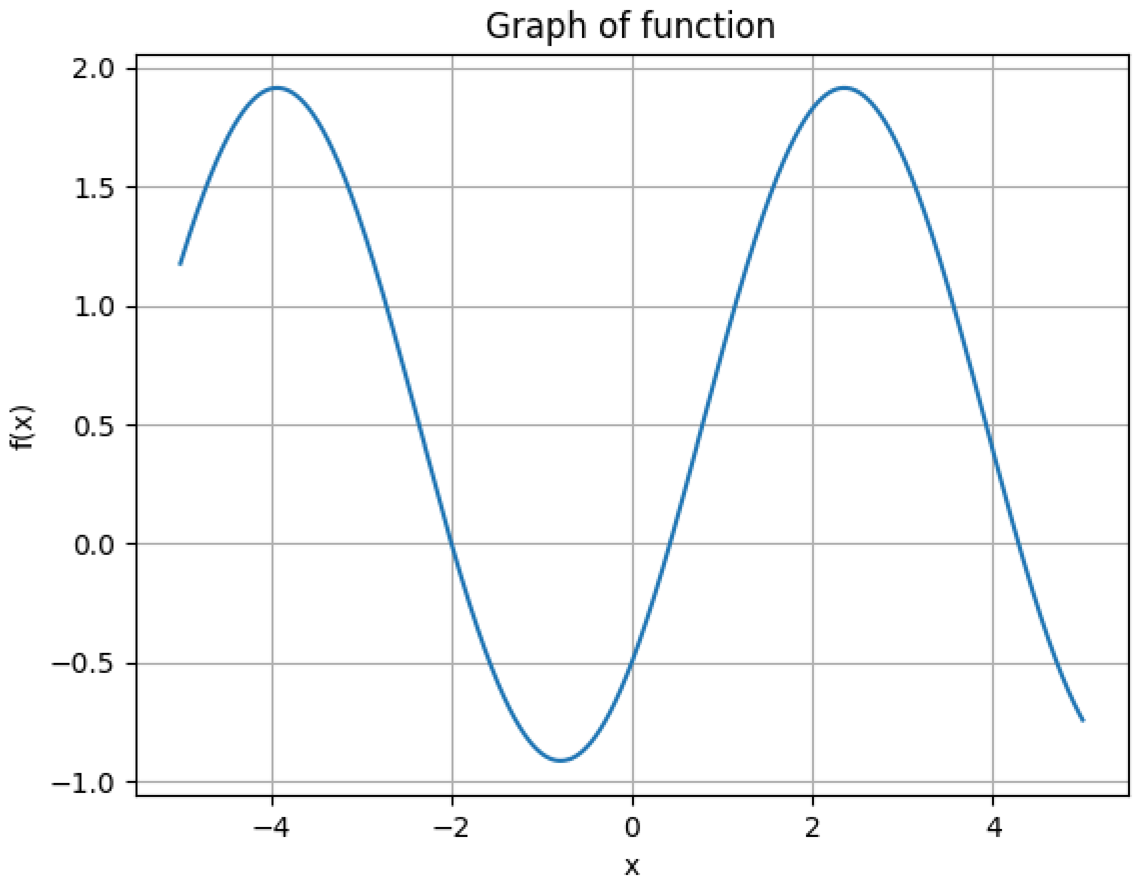
1. The root of the function is 1.1285400390625 for the initial points 0 and 2.

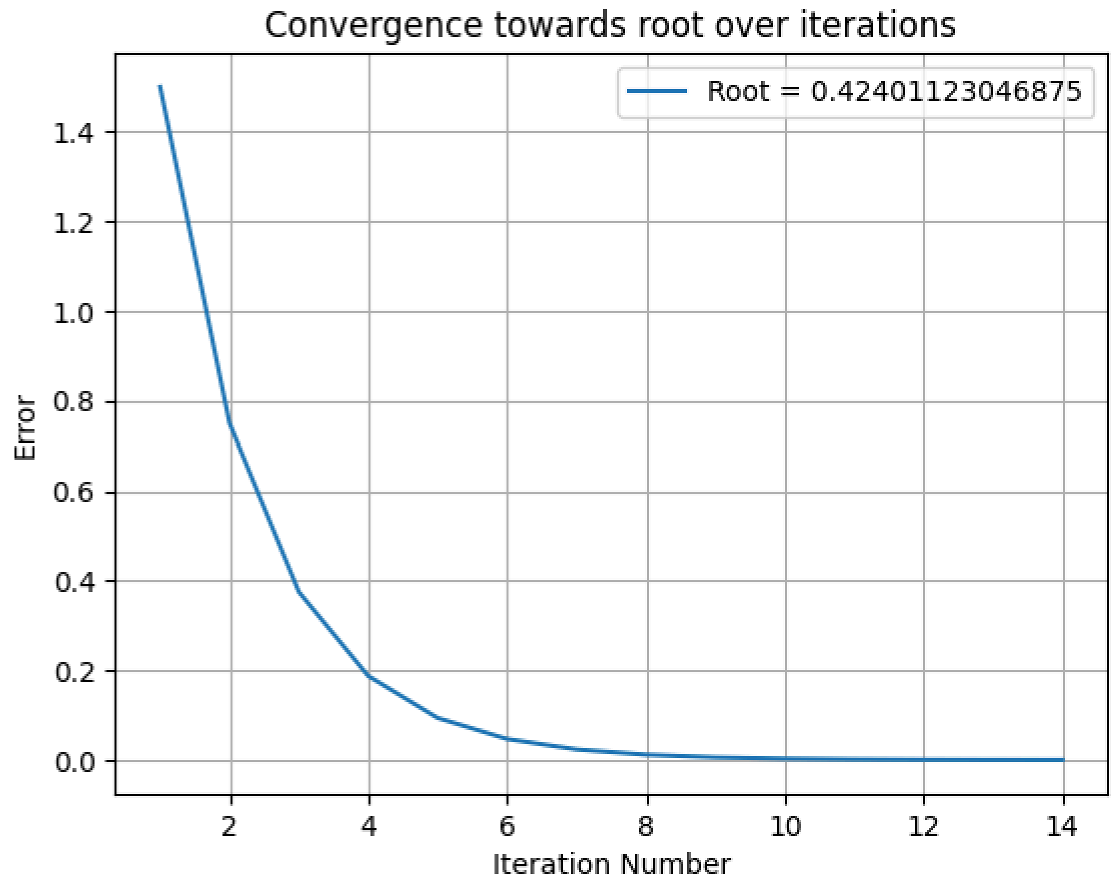
2. We can see the decrease in error with every iteration which shows convergence towards the root.

$$cos(x) = sin(x) + 1/2$$

In [ ]:
```
def fun(x):
    return 0.5 + np.sin(x) - np.cos(x)
df = bisection_get_roots(-1,2,err)
df
```

```
root is : 0.42401123046875
```





Out[ ]:

	Iteration	an	bn	c	b-c	f(c)
0	1	-1.000000	2.000000	0.500000	1.500000	0.101843
1	2	-1.000000	0.500000	-0.250000	0.750000	-0.716316
2	3	-0.250000	0.500000	0.125000	0.375000	-0.367523
3	4	0.125000	0.500000	0.312500	0.187500	-0.144129
4	5	0.312500	0.500000	0.406250	0.093750	-0.023442
5	6	0.406250	0.500000	0.453125	0.046875	0.038694
6	7	0.406250	0.453125	0.429688	0.023438	0.007491
7	8	0.406250	0.429688	0.417969	0.011719	-0.008010
8	9	0.417969	0.429688	0.423828	0.005859	-0.000268
9	10	0.423828	0.429688	0.426758	0.002930	0.003609
10	11	0.423828	0.426758	0.425293	0.001465	0.001670
11	12	0.423828	0.425293	0.424561	0.000732	0.000701
12	13	0.423828	0.424561	0.424194	0.000366	0.000216
13	14	0.423828	0.424194	0.424011	0.000183	-0.000026

Result:

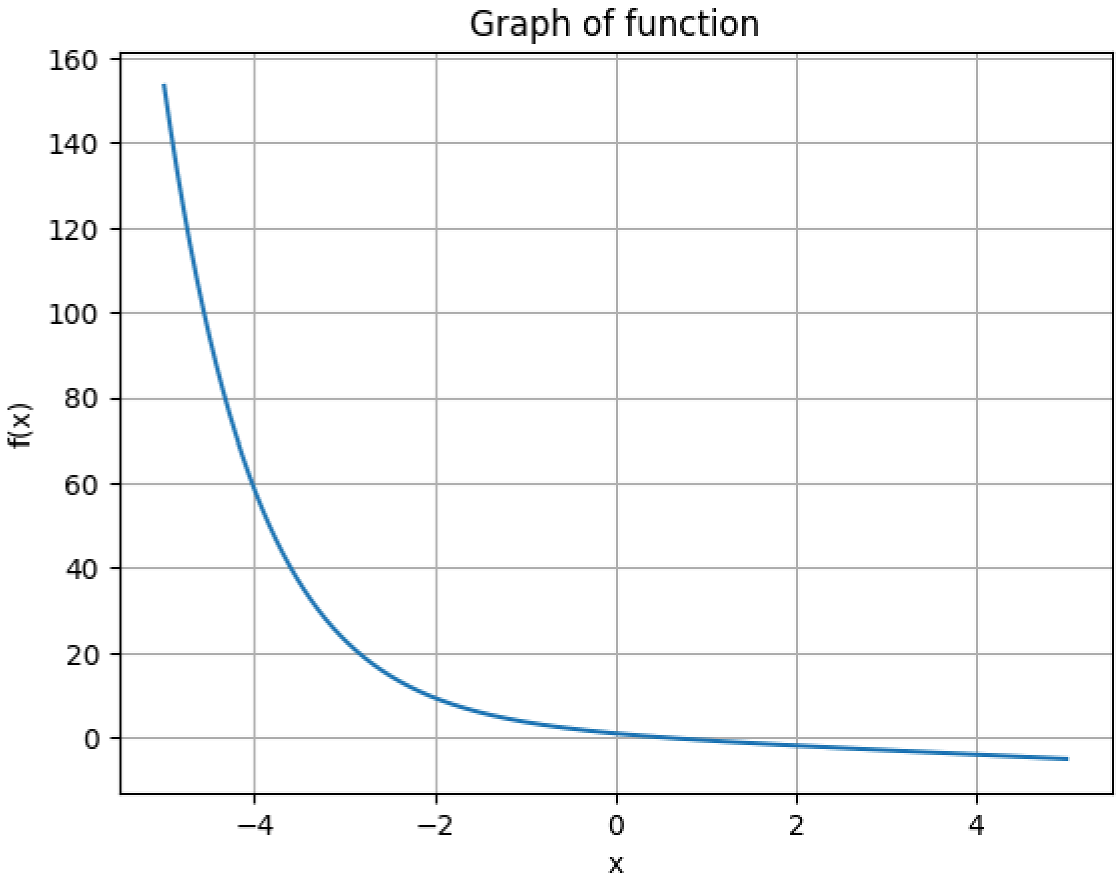
1. The root of the function is 0.42401123046875 for the initial points -1 and 2.

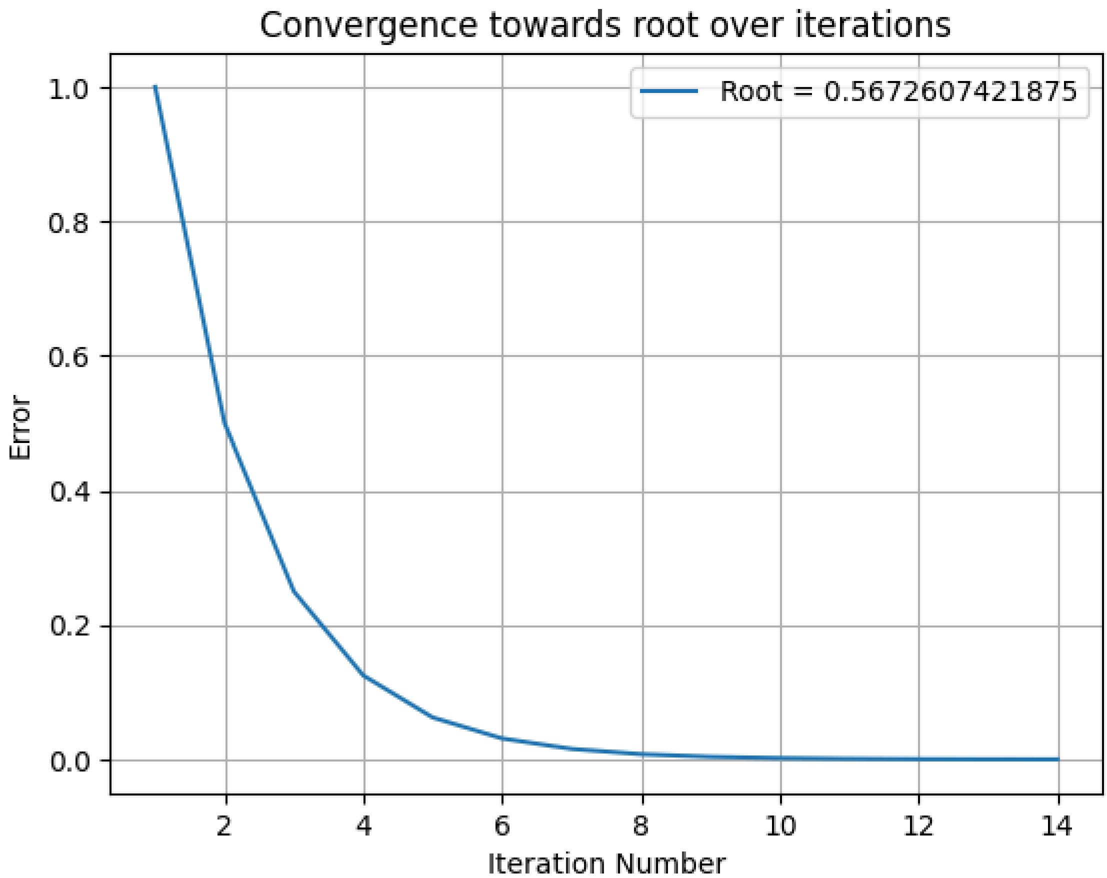
2. We can see the decrease in error with every iteration which shows convergence towards the root.

$$e^{-x} - x$$

In [ ]:
```
def fun(x):
    return np.exp(-x) - x
df = bisection_get_roots(0,2,err)
df
```

```
root is : 0.5672607421875
```





Out[ ]:

	Iteration	an	bn	c	b-c	f(c)
0	1	0.000000	2.000000	1.000000	1.000000	-0.632121
1	2	0.000000	1.000000	0.500000	0.500000	0.106531
2	3	0.500000	1.000000	0.750000	0.250000	-0.277633
3	4	0.500000	0.750000	0.625000	0.125000	-0.089739
4	5	0.500000	0.625000	0.562500	0.062500	0.007283
5	6	0.562500	0.625000	0.593750	0.031250	-0.041498
6	7	0.562500	0.593750	0.578125	0.015625	-0.017176
7	8	0.562500	0.578125	0.570312	0.007812	-0.004964
8	9	0.562500	0.570312	0.566406	0.003906	0.001155
9	10	0.566406	0.570312	0.568359	0.001953	-0.001905
10	11	0.566406	0.568359	0.567383	0.000977	-0.000375
11	12	0.566406	0.567383	0.566895	0.000488	0.000390
12	13	0.566895	0.567383	0.567139	0.000244	0.000007
13	14	0.567139	0.567383	0.567261	0.000122	-0.000184

Result:

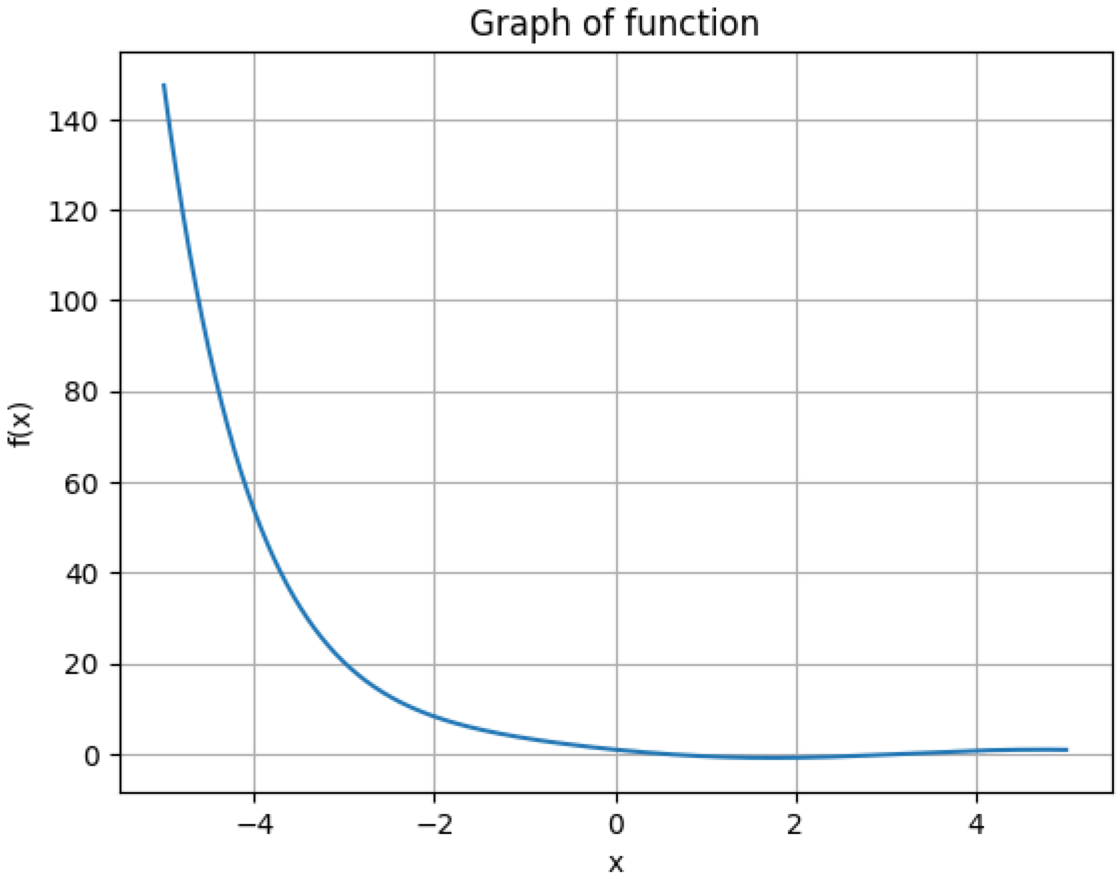
1. The root of the function is 0.5672607421875 for the initial points 0 and 2.

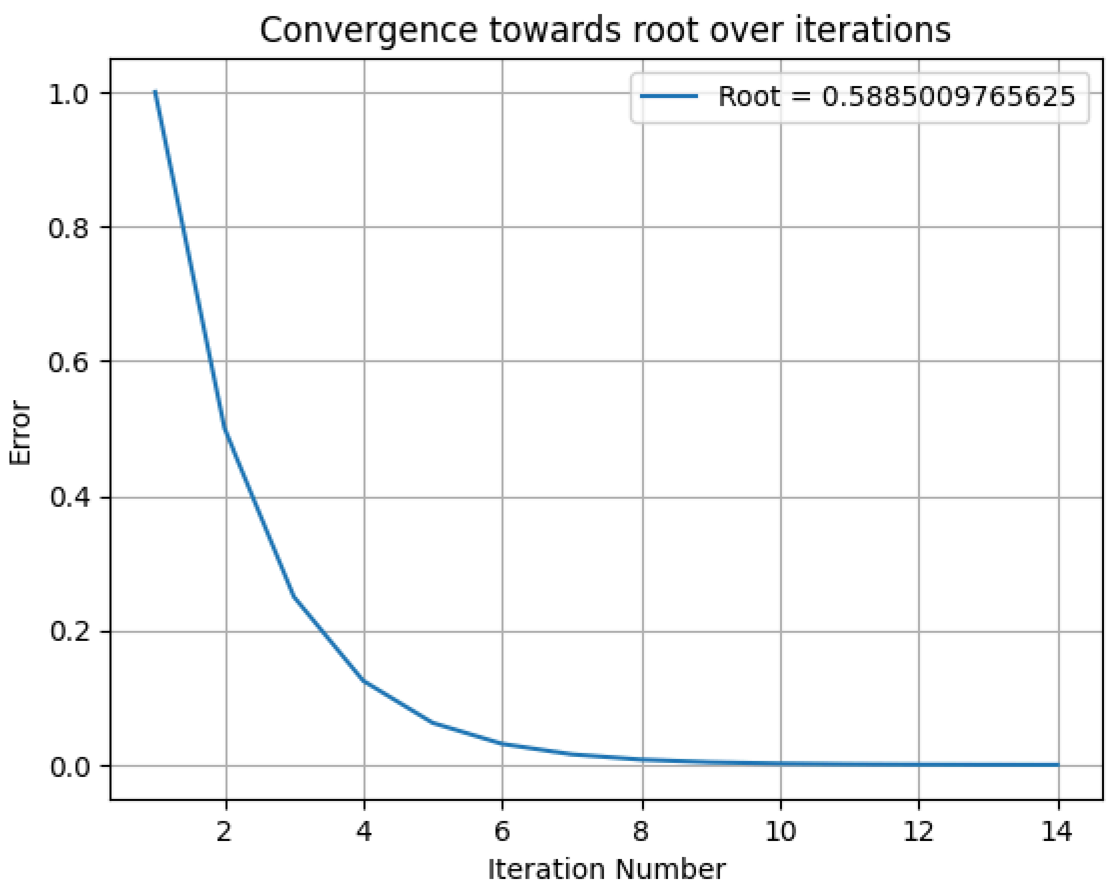
2. We can see the decrease in error with every iteration which shows convergence towards the root.

$$e^{-x} = sin(x)$$

In [ ]:
```
def fun(x):
    return np.exp(-x) - np.sin(x)
df = bisection_get_roots(0,2,err)
df
```

```
root is : 0.5885009765625
```





Out[ ]:

	Iteration	an	bn	c	b-c	f(c)
0	1	0.000000	2.000000	1.000000	1.000000	-0.473592
1	2	0.000000	1.000000	0.500000	0.500000	0.127105
2	3	0.500000	1.000000	0.750000	0.250000	-0.209272
3	4	0.500000	0.750000	0.625000	0.125000	-0.049836
4	5	0.500000	0.625000	0.562500	0.062500	0.036480
5	6	0.562500	0.625000	0.593750	0.031250	-0.007221
6	7	0.562500	0.593750	0.578125	0.015625	0.014495
7	8	0.578125	0.593750	0.585938	0.007812	0.003603
8	9	0.585938	0.593750	0.589844	0.003906	-0.001817
9	10	0.585938	0.589844	0.587891	0.001953	0.000891
10	11	0.587891	0.589844	0.588867	0.000977	-0.000464
11	12	0.587891	0.588867	0.588379	0.000488	0.000213
12	13	0.588379	0.588867	0.588623	0.000244	-0.000125
13	14	0.588379	0.588623	0.588501	0.000122	0.000044

Result:

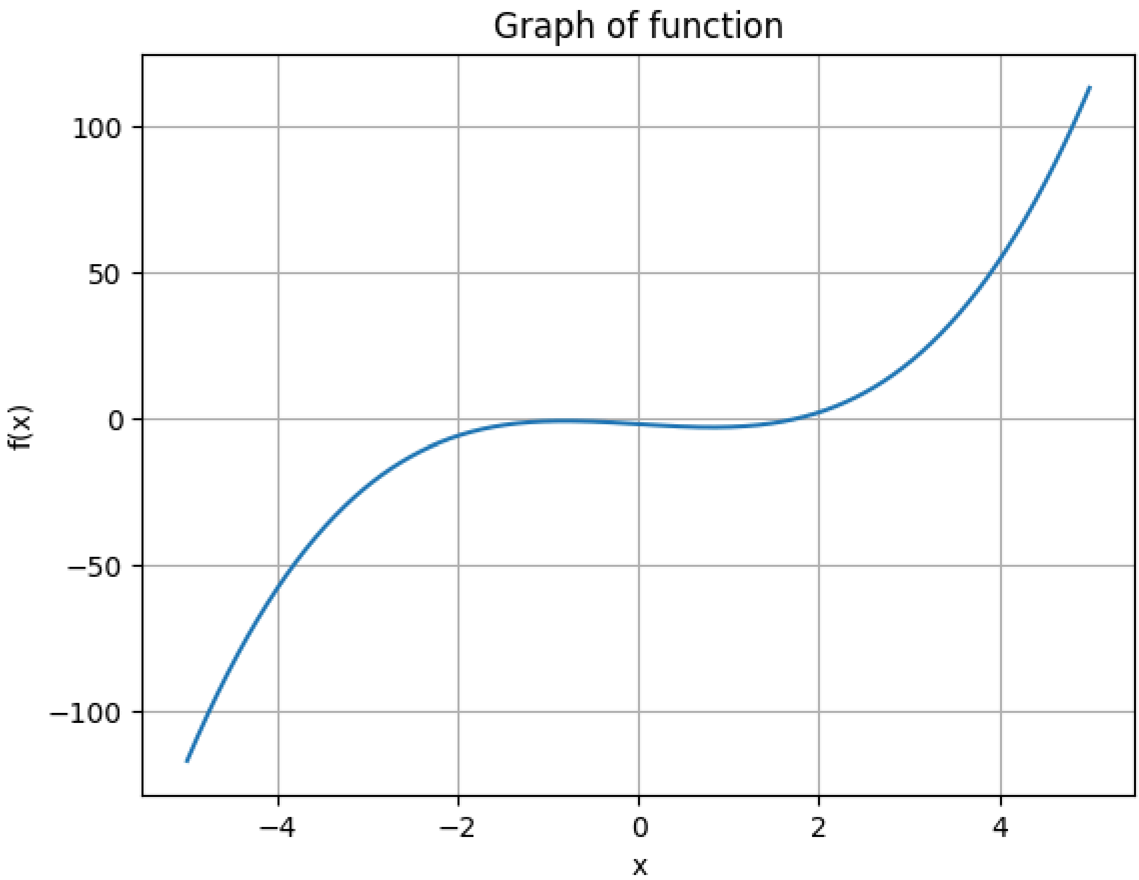
1. The root of the function is 0.5885009765625 for the initial points 0 and 2.

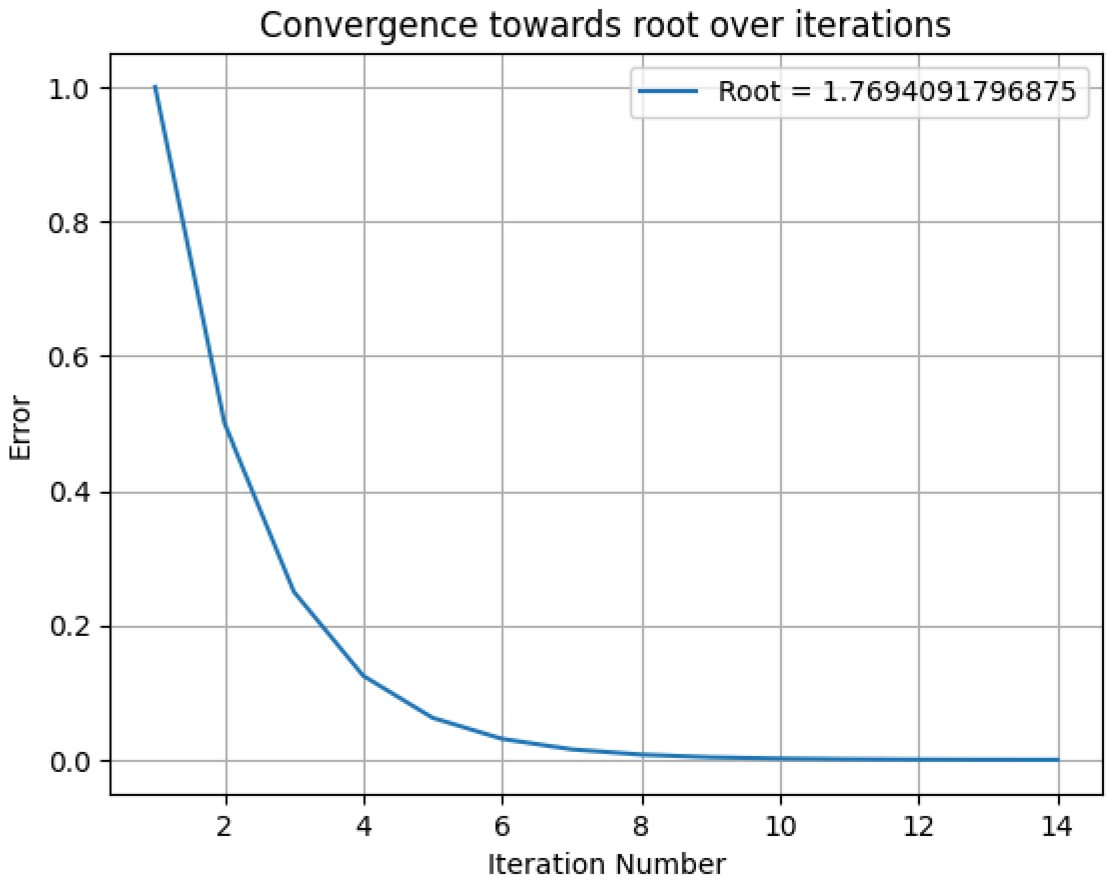
2. We can see the decrease in error with every iteration which shows convergence towards the root.

$$x^3 - 2 * x - 2$$

In [ ]:
```
def fun(x):
    return x**3 - 2*x - 2
df = bisection_get_roots(0,2,err)
df
```

```
root is : 1.7694091796875
```





Out[ ]:

	Iteration	an	bn	c	b-c	f(c)
0	1	0.000000	2.000000	1.000000	1.000000	-3.000000
1	2	1.000000	2.000000	1.500000	0.500000	-1.625000
2	3	1.500000	2.000000	1.750000	0.250000	-0.140625
3	4	1.750000	2.000000	1.875000	0.125000	0.841797
4	5	1.750000	1.875000	1.812500	0.062500	0.329346
5	6	1.750000	1.812500	1.781250	0.031250	0.089142
6	7	1.750000	1.781250	1.765625	0.015625	-0.027035
7	8	1.765625	1.781250	1.773438	0.007812	0.030729
8	9	1.765625	1.773438	1.769531	0.003906	0.001766
9	10	1.765625	1.769531	1.767578	0.001953	-0.012655
10	11	1.767578	1.769531	1.768555	0.000977	-0.005449
11	12	1.768555	1.769531	1.769043	0.000488	-0.001843
12	13	1.769043	1.769531	1.769287	0.000244	-0.000039
13	14	1.769287	1.769531	1.769409	0.000122	0.000864

Result:

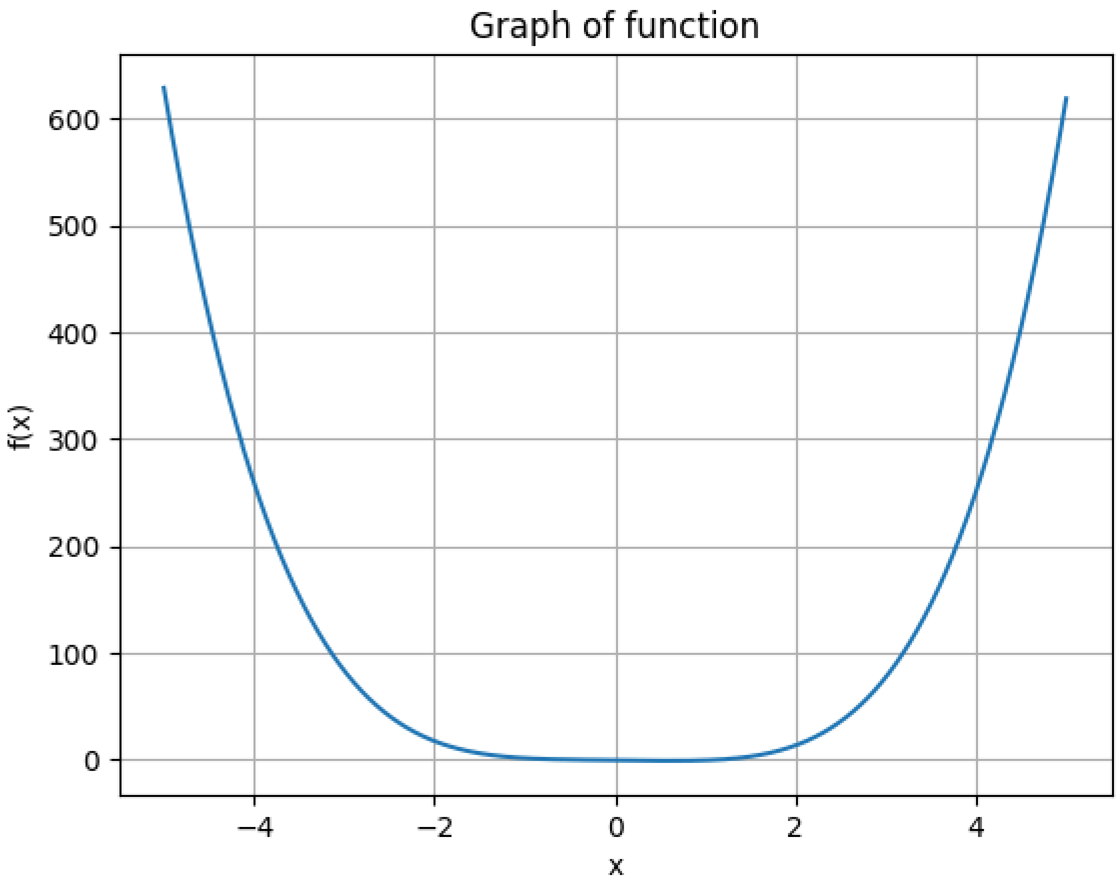
1. The root of the function is 1.7694091796875 for the initial points 0 and 2.

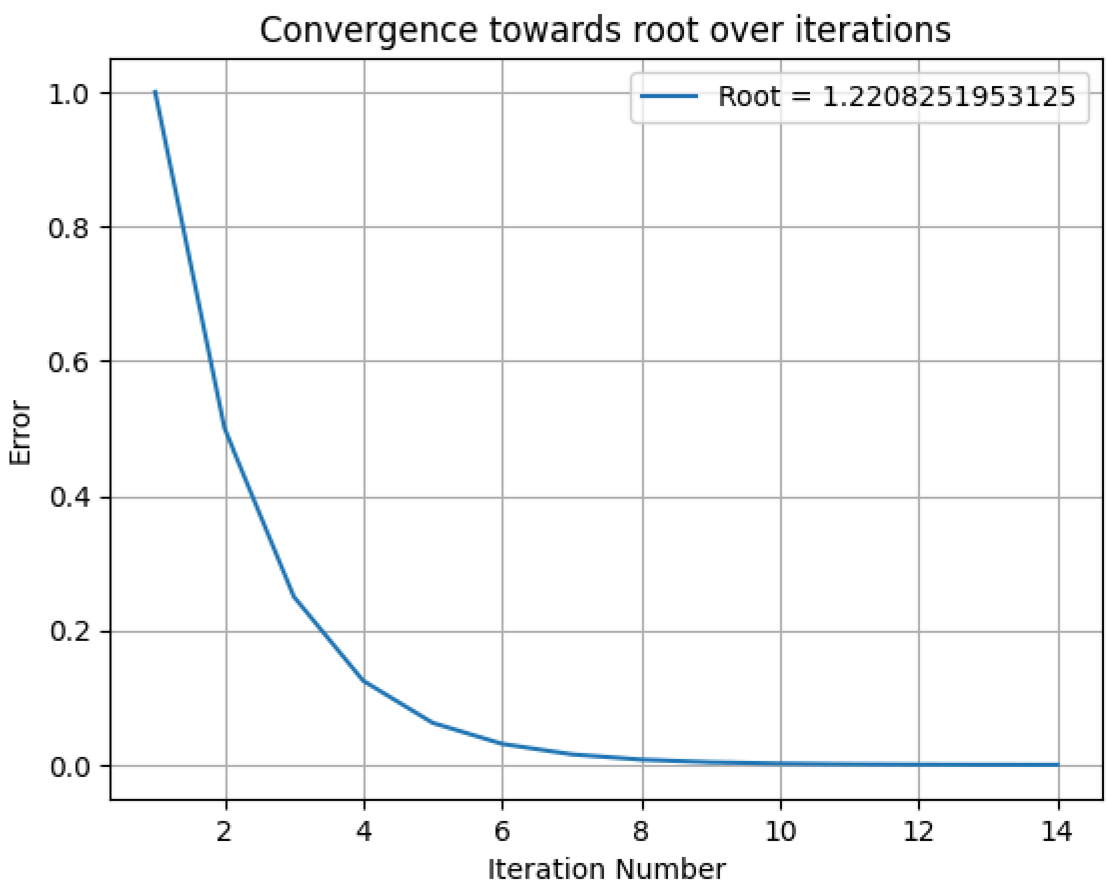
2. We can see the decrease in error with every iteration which shows convergence towards the root.

$$x^4 - x - 1$$

In [ ]:
```
def fun(x):
    return x**4 - x - 1
df = bisection_get_roots(0,2,err)
df
```

```
root is : 1.2208251953125
```





Out[ ]:

	Iteration	an	bn	c	b-c	f(c)
0	1	0.000000	2.000000	1.000000	1.000000	-1.000000
1	2	1.000000	2.000000	1.500000	0.500000	2.562500
2	3	1.000000	1.500000	1.250000	0.250000	0.191406
3	4	1.000000	1.250000	1.125000	0.125000	-0.523193
4	5	1.125000	1.250000	1.187500	0.062500	-0.198959
5	6	1.187500	1.250000	1.218750	0.031250	-0.012481
6	7	1.218750	1.250000	1.234375	0.015625	0.087231
7	8	1.218750	1.234375	1.226562	0.007812	0.036824
8	9	1.218750	1.226562	1.222656	0.003906	0.012035
9	10	1.218750	1.222656	1.220703	0.001953	-0.000257
10	11	1.220703	1.222656	1.221680	0.000977	0.005880
11	12	1.220703	1.221680	1.221191	0.000488	0.002809
12	13	1.220703	1.221191	1.220947	0.000244	0.001276
13	14	1.220703	1.220947	1.220825	0.000122	0.000509

Result:

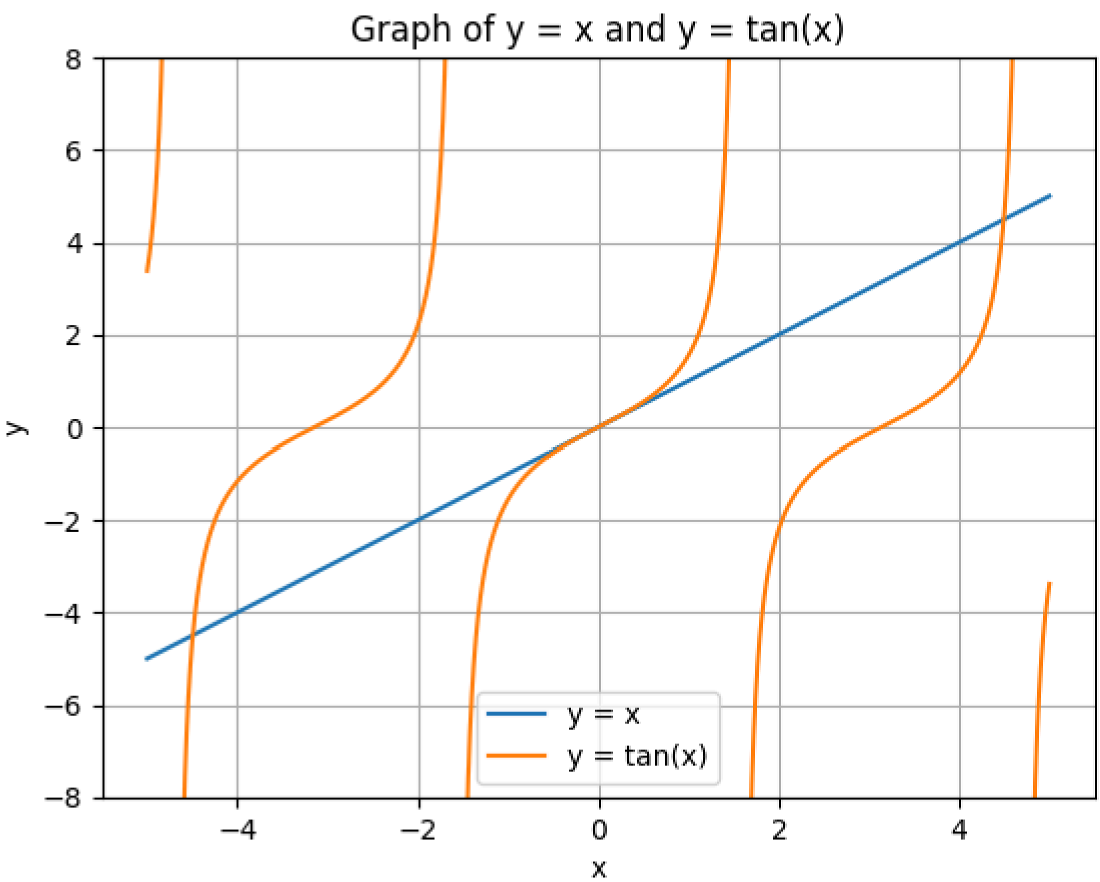
1. The root of the function is 1.2208251953125 for the initial points 0 and 2.

2. We can see the decrease in error with every iteration which shows convergence towards the root.

$$x = tan(x)$$

In [ ]:
```
x = np.arange(-5,5+err,err)
y = np.tan(x)
y[:-1][np.diff(y) < 0] = np.nan
plt.plot(x,x,label = 'y = x')
plt.plot(x,y,label = 'y = tan(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.ylim(-8,8)
plt.title('Graph of y = x and y = tan(x)')
plt.legend()
plt.grid(True)
plt.plot()
```

Out[ ]: []



In [ ]:
```
def bisection_get_roots(a,b,err):
    n = maxiter(a,b,err)
    roots = []
    x_range = np.arange(-5,5+err,err)
    for i in range(n):
        x = (a+b)/2
        roots.append(x)
        fa = fun(a)
        fb = fun(b)
        fx = fun(x)
```

```python
        if fa*fx == 0 or fb*fx == 0:
            return roots
        elif fa*fx < 0 :
            b = x
        else:
            a = x
    return roots
```

In [ ]:
```python
def fun(x):
    return np.tan(x) - x
print('root greater than pi/2 =',bisection_get_roots(2,4.5,err)[-1])
```

```
root greater than pi/2 = 4.493438720703125
```

In [ ]:
```python
r1 = bisection_get_roots(100,102.2,err)[-1]
r2 = bisection_get_roots(98,98.96,err)[-1]
r = 0
if(r1 - 100 < 100 - r2):
    r = r1
else:
    r = r2
print('root nearest to 100 is',r)
```

```
root nearest to 100 is 98.9500390625
```

Machine Epsilon

1. Information about machine epsilon: Machine epsilon is defined to be the smallest floating-point value, e, that satisfies the equation, 1+e!=1.
2. Algorithm for finding machine epsilon: Start with e=1. Divide e by 2 and check if 1+e==1. If not, keep on repeating the process till you get 1+e==1. This value e is or machine epsilon.

Q - 1

In [ ]:
```python
def MachineEpsilon(EPS):
        while ((1+EPS) != 1):
                prev_EPS = EPS
                EPS = EPS / 2
        print("Machine Epsilon is : " ,prev_EPS)
MachineEpsilon(1)
```

```
Machine Epsilon is :  2.220446049250313e-16
```

Result: The value of machine epsilon from the above method comes out to be 2.220446049250313e-16.

Q - 2

In [ ]:
```python
def MachineEpsilonN(n,EPS):
        while ((n+EPS) != n):
                prev_EPS = EPS
```

```
                EPS = EPS / 2
        print("Machine Epsilon for n =",n,"is :" ,prev_EPS)

MachineEpsilonN(5,0.5)
```

```
Machine Epsilon for n = 5 is : 8.881784197001252e-16
```

Results: In this question we have generalised the value of n. In earlier question we had assumed n=1 and solved for epsilon. In this question we have taken n=5. The value of machine epsilon in this case is 8.881784197001252e-16. This shows that machine epsilon changes with different values of n. Also another observation is that value of epsilon increases with increase in n (can be noticed by changing values of n).