# Computational and Numerical Methods Lab - 3

Abhimanyu Karia: 202201435
Devarshi Patel : 202201447

# Newton-Raphson method

-> The Newton-Raphson method is a way to quickly find a good approximation for the root of a real-valued function f(x)=0. It uses the idea that a continuous and differentiable function can be approximated by a straight line tangent to it.

->Algorithm for Newton-Raphson method:

1. General formula: $x_{n+1}$=$x_n$-$f(x_n)/f^{'}(x_n)$
2. We start with an initial point $x_0$.
3. Using the formula, keep on calculating subsequent $x_n$ till we get the desired accuracy.

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

In [ ]:
```
err = 1e-7
```

# Q1)

$$f(x) = x^6 - x - 1$$

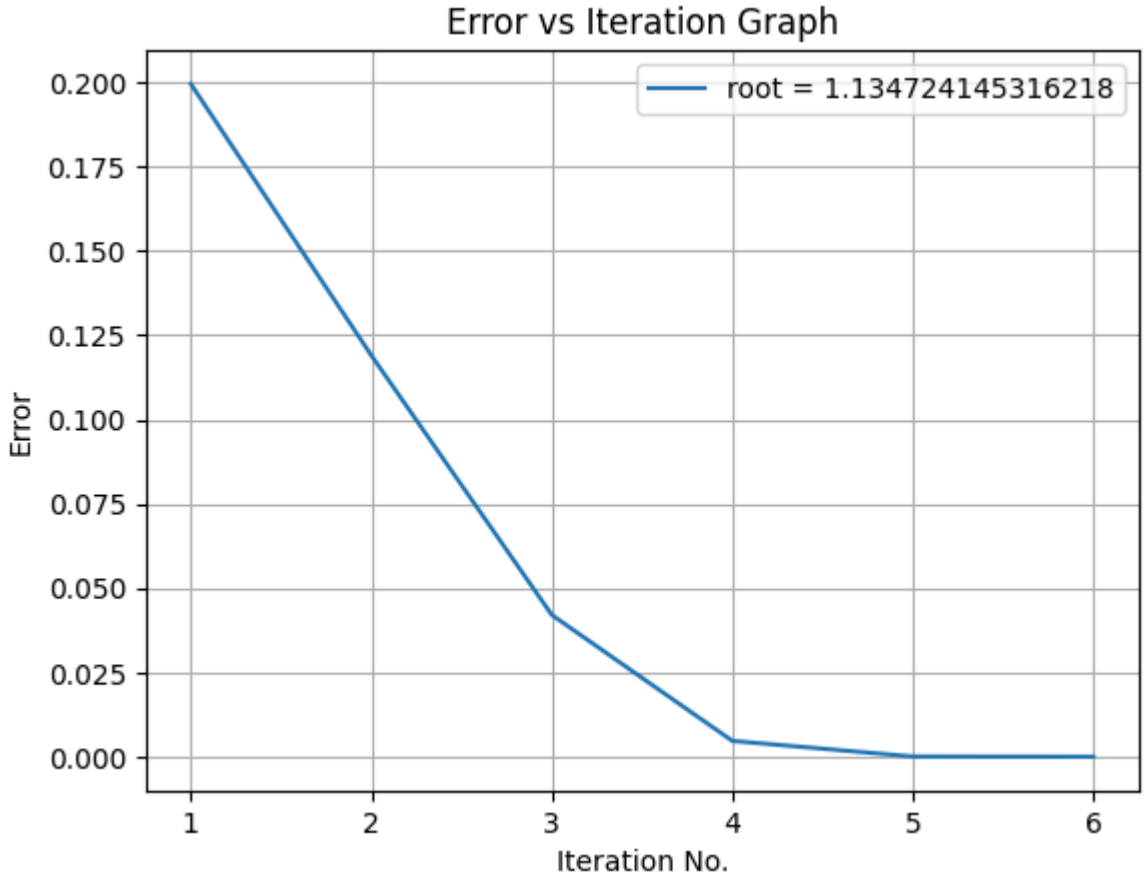
In [ ]:
```
def fun(x):
    return x**6 - x - 1

def dfun(x):
    return 6*(x**5) - 1

def newton_raphson(x0,err):
    data = []
    roots = []
    error = []
    present = x0
    fpresent = fun(x0)
    dfpresent = dfun(x0)
    next = present-(fpresent/dfpresent)
    roots.append(present)
    i = 1
    while(abs(next - present) > err):
```

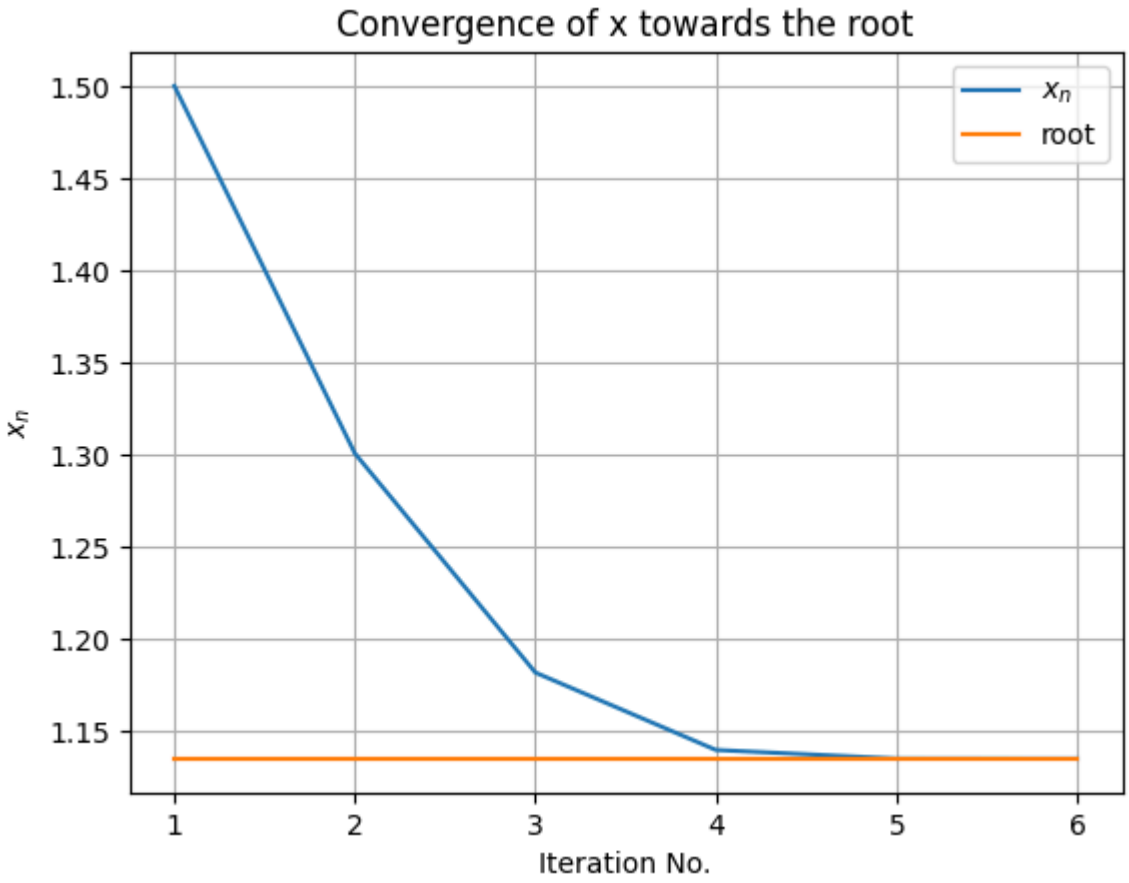
```
        prev = present
        present = next
        fpresent = fun(present)
        dfpresent = dfun(present)
        next = present-(fpresent/dfpresent)
        data.append([i,present,fpresent,next,present - prev])
        roots.append(present)
        error.append(abs(present - prev))
        i += 1
    alpha = next
    data.append([i,next,fun(next),next - (fun(next)/dfun(next)),next - present])
    error.append(next- present)
    for i in range(0,len(data)):
        data[i].append(alpha - roots[i])

    df = pd.DataFrame(data,columns =['iter', 'x\u2099', 'f(x\u2099)', 'x\u2099\u
    iter = np.arange(1,len(error)+1,1)
    plt.figure(1)
    plt.plot(iter,error,label = 'root = ' + str(roots[-1]))
    plt.legend()
    plt.xlabel('Iteration No.')
    plt.ylabel('Error')
    plt.title('Error vs Iteration Graph')
    plt.grid(True)
    plt.plot()
    plt.figure(2)
    r = [roots[-1]]*len(iter)
    plt.plot(iter,roots,label = '$x_n$')
    plt.plot(iter,r,label = 'root')
    plt.legend()
    plt.xlabel('Iteration No.')
    plt.ylabel('$x_n$')
    plt.title('Convergence of x towards the root')
    plt.grid(True)
    plt.plot()

    return df
```

In [ ]:
```
df = newton_raphson(1.5,err)
df
```

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 1.300491 | 2.537264e+00 | 1.181480 | -1.995091e-01 | -3.652759e-01 |
| **1** | 2 | 1.181480 | 5.384586e-01 | 1.139456 | -1.190105e-01 | -1.657667e-01 |
| **2** | 3 | 1.139456 | 4.923525e-02 | 1.134778 | -4.202483e-02 | -4.675628e-02 |
| **3** | 4 | 1.134778 | 5.503239e-04 | 1.134724 | -4.677965e-03 | -4.731452e-03 |
| **4** | 5 | 1.134724 | 7.113585e-08 | 1.134724 | -5.347992e-05 | -5.348684e-05 |
| **5** | 6 | 1.134724 | 1.554312e-15 | 1.134724 | -6.914698e-09 | -6.914698e-09 |

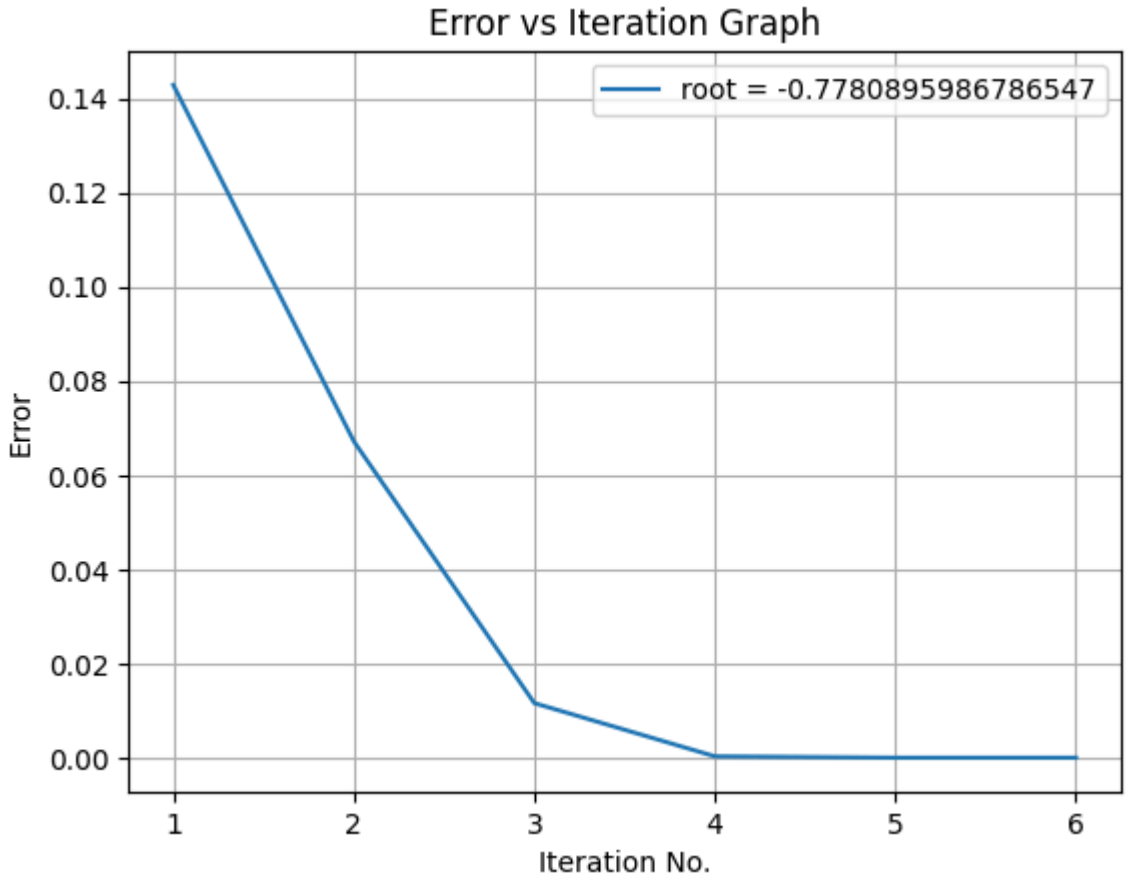


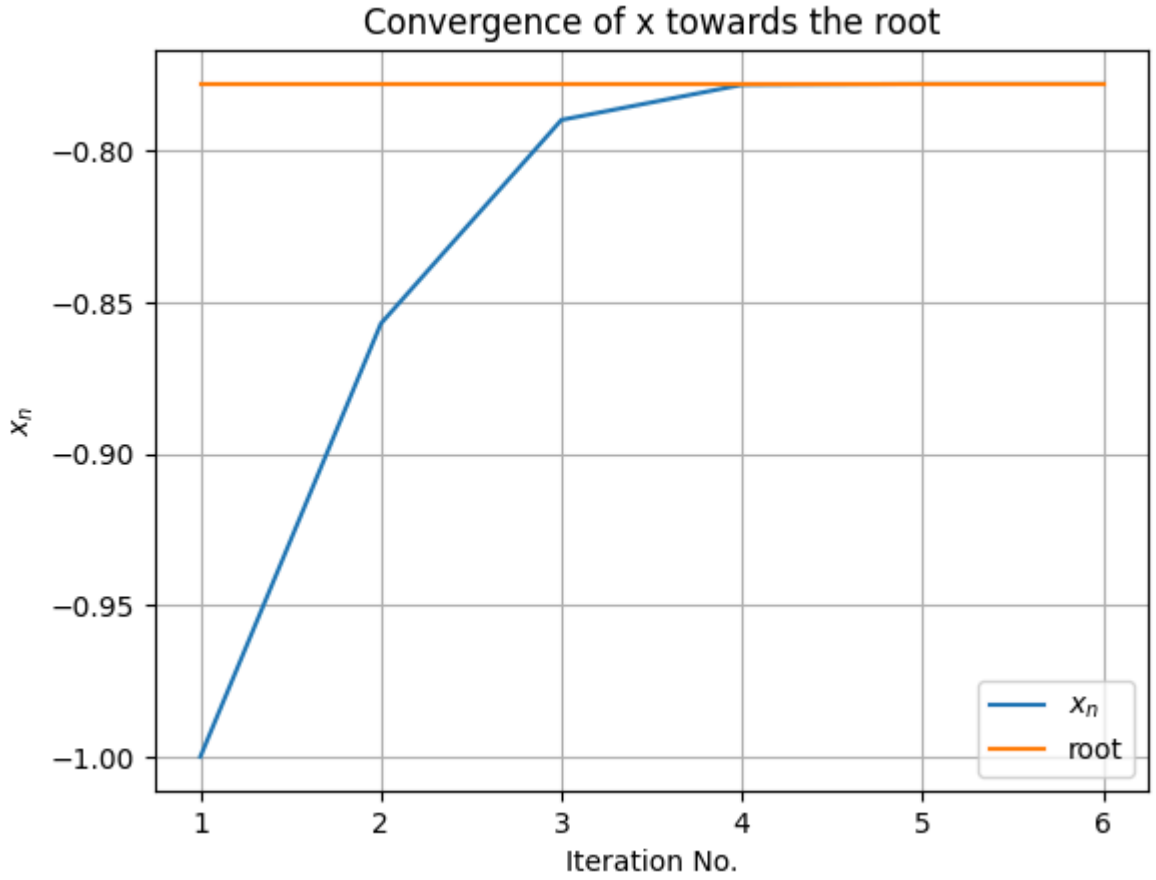




In [ ]:
```
df = newton_raphson(-1,err)
df
```

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | -0.857143 | 2.537123e-01 | -0.789952 | 1.428571e-01 | 2.219104e-01 |
| **1** | 2 | -0.789952 | 3.295042e-02 | -0.778373 | 6.719101e-02 | 7.905326e-02 |
| **2** | 3 | -0.778373 | 7.680138e-04 | -0.778090 | 1.157914e-02 | 1.186225e-02 |
| **3** | 4 | -0.778090 | 4.406060e-07 | -0.778090 | 2.829499e-04 | 2.831125e-04 |
| **4** | 5 | -0.778090 | 1.452172e-13 | -0.778090 | 1.625135e-07 | 1.625136e-07 |
| **5** | 6 | -0.778090 | 2.220446e-16 | -0.778090 | 5.351275e-14 | 5.351275e-14 |






Result:

1. For the initial point 1.5 the root is 1.134724145316218.
2. For the initial point -1 the root is -0.7780895986786547.

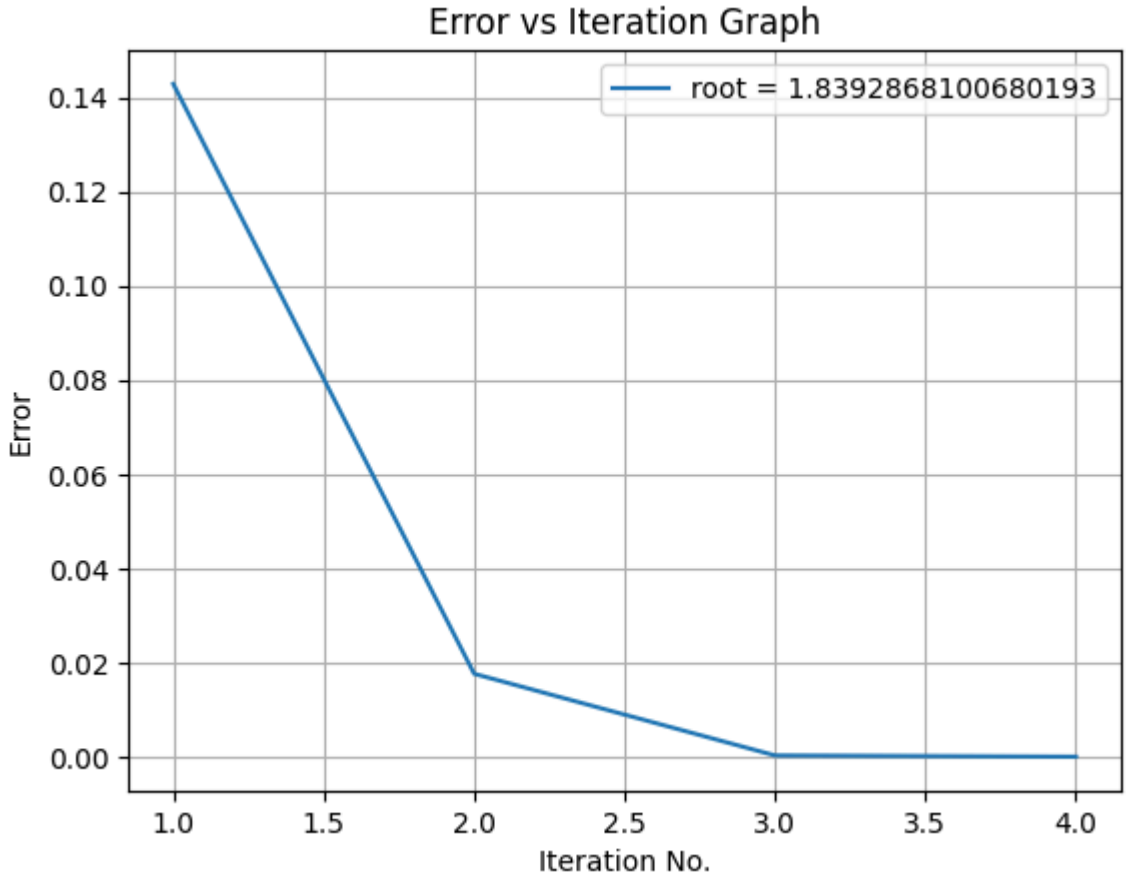
# Q2)

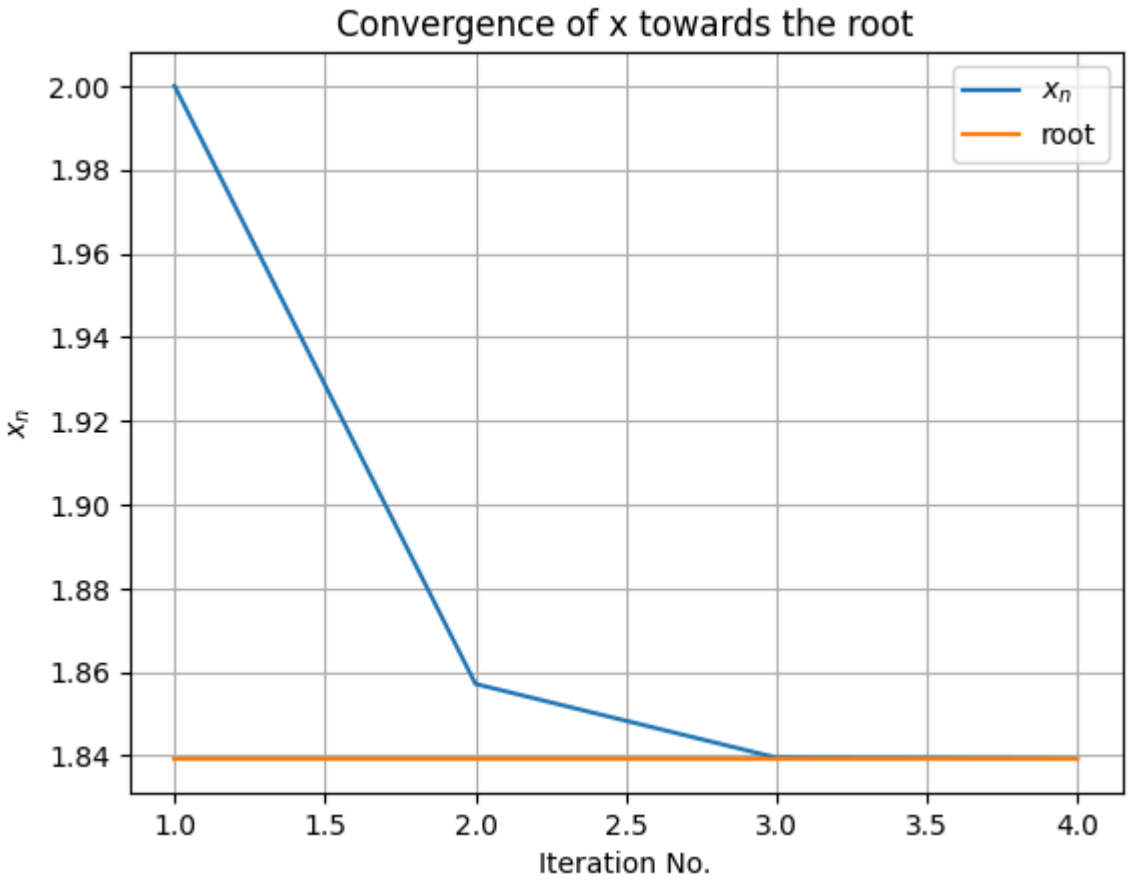
$$f(x) = x^3 - x^2 - x - 1$$

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

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 1.857143 | 9.912536e-02 | 1.839545 | -1.428571e-01 | -1.607132e-01 |
| **1** | 2 | 1.839545 | 1.410329e-03 | 1.839287 | -1.759834e-02 | -1.785610e-02 |
| **2** | 3 | 1.839287 | 3.000700e-07 | 1.839287 | -2.577034e-04 | -2.577582e-04 |
| **3** | 4 | 1.839287 | 1.376677e-14 | 1.839287 | -5.485386e-08 | -5.485386e-08 |




Result:

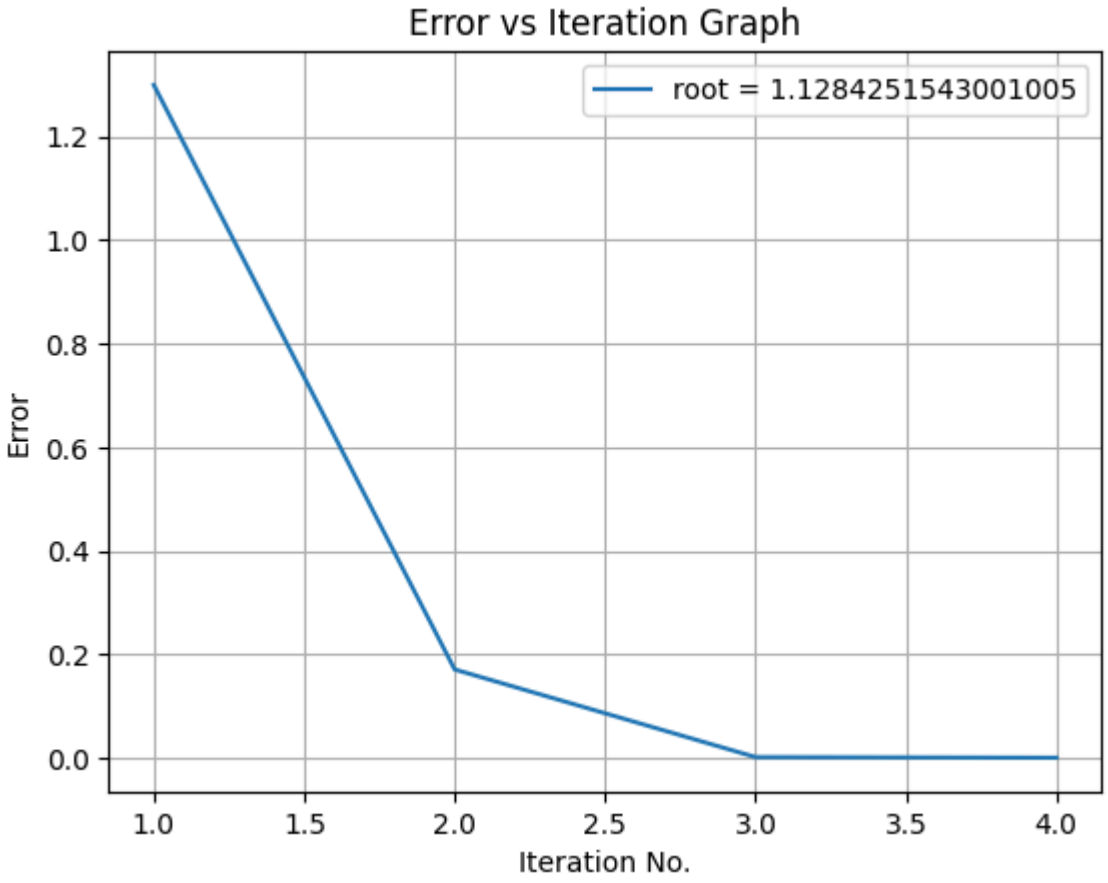
1. For the initial point 2,the root is 1.8392868100680193

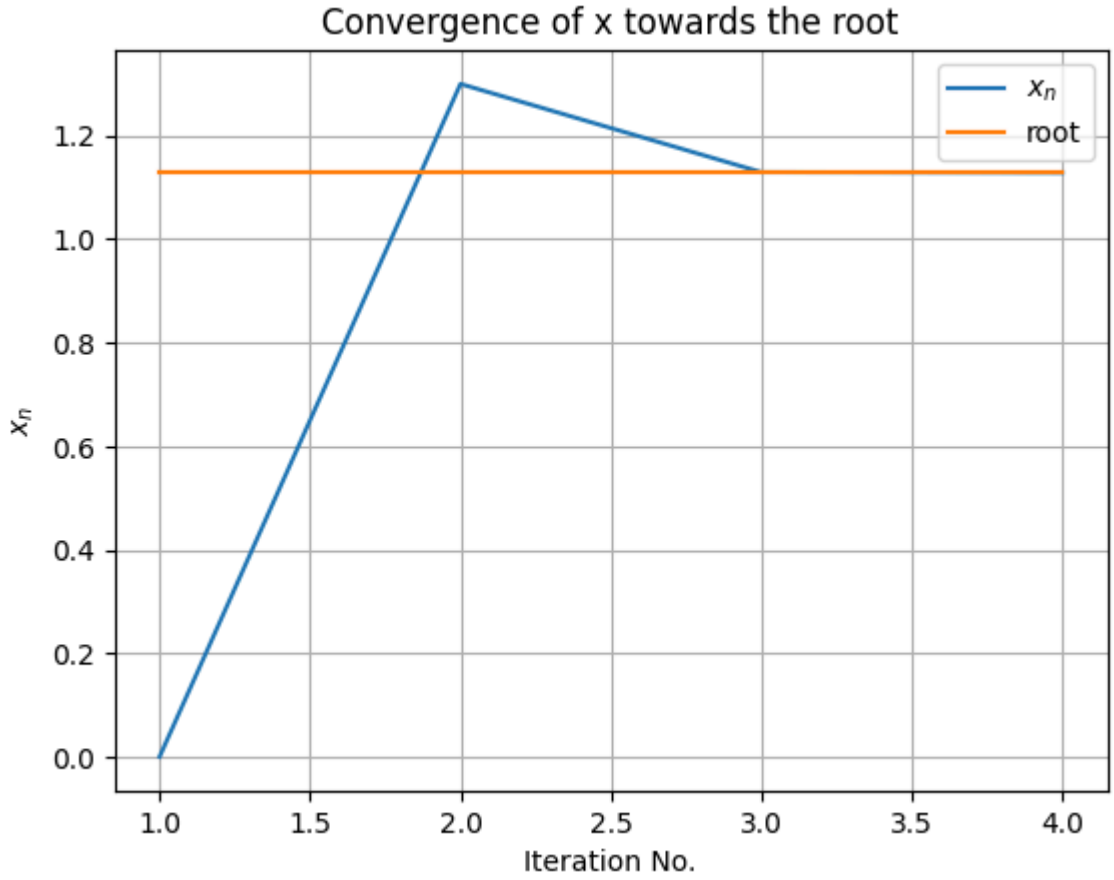
$$f(x) = 1 + 0.3 * cos(x)$$

In [ ]:
```
def fun(x):
    return 1 + 0.3*np.cos(x) - x
def dfun(x):
    return -1 - 0.3*np.sin(x)

df = newton_raphson(0,err)
df
```

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 1.300000 | -2.197504e-01 | 1.129528 | 1.300000e+00 | 1.128425e+00 |
| **1** | 2 | 1.129528 | -1.401569e-03 | 1.128425 | -1.704723e-01 | -1.715749e-01 |
| **2** | 3 | 1.128425 | -7.792977e-08 | 1.128425 | -1.102501e-03 | -1.102562e-03 |
| **3** | 4 | 1.128425 | -2.220446e-16 | 1.128425 | -6.130788e-08 | -6.130788e-08 |






Result:

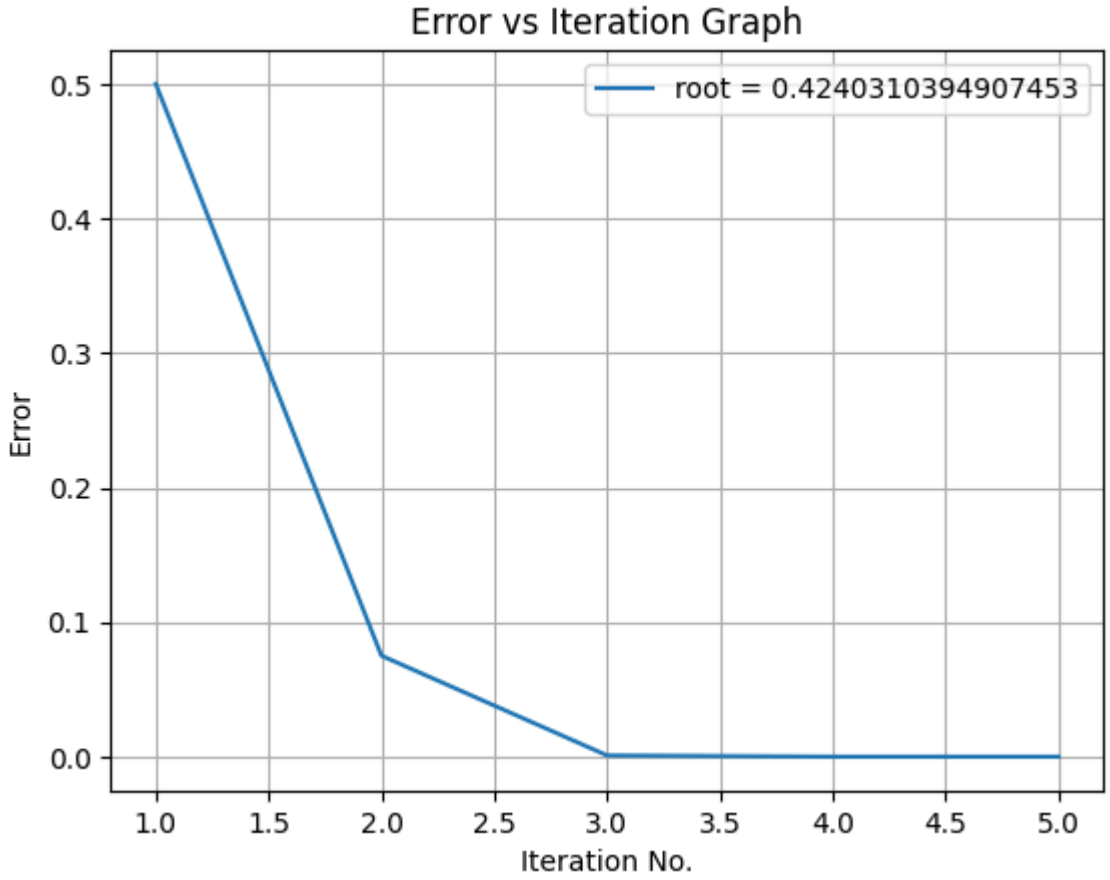
1. For the initial point 0,the root is 1.1284251543001005.

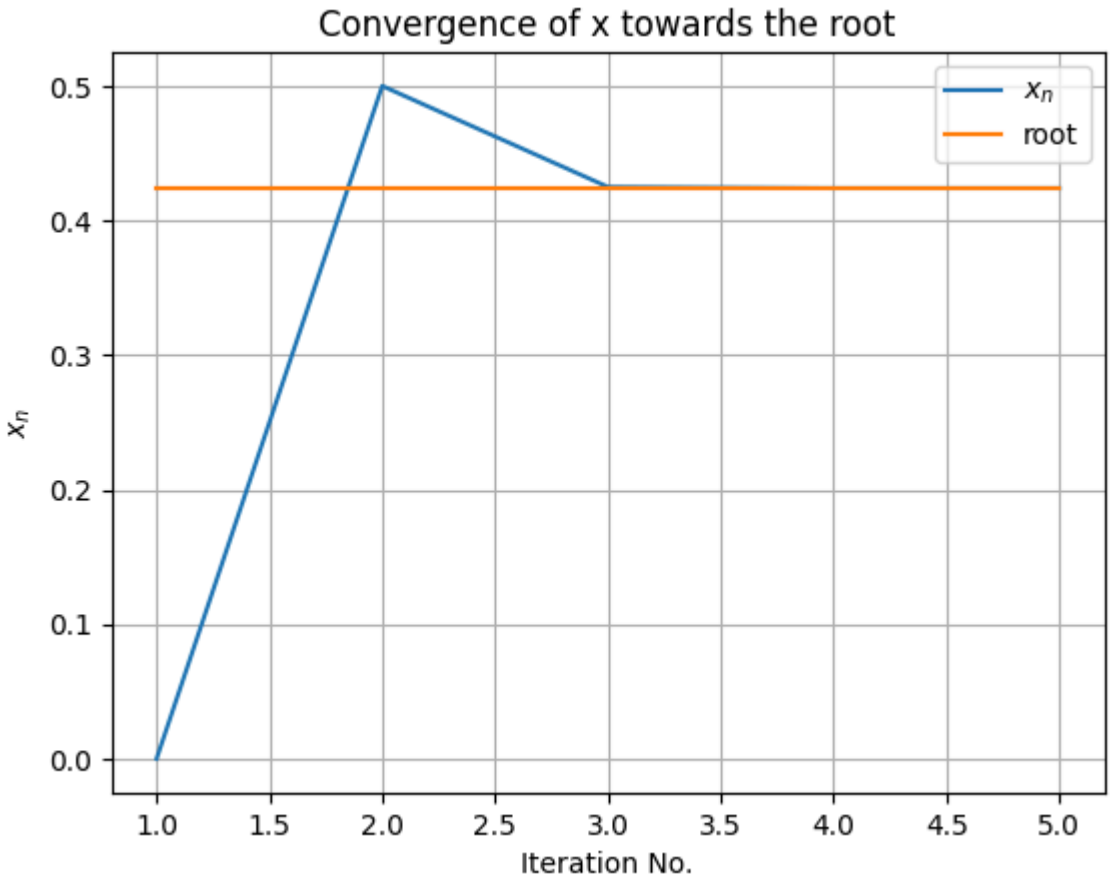
$$f(x) = sin(x) + 1/2 - cos(x)$$

In [ ]:
```
def fun(x):
    return 0.5 + np.sin(x) - np.cos(x)

def dfun(x):
    return np.cos(x) + np.sin(x)
df = newton_raphson(0,err)
df
```

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 0.500000 | 1.018430e-01 | 0.424950 | 5.000000e-01 | 4.240310e-01 |
| **1** | 2 | 0.424950 | 1.216351e-03 | 0.424031 | -7.504965e-02 | -7.596896e-02 |
| **2** | 3 | 0.424031 | 2.108693e-07 | 0.424031 | -9.191557e-04 | -9.193151e-04 |
| **3** | 4 | 0.424031 | 6.328271e-15 | 0.424031 | -1.594022e-07 | -1.594022e-07 |
| **4** | 5 | 0.424031 | 0.000000e+00 | 0.424031 | -4.773959e-15 | -4.773959e-15 |




Result:

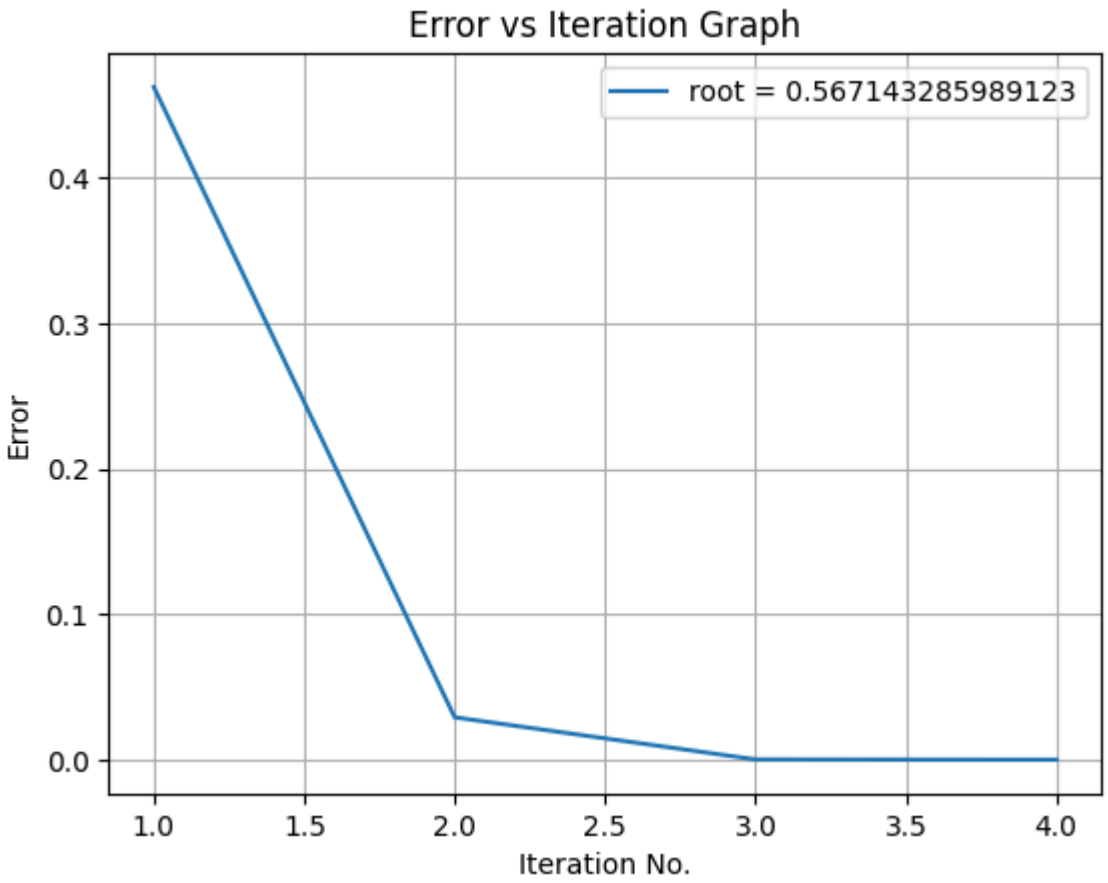
1. For the initial point 0,the root is 0.42403103949074533

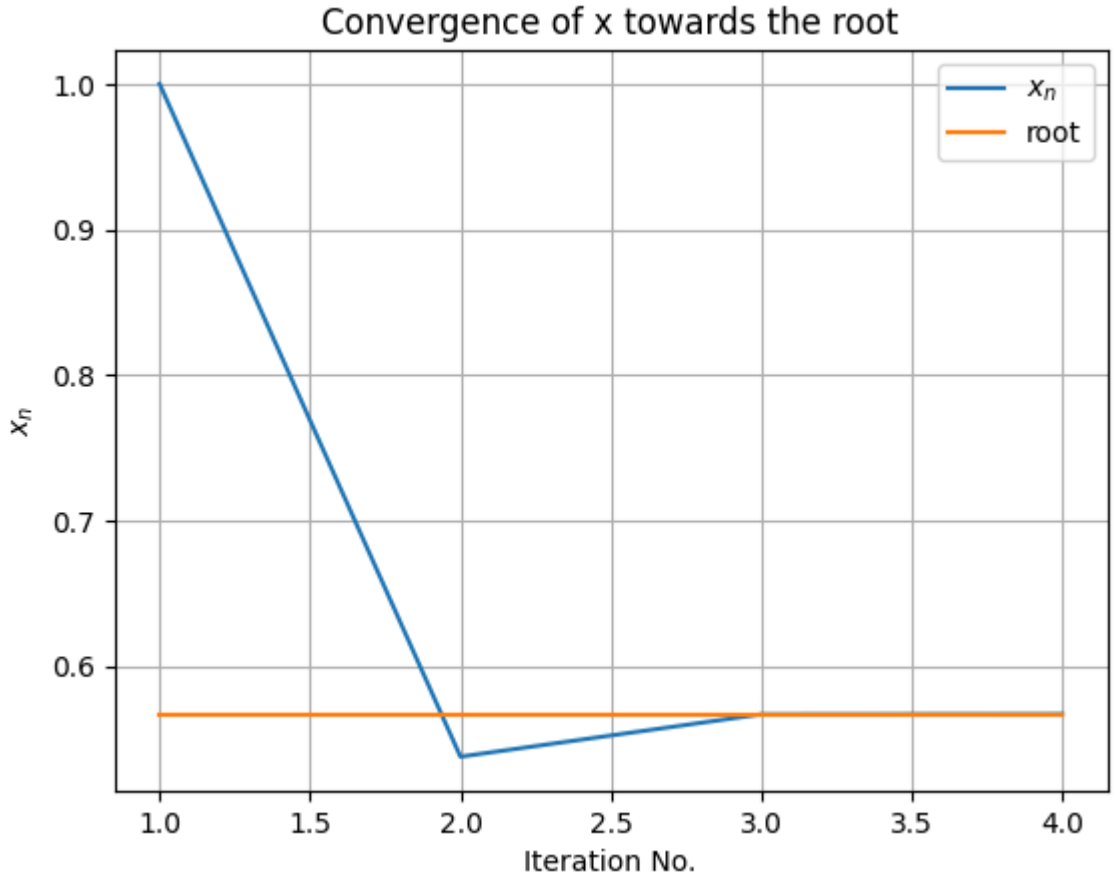
$$f(x) = e^{-x} - x$$

In [ ]:
```
def fun(x):
    return np.exp(-x) - x

def dfun(x):
    return -1 - np.exp(-x)
df = newton_raphson(1,err)
df
```

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 0.537883 | 4.610049e-02 | 0.566987 | -4.621172e-01 | -4.328567e-01 |
| **1** | 2 | 0.566987 | 2.449499e-04 | 0.567143 | 2.910415e-02 | 2.926045e-02 |
| **2** | 3 | 0.567143 | 6.927809e-09 | 0.567143 | 1.562946e-04 | 1.562990e-04 |
| **3** | 4 | 0.567143 | 0.000000e+00 | 0.567143 | 4.420661e-09 | 4.420661e-09 |






Result:

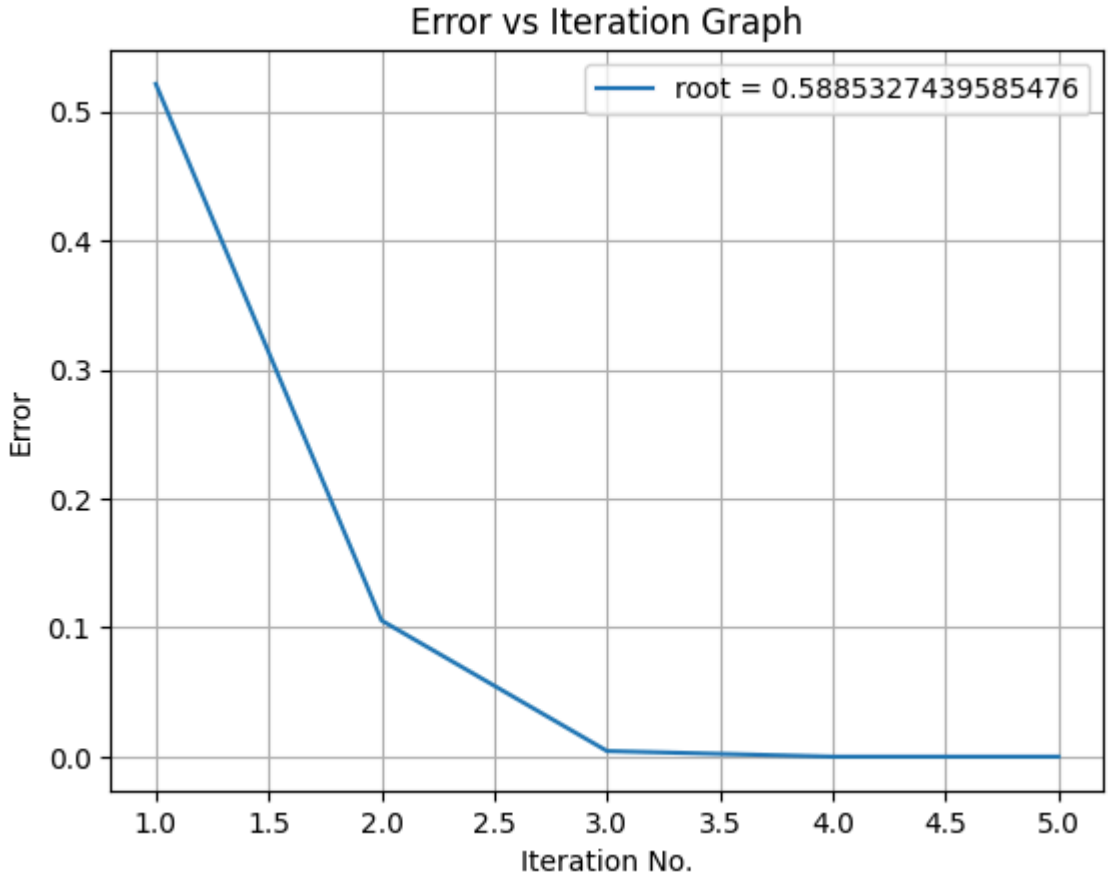
1. For the initial point 1,the root is 0.537143285989123

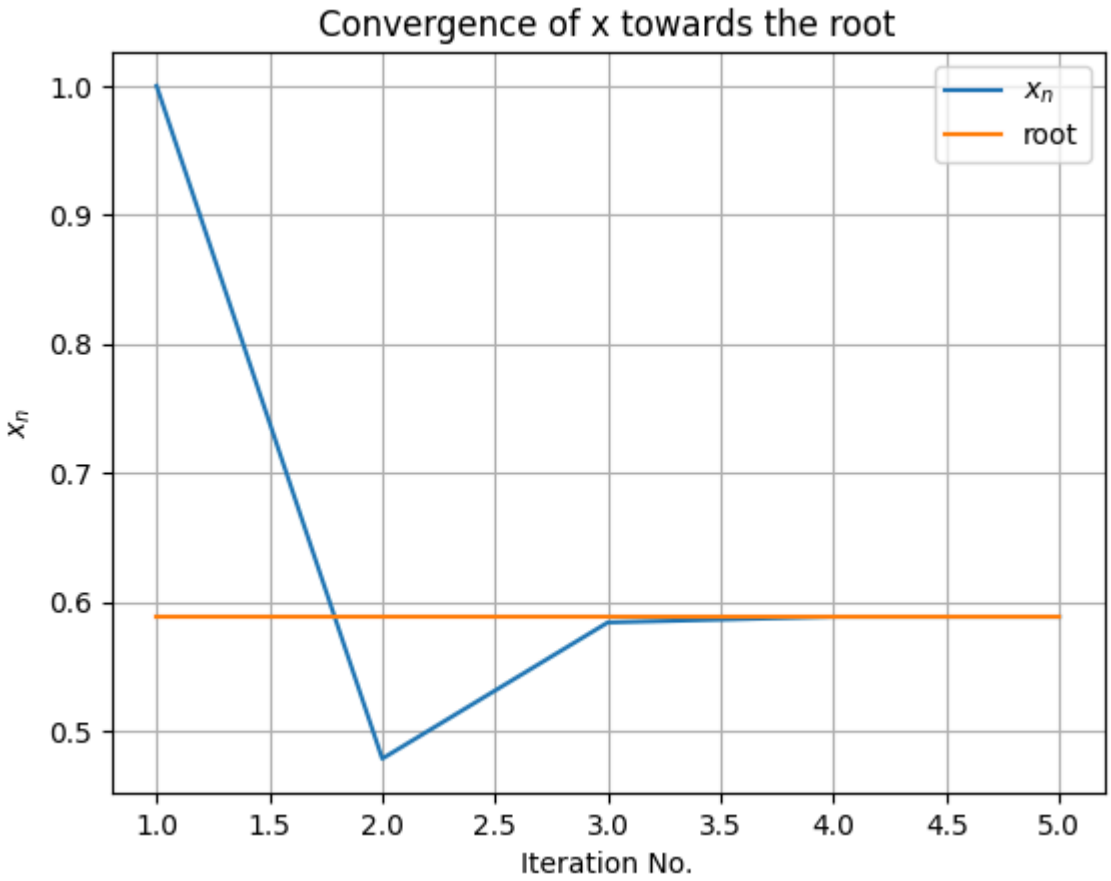
$$f(x) = e^{-x} - sin(x)$$

In [ ]:
```
def fun(x):
    return np.exp(-x) - np.sin(x)
def dfun(x):
    return -np.exp(-x) - np.cos(x)
df = newton_raphson(1,err)
df
```

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 0.478528 | 1.592222e-01 | 0.584157 | -5.214722e-01 | -4.114673e-01 |
| **1** | 2 | 0.584157 | 6.079306e-03 | 0.588525 | 1.056292e-01 | 1.100050e-01 |
| **2** | 3 | 0.588525 | 1.058452e-05 | 0.588533 | 4.368093e-03 | 4.375725e-03 |
| **3** | 4 | 0.588533 | 3.233336e-11 | 0.588533 | 7.631751e-06 | 7.631774e-06 |
| **4** | 5 | 0.588533 | 0.000000e+00 | 0.588533 | 2.331346e-11 | 2.331346e-11 |




Result:

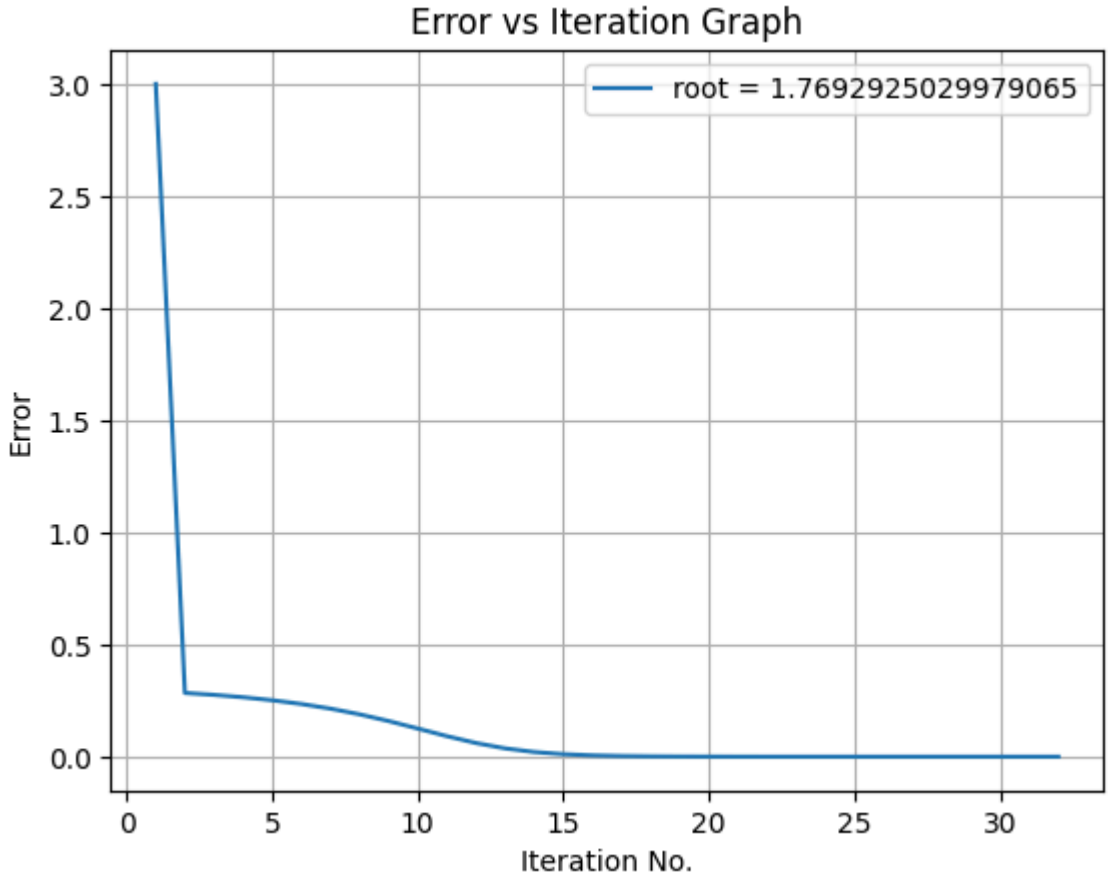
1. For the initial point 1,the root is 0.5885327439585476

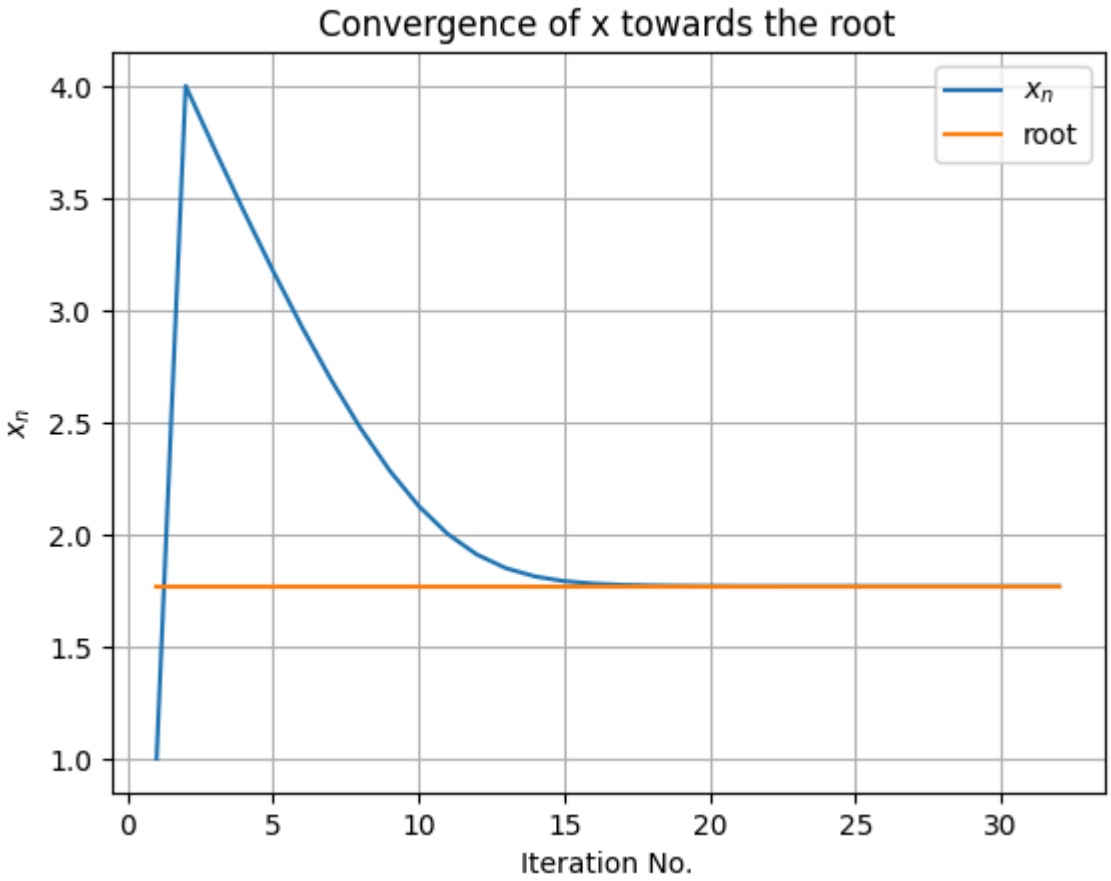
$$f(x) = x^3 - 2x - 2$$

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

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 4.000000 | 5.400000e+01 | 3.715789 | 3.000000e+00 | 7.692924e-01 |
| **1** | 2 | 3.715789 | 4.187267e+01 | 3.440153 | -2.842105e-01 | -2.230708e+00 |
| **2** | 3 | 3.440153 | 3.183272e+01 | 3.175188 | -2.756363e-01 | -1.946497e+00 |
| **3** | 4 | 3.175188 | 2.366128e+01 | 2.923565 | -2.649656e-01 | -1.670861e+00 |
| **4** | 5 | 2.923565 | 1.714127e+01 | 2.688642 | -2.516222e-01 | -1.405895e+00 |
| **5** | 6 | 2.688642 | 1.205835e+01 | 2.474487 | -2.349239e-01 | -1.154273e+00 |
| **6** | 7 | 2.474487 | 8.202531e+00 | 2.285726 | -2.141540e-01 | -9.193491e-01 |
| **7** | 8 | 2.285726 | 5.370425e+00 | 2.126958 | -1.887613e-01 | -7.051951e-01 |
| **8** | 9 | 2.126958 | 3.368340e+00 | 2.001586 | -1.587679e-01 | -5.164338e-01 |
| **9** | 10 | 2.001586 | 2.015878e+00 | 1.910193 | -1.253720e-01 | -3.576659e-01 |
| **10** | 11 | 1.910193 | 1.149597e+00 | 1.849400 | -9.139340e-02 | -2.322939e-01 |
| **11** | 12 | 1.849400 | 6.266639e-01 | 1.812486 | -6.079323e-02 | -1.409005e-01 |
| **12** | 13 | 1.812486 | 3.292345e-01 | 1.791730 | -3.691385e-02 | -8.010726e-02 |
| **13** | 14 | 1.791730 | 1.685280e-01 | 1.780684 | -2.075537e-02 | -4.319341e-02 |
| **14** | 15 | 1.780684 | 8.488641e-02 | 1.775001 | -1.104669e-02 | -2.243804e-02 |
| **15** | 16 | 1.775001 | 4.237047e-02 | 1.772134 | -5.682292e-03 | -1.139135e-02 |
| **16** | 17 | 1.772134 | 2.104740e-02 | 1.770702 | -2.867302e-03 | -5.709061e-03 |
| **17** | 18 | 1.770702 | 1.042947e-02 | 1.769990 | -1.432191e-03 | -2.841759e-03 |
| **18** | 19 | 1.769990 | 5.161629e-03 | 1.769638 | -7.116422e-04 | -1.409568e-03 |
| **19** | 20 | 1.769638 | 2.552949e-03 | 1.769463 | -3.526804e-04 | -6.979257e-04 |
| **20** | 21 | 1.769463 | 1.262303e-03 | 1.769377 | -1.745548e-04 | -3.452453e-04 |
| **21** | 22 | 1.769377 | 6.240497e-04 | 1.769334 | -8.633752e-05 | -1.706905e-04 |
| **22** | 23 | 1.769334 | 3.084906e-04 | 1.769313 | -4.269011e-05 | -8.435295e-05 |
| **23** | 24 | 1.769313 | 1.524925e-04 | 1.769303 | -2.110502e-05 | -4.166284e-05 |
| **24** | 25 | 1.769303 | 7.537839e-05 | 1.769297 | -1.043302e-05 | -2.055782e-05 |
| **25** | 26 | 1.769297 | 3.725987e-05 | 1.769295 | -5.157237e-06 | -1.012481e-05 |
| **26** | 27 | 1.769295 | 1.841764e-05 | 1.769294 | -2.549270e-06 | -4.967572e-06 |
| **27** | 28 | 1.769294 | 9.103859e-06 | 1.769293 | -1.260116e-06 | -2.418302e-06 |
| **28** | 29 | 1.769293 | 4.500043e-06 | 1.769293 | -6.228784e-07 | -1.158185e-06 |
| **29** | 30 | 1.769293 | 2.224372e-06 | 1.769293 | -3.078895e-07 | -5.353070e-07 |
| **30** | 31 | 1.769293 | 1.099508e-06 | 1.769292 | -1.521900e-07 | -2.274175e-07 |
| **31** | 32 | 1.769292 | 5.434867e-07 | 1.769292 | -7.522755e-08 | -7.522755e-08 |




Result:

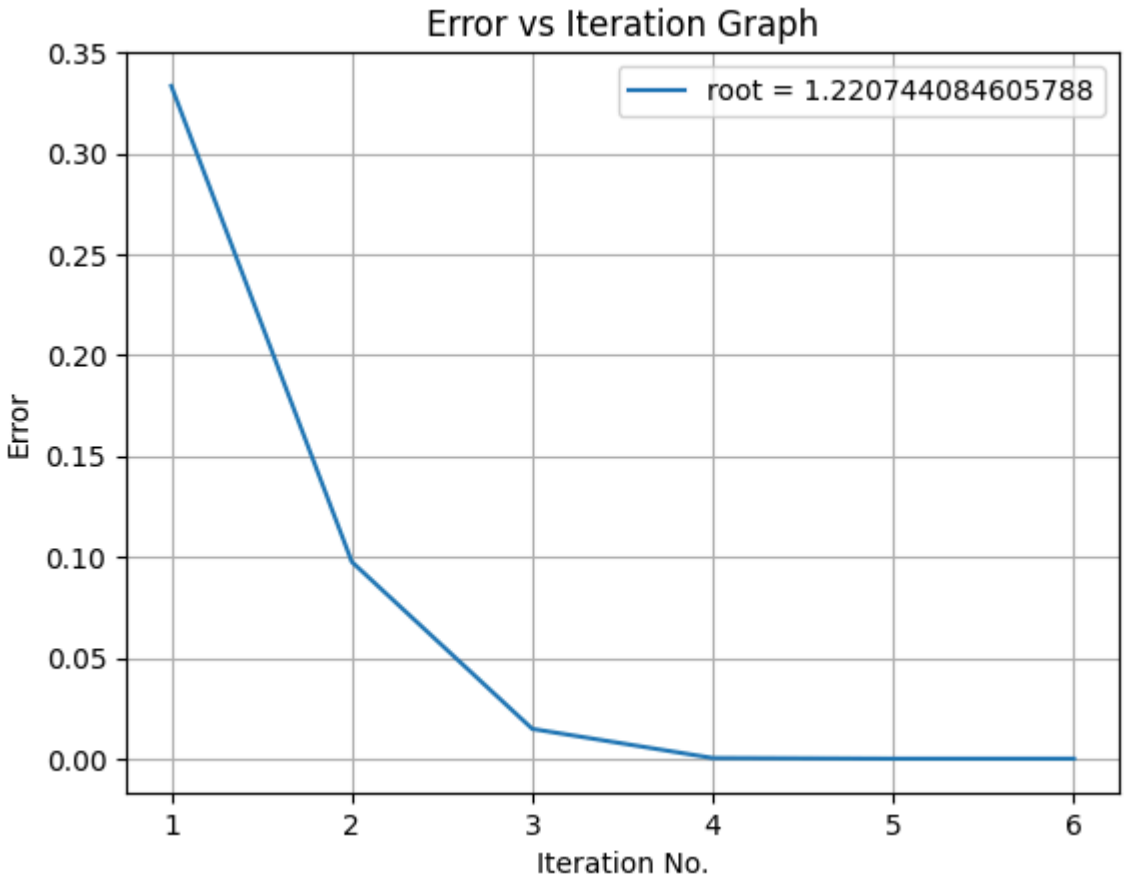
1. For the initial point 1,the root is 1.7692925029979065

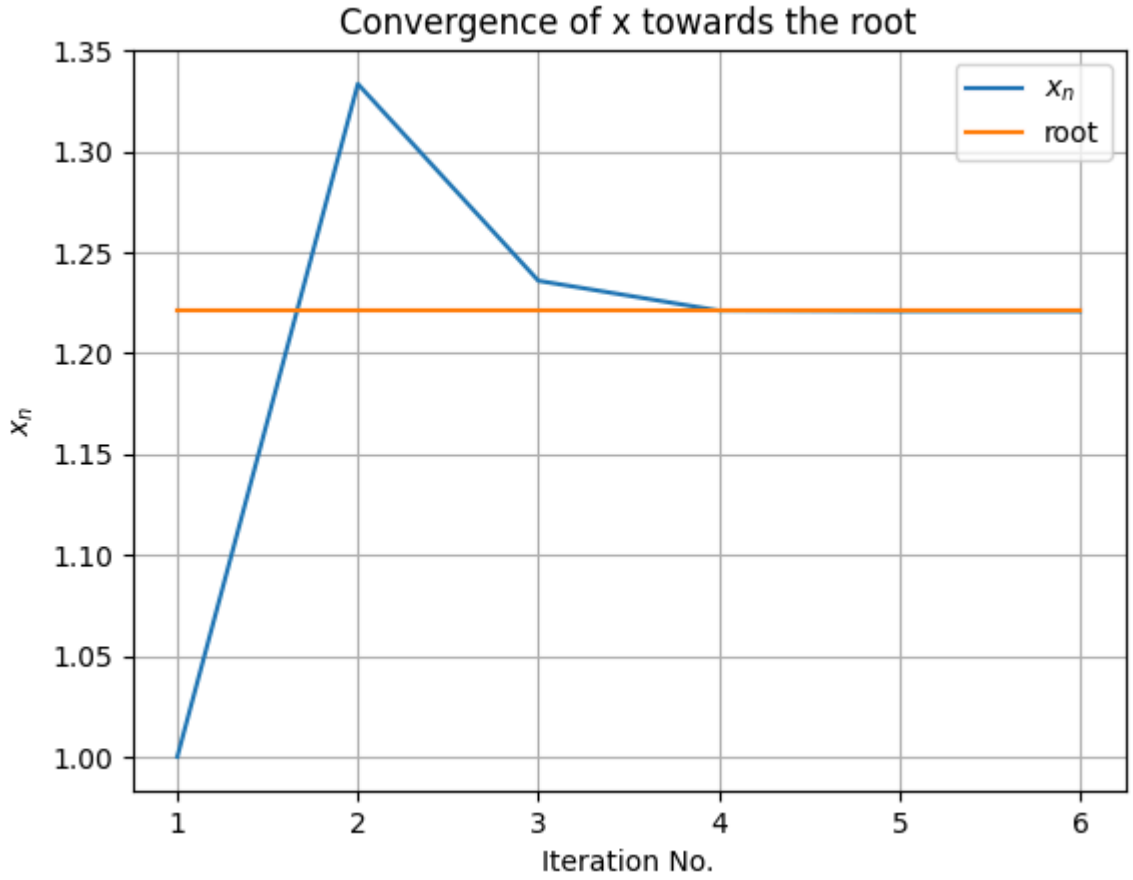
$$f(x) = x^4 - x - 1$$

In [ ]:
```
def fun(x):
    return x**4 - x - 1
def dfun(x):
    return 4*x**3 - 1
df = newton_raphson(1,err)
df
```

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 1.333333 | 8.271605e-01 | 1.235808 | 3.333333e-01 | 2.207441e-01 |
| **1** | 2 | 1.235808 | 9.659633e-02 | 1.221059 | -9.752547e-02 | -1.125892e-01 |
| **2** | 3 | 1.221059 | 1.977478e-03 | 1.220744 | -1.474887e-02 | -1.506378e-02 |
| **3** | 4 | 1.220744 | 8.862016e-07 | 1.220744 | -3.147685e-04 | -3.149097e-04 |
| **4** | 5 | 1.220744 | 1.789680e-13 | 1.220744 | -1.411893e-07 | -1.411893e-07 |
| **5** | 6 | 1.220744 | 4.440892e-16 | 1.220744 | -2.842171e-14 | -2.842171e-14 |






Result:

1. For the initial point 1,the root is 1.220744084605788

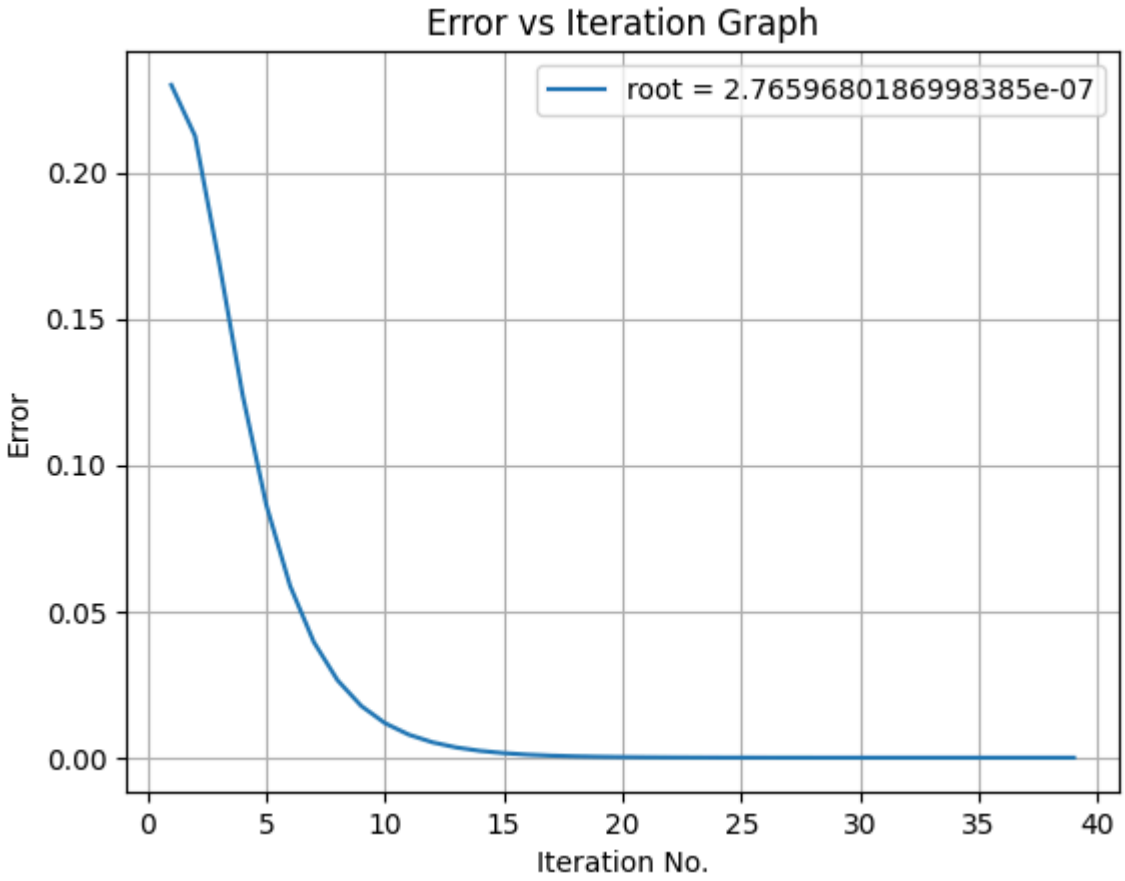
$$f(x) = tan(x) - x$$

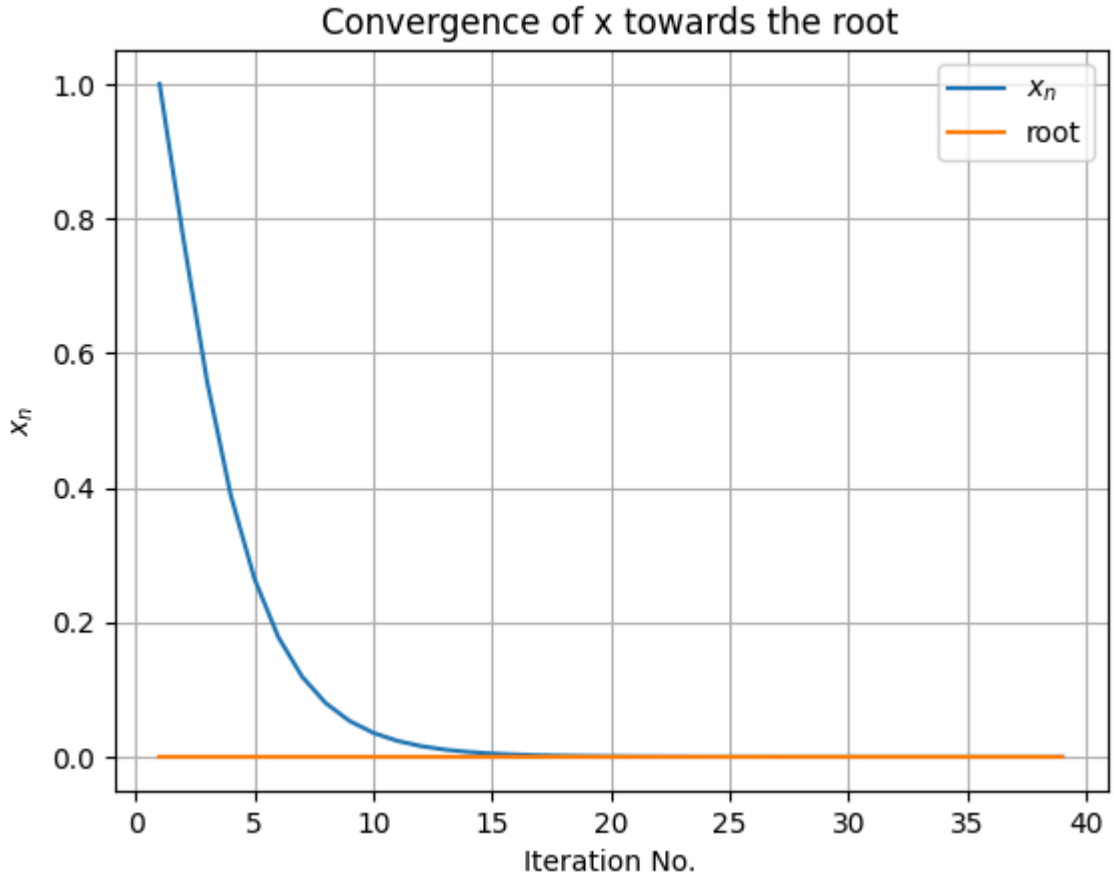
In [ ]:
```
def fun(x):
    return np.tan(x) - x
def dfun(x):
    return (1/(np.cos(x)**2))- 1
df = newton_raphson(1,err)
df
```

Out[ ]:

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **0** | 1 | 7.701903e-01 | 1.998473e-01 | 5.578066e-01 | -2.298097e-01 | -9.999998e-01 |
| **1** | 2 | 5.578066e-01 | 6.609158e-02 | 3.880140e-01 | -2.123837e-01 | -7.701901e-01 |
| **2** | 3 | 3.880140e-01 | 2.072125e-02 | 2.639827e-01 | -1.697926e-01 | -5.578064e-01 |
| **3** | 4 | 2.639827e-01 | 6.307930e-03 | 1.776401e-01 | -1.240314e-01 | -3.880139e-01 |
| **4** | 5 | 1.776401e-01 | 1.892427e-03 | 1.189273e-01 | -8.634258e-02 | -2.639825e-01 |
| **5** | 6 | 1.189273e-01 | 5.638804e-04 | 7.943465e-02 | -5.871283e-02 | -1.776399e-01 |
| **6** | 7 | 7.943465e-02 | 1.674967e-04 | 5.300103e-02 | -3.949260e-02 | -1.189271e-01 |
| **7** | 8 | 5.300103e-02 | 4.968439e-05 | 3.534726e-02 | -2.643362e-02 | -7.943447e-02 |
| **8** | 9 | 3.534726e-02 | 1.472865e-05 | 2.356877e-02 | -1.765377e-02 | -5.300084e-02 |
| **9** | 10 | 2.356877e-02 | 4.365015e-06 | 1.571367e-02 | -1.177849e-02 | -3.534707e-02 |
| **10** | 11 | 1.571367e-02 | 1.293466e-06 | 1.047613e-02 | -7.855091e-03 | -2.356858e-02 |
| **11** | 12 | 1.047613e-02 | 3.832659e-07 | 6.984187e-03 | -5.237547e-03 | -1.571349e-02 |
| **12** | 13 | 6.984187e-03 | 1.135625e-07 | 4.656155e-03 | -3.491940e-03 | -1.047594e-02 |
| **13** | 14 | 4.656155e-03 | 3.364843e-08 | 3.104112e-03 | -2.328032e-03 | -6.984003e-03 |
| **14** | 15 | 3.104112e-03 | 9.969945e-09 | 2.069411e-03 | -1.552043e-03 | -4.655971e-03 |
| **15** | 16 | 2.069411e-03 | 2.954063e-09 | 1.379608e-03 | -1.034701e-03 | -3.103928e-03 |
| **16** | 17 | 1.379608e-03 | 8.752785e-10 | 9.197390e-04 | -6.898029e-04 | -2.069226e-03 |
| **17** | 18 | 9.197390e-04 | 2.593419e-10 | 6.131594e-04 | -4.598691e-04 | -1.379423e-03 |
| **18** | 19 | 6.131594e-04 | 7.684205e-11 | 4.087729e-04 | -3.065796e-04 | -9.195543e-04 |
| **19** | 20 | 4.087729e-04 | 2.276802e-11 | 2.725153e-04 | -2.043864e-04 | -6.129747e-04 |
| **20** | 21 | 2.725153e-04 | 6.746079e-12 | 1.816769e-04 | -1.362576e-04 | -4.085883e-04 |
| **21** | 22 | 1.816769e-04 | 1.998838e-12 | 1.211179e-04 | -9.083843e-05 | -2.723306e-04 |
| **22** | 23 | 1.211179e-04 | 5.922484e-13 | 8.074527e-05 | -6.055896e-05 | -1.814922e-04 |
| **23** | 24 | 8.074527e-05 | 1.754810e-13 | 5.383018e-05 | -4.037264e-05 | -1.209332e-04 |
| **24** | 25 | 5.383018e-05 | 5.199437e-14 | 3.588679e-05 | -2.691509e-05 | -8.056059e-05 |
| **25** | 26 | 3.588679e-05 | 1.540574e-14 | 2.392453e-05 | -1.794339e-05 | -5.364550e-05 |
| **26** | 27 | 2.392453e-05 | 4.564664e-15 | 1.594968e-05 | -1.196226e-05 | -3.570210e-05 |
| **27** | 28 | 1.594968e-05 | 1.352491e-15 | 1.063312e-05 | -7.974845e-06 | -2.373984e-05 |
| **28** | 29 | 1.063312e-05 | 4.007381e-16 | 7.088754e-06 | -5.316558e-06 | -1.576500e-05 |
| **29** | 30 | 7.088754e-06 | 1.187379e-16 | 4.725833e-06 | -3.544368e-06 | -1.044844e-05 |
| **30** | 31 | 4.725833e-06 | 3.518151e-17 | 3.150551e-06 | -2.362922e-06 | -6.904070e-06 |
| **31** | 32 | 3.150551e-06 | 1.042401e-17 | 2.100385e-06 | -1.575282e-06 | -4.541148e-06 |
| **32** | 33 | 2.100385e-06 | 3.088706e-18 | 1.400249e-06 | -1.050166e-06 | -2.965866e-06 |

| | iter | $x_n$ | $f(x_n)$ | $x_{n+1}$ | $x_n - x_{n-1}$ | $a - x_{n-1}$ |
|---|---|---|---|---|---|---|
| **33** | 34 | 1.400249e-06 | 9.152191e-19 | 9.334563e-07 | -7.001356e-07 | -1.915700e-06 |
| **34** | 35 | 9.334563e-07 | 2.711564e-19 | 6.222484e-07 | -4.667928e-07 | -1.215564e-06 |
| **35** | 36 | 6.222484e-07 | 8.036225e-20 | 4.147258e-07 | -3.112079e-07 | -7.487716e-07 |
| **36** | 37 | 4.147258e-07 | 2.376986e-20 | 2.765968e-07 | -2.075226e-07 | -4.375636e-07 |
| **37** | 38 | 2.765968e-07 | 7.040961e-21 | 1.846847e-07 | -1.381290e-07 | -2.300410e-07 |
| **38** | 39 | 1.846847e-07 | 2.091113e-21 | 1.235319e-07 | -9.191209e-08 | -9.191209e-08 |






Result:

1. For the initial point 1,the root is 2.7659680186998385e-07

# Q4)

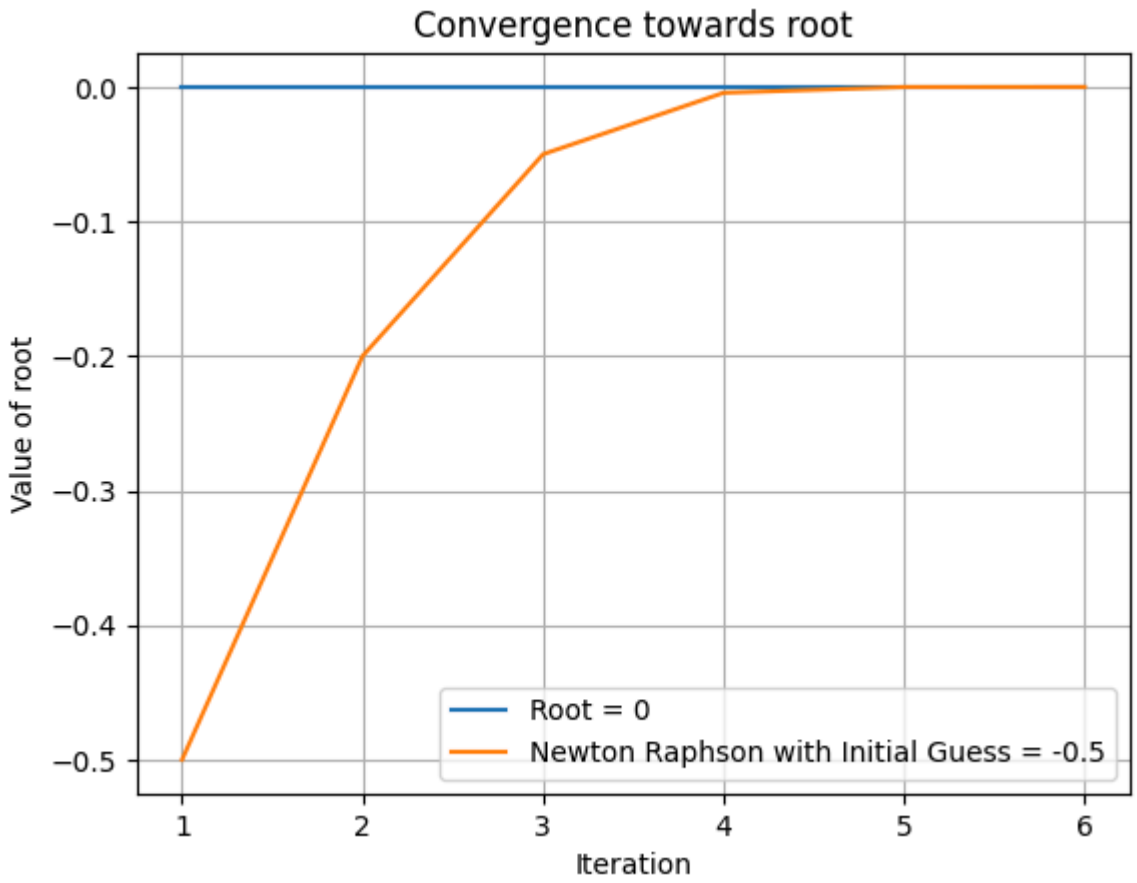
$$f(x) = a + x(x-1)^2$$

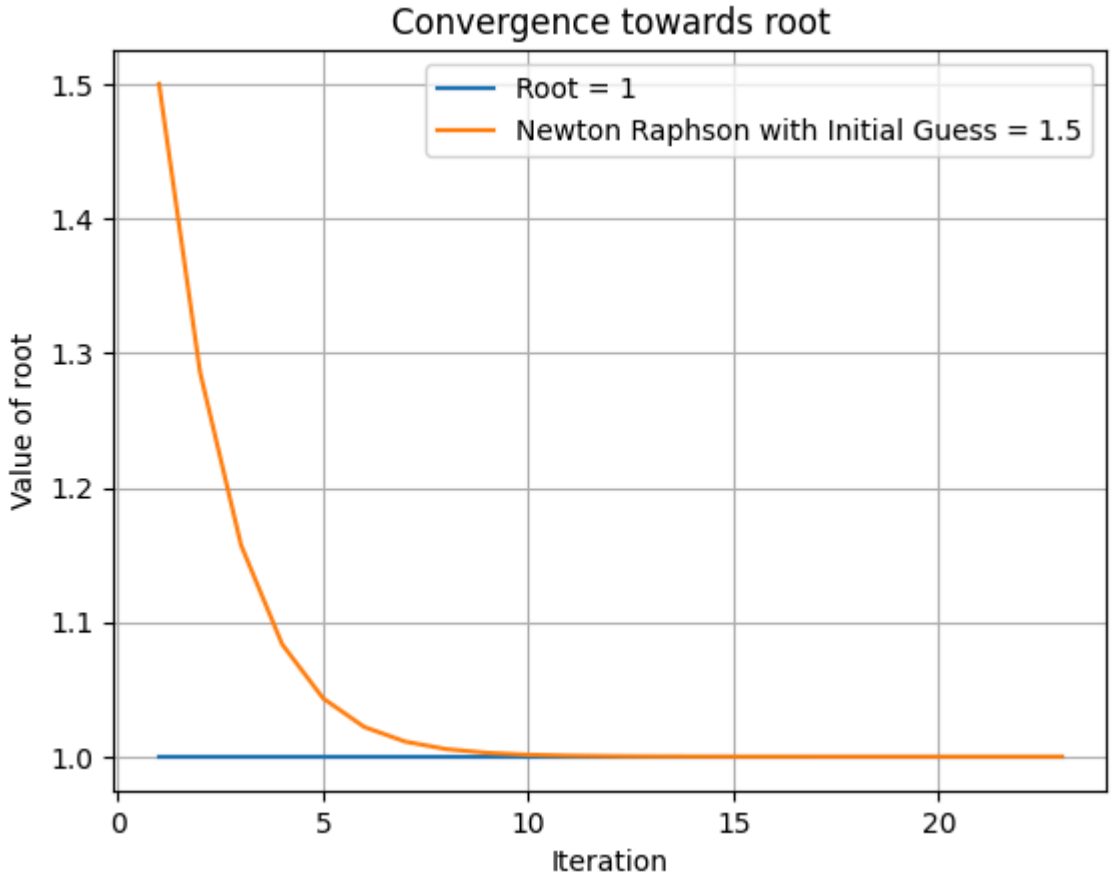
In [ ]:
```
def newton_raphson(a,x0,err):
    roots = []
    present = x0
    fpresent = fun(a,x0)
    dfpresent = dfun(x0)
    next = present-(fpresent/dfpresent)
    roots.append(present)
    i = 1
    while(abs(next - present) > err):
        prev = present
        present = next
        fpresent = fun(a,present)
        dfpresent = dfun(present)
        next = present-(fpresent/dfpresent)
        roots.append(present)
        i += 1
    return roots

def fun(a,x):
    return a + x*((x-1)**2)
```

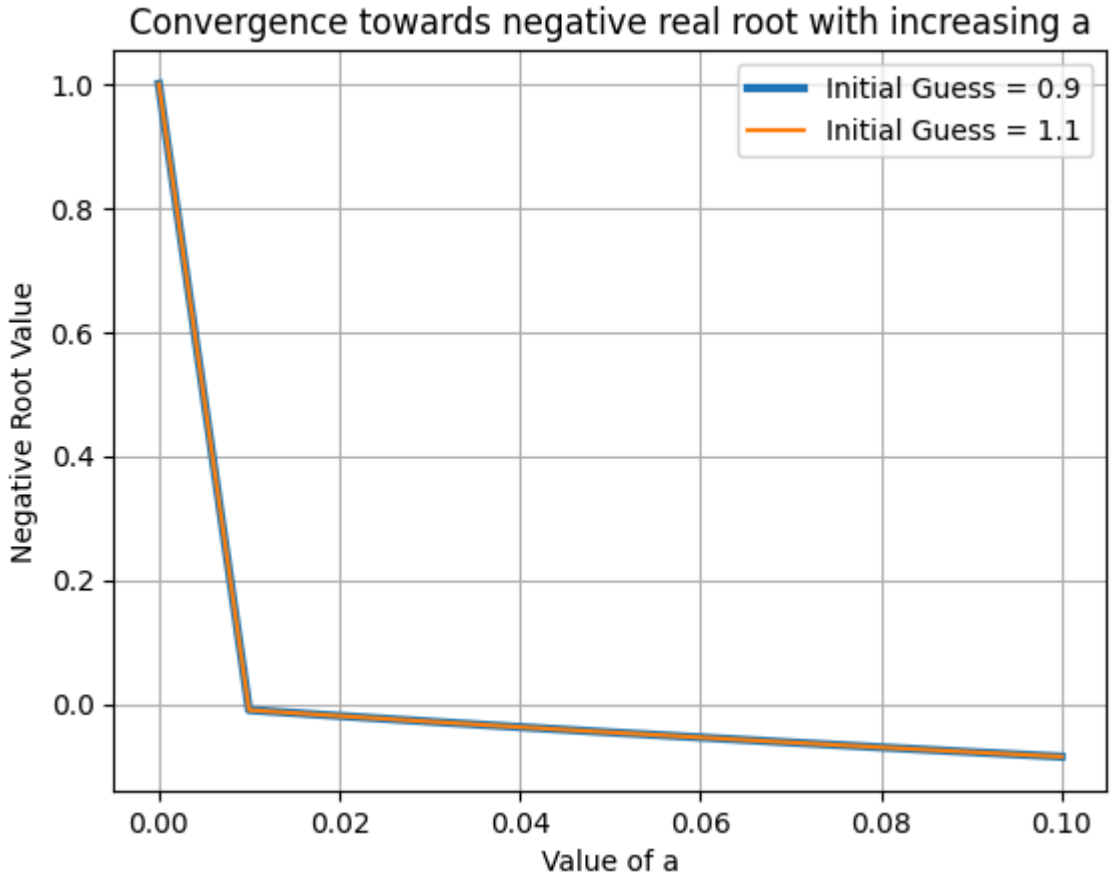
```
def dfun(x):
    return (x-1)**2 + 2*x*(x-1)

root0 = newton_raphson(0,-0.5,err)
root1 = newton_raphson(0,1.5,err)
r0 = [0]*len(root0)
r1 = [1]*len(root1)
iter0 = np.arange(1,len(root0)+1,1)
iter1 = np.arange(1,len(root1)+1,1)
plt.figure(1)
plt.plot(iter0,r0,label = 'Root = 0')
plt.plot(iter0,root0,label = 'Newton Raphson with Initial Guess = -0.5')
plt.legend()
plt.grid(True)
plt.xlabel('Iteration')
plt.ylabel('Value of root')
plt.title('Convergence towards root')
plt.plot()

plt.figure(2)
plt.plot(iter1,r1,label = 'Root = 1')
plt.plot(iter1,root1,label = 'Newton Raphson with Initial Guess = 1.5')
plt.legend()
plt.grid(True)
plt.xlabel('Iteration')
plt.ylabel('Value of root')
plt.title('Convergence towards root')
plt.plot()
a_val = [0,0.01,0.02,0.03,0.04,0.05,0.07,0.08,0.09,0.1]
roots0 = []
iter0 = []
roots1 = []
iter1 = []

for a in a_val:
    x = newton_raphson(a,0.99,err)
    roots0.append(x[-1])
    iter0.append(len(x))

for a in a_val:
    x = newton_raphson(a,1.01,err)
    roots1.append(x[-1])
    iter1.append(len(x))

plt.figure(3)
plt.plot(a_val,roots0,label = 'Initial Guess = ' + str(0.9),linewidth = 3)
plt.plot(a_val,roots1,label = 'Initial Guess = ' + str(1.1))
plt.legend()
plt.grid(True)
plt.xlabel('Value of a')
plt.ylabel('Negative Root Value')
plt.title('Convergence towards negative real root with increasing a')
plt.plot()

plt.figure(4)
plt.plot(a_val,iter0,label = 'Initial Guess = ' + str(0.9))
plt.plot(a_val,iter1,label = 'Initial Guess = ' + str(1.1))
plt.legend()
plt.grid(True)
plt.xlabel('a')
```

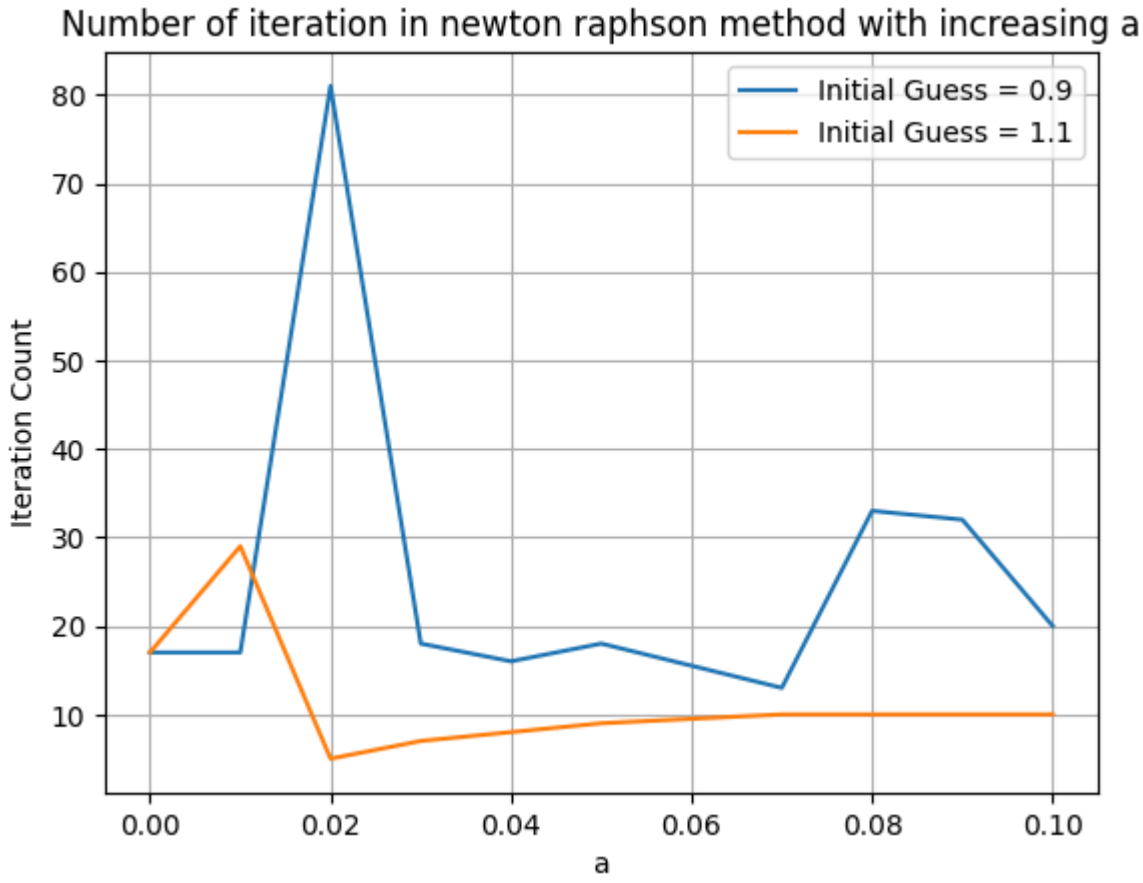
```
plt.ylabel('Iteration Count')
plt.title('Number of iteration in newton raphson method with increasing a')
plt.plot()
```

Out[ ]: []






Result:

1. We notice, with increasing value of a , we can see the convergence towards the negative root of the function.

2. We also notice the relation between a and number of iterations to find the root are directly proportional.