In [3]:

```
#Question 1

import numpy as np
from math import *

def fprime(y,N):
    fpy = N*y**(N-1)
    return fpy


def f(y,x,N):
    fy = (y**N - x)
    return fy

def main(x,N):
    yi = 2.0
    err_stop = 1e-7
    rel_err = 1.1 * err_stop
    max_iter = 200
    for i in range(0, max_iter):
        yi_plus_1 = yi-f(yi,x,N)/fprime(yi,N)
        # calc rel_err here and compare to err_stop
        # if rel_err is less than rel_err stop the loop now
        rel_err = abs((yi_plus_1-yi)/yi_plus_1)
        if rel_err <= err_stop:
            break
        yi = yi_plus_1
    print(yi_plus_1, i, rel_err)
main(2,2)
```

```
1.4142135623730951 4 1.1276404038266872e-12
```

In [2]:

```
#Question 2

import numpy as np
from math import *


def my_fixed_point(f,g,tol,max_iter):
    F = lambda x: f(x) - g(x)
    a = 0.8  # define and a and b (bracketing values)
    b = 0.9
    x_start = a
    x_end = b
    tol = 1e-7
    root_data = [a,b]
    a_b=[a,b]
    if F(a) * F(b) < 0:
        rel_err = 1.1 * tol
        max_iter = 200
        xr_old = (a + b) / 2
        for i in range(0, max_iter):
            xr = (a + b) / 2
            root_data.append(xr)
            if F(a) * F(xr) < 0:
                b = xr
            else:
                a = xr
            if i > 0:
                # calc rel_err here and compare to err_stop
                # if rel_err is less than rel_err stop the loop now
                rel_err = abs((xr-xr_old)/xr)
                if rel_err <= tol:
                    break
            else:
                xr_old = xr
                a_b.append(a)
                a_b.append(b)
    else:
        print("Your a and b values do not bracket the root")
        return []

    return xr_old


f = lambda x: x - 2.0 * exp(-x)
g = lambda x: 0
tol = 1e-7
max_iter = 200
print(my_fixed_point(f,g,tol,max_iter))
```

```
0.8500000000000001
```

In [4]:

```
#Question 5
import numpy as np
from math import *

def my_newton(f,df,x_o,tol):
    R = []
    E = []
    R.append(x_o)
    E.append(abs(f(R[-1])))
    max_iter = 200
    i = 0
    while i < tol:
        R.append(x_o-f(x_o)/df(x_o))
        E.append(abs(f(R[-1])))
        if E[-1] <= tol:
            return [R,E]
        x_o = R[-1]
        i = i + 1

    return [R,E]


#Test 1
f = lambda x: x**2 - 2
df = lambda x: 2*x
[R, E] = my_newton(f, df, 1, 1e-7)
print([R,E])
```

```
[[1, 1.5], [1, 0.25]]
```

In [3]:

```
#SP1
from math import *
import numpy as np

def f(x):
    f = (1.2*x**3) + (2*x**2) - (20*x) - 10
    return f

def Secant():
    x_i = -4.0
    x_mi = -5.0
    err_stop = 1e-7
    rel_err = 1.1 * err_stop
    max_iter = 200
    for i in range(0, max_iter):
        xi_plus_1 = x_i - ((f(x_i)*(x_mi - x_i))/(f(x_mi) - f(x_i)))
        rel_err = abs((xi_plus_1-x_i)/xi_plus_1)
        if rel_err <= err_stop:
            break
        x_mi = x_i
        x_i = xi_plus_1

    return x_i

print("x =", Secant() )
```

```
x = -4.7855156826748075
```

In [5]:

```
#SP2

#1.  x^3 - x -  exp(x) - 2 = 0;

#g_1(x)  = x^3 - x - exp(x) - 2  (1)

#g_2(x)  = (x + exp(x) + 2)^(1/3) (2)

#g_3(x)  = log(x^3 - x - 2) (3)

#g_4(x) = (exp(x) + 2)/(x^2 -1) (4)

#Test Cases: x1 = 2 and x2 = 3

#2.

#g_2(x)  = (x + exp(x) + 2)^(1/3) (2)

#g'_2(x) = (exp(x) + 1)/3*(exp(x) + x + 2)^(2/3)

#g'_2(2) = .5524,  g'_2(3) = .8202

#|g_2(x)| < 1, use g_2(x)

#3.

from math import *
import numpy as np

def f(x):
    f = pow(x,3) - x - exp(x) -2
    return f

def g(x):
    f = pow((x+exp(x)+2), (1/3))
    return f

xold = 2.0     #initial guess of x
root_approx = [xold]

err_stop = 1e-7     #adjust as needed...
rel_err = 1.1*err_stop  #set value large enough to start process
count = 0
max_iter = 5


for i in range(0,max_iter):
    count+=1
    f = g(xold)
    xnew = f
    root_approx.append(xnew)
    if count > 1:
        rel_err = abs((xnew-xold)/xnew)
        if rel_err <= err_stop:
            break
    xold = xnew


print("x = ", xold)
```

```
x =  2.5862258027406324
```

In [2]:

```
#SP3
from math import *
import numpy as np

def f(x):
    f = x - 2*exp(-x)
    return f

def Secant():
    x_i = 1
    x_mi = 0
    err_stop = 1e-7
    rel_err = 1.1 * err_stop
    max_iter = 200
    for i in range(0, max_iter):
        xi_plus_1 = x_i - ((f(x_i)*(x_mi - x_i))/(f(x_mi) - f(x_i)))
        rel_err = abs((xi_plus_1-x_i)/xi_plus_1)
        if rel_err <= err_stop:
            break
        x_mi = x_i
        x_i = xi_plus_1

    return x_i

print("x =", Secant() )
```

```
x = 0.852605503703827
```

In [11]:

```
#SP4

#1.  x^2 - 5x^(1/3) + 1 = 0;

#g_1(x)  = ((x^2 -1)/(5))^(3)  (1)

#g_2(x)  = sqrt(5x^(1/3) +1) (2)

#Test Cases: x1 = 2 and x2 = 2.5

#2.

#g_2(x)  = sqrt(5x^(1/3) +1) (2)

#g'_2(x) = 5*(1/((6*sqrt(5x^(1/3) +1))*x^(2/3))

#g'_2(2) =.1943 ,  g'_2(2.5) = .1621

#|g_2(x)| < 1, use g_2(x)

#3.

from math import *
import numpy as np

def f(x):
    f = pow(x,2) - (5*pow(x,(1/3)))  + 1
    return f

def g(x):
    f = sqrt(5*pow(x,(1/3)) + 1)
    return f

xold = 2     #initial guess of x
root_approx = [xold]

err_stop = 1e-7     #adjust as needed...
rel_err = 1.1*err_stop  #set value large enough to start process
count = 0
max_iter = 5


for i in range(0,max_iter):
    count+=1
    f = g(xold)
    xnew = f
    root_approx.append(xnew)
    if count > 1:
        rel_err = abs((xnew-xold)/xnew)
        if rel_err <= err_stop:
            break
    xold = xnew


print("x = ", xold)
```

```
x =  2.843045765247176
```

In [ ]: