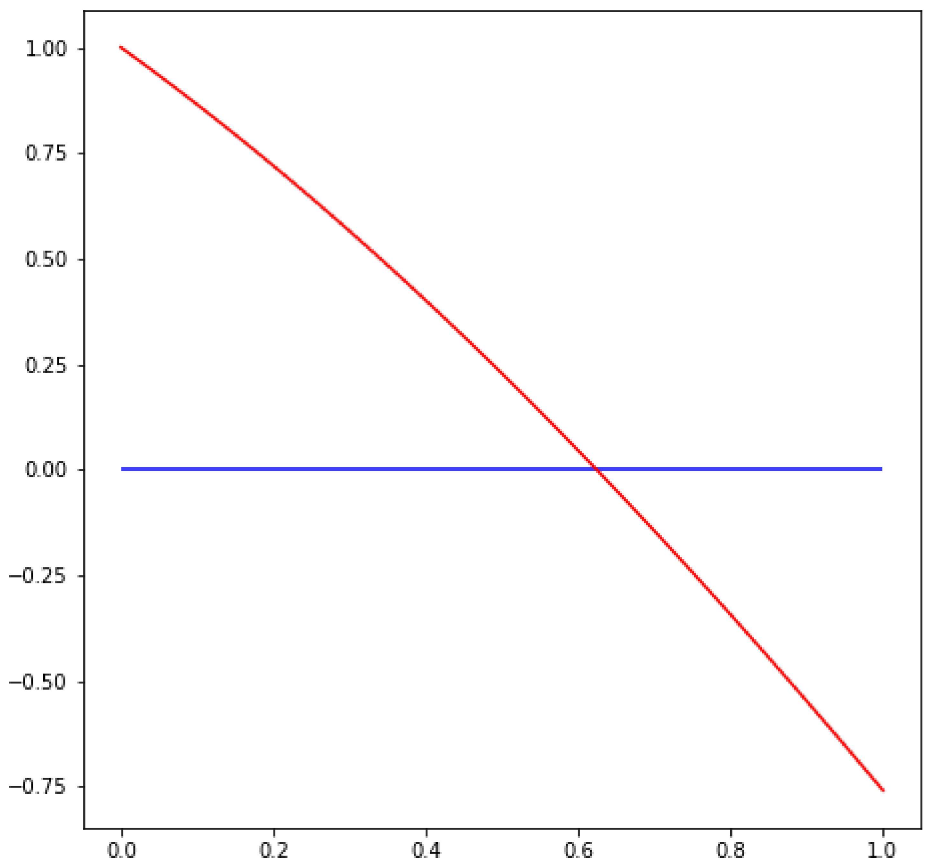
Lab Task 1: Plot all the given functions to observe the roots by visualization, fill the table by your visual guess of root. We have plotted one function for you.

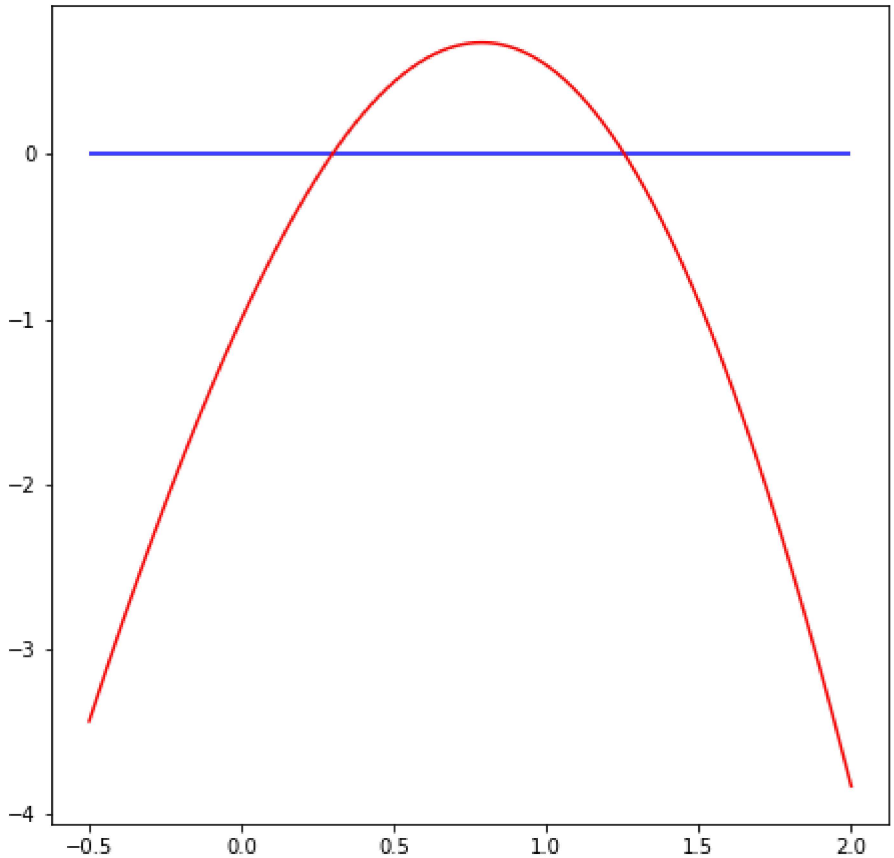
1) $f(x) = cos(x) - 1.3x$
2) $f(x) = xcos(x) - 2x^2 + 3x - 1$
3) $f(x) = 2xcos(2x) - (x+1)^2$

```
import numpy as np
from matplotlib import pyplot as plt

plt.rcParams["figure.figsize"] = [7.50, 7.50]

def f1(x):
    return (np.cos(x)- 1.3*x)

x = np.linspace(0,1, 1000)
plt.plot(x,f1(x), color='red')
plt.hlines(y=0,xmin=0,xmax=1,color='blue')
plt.show()
```

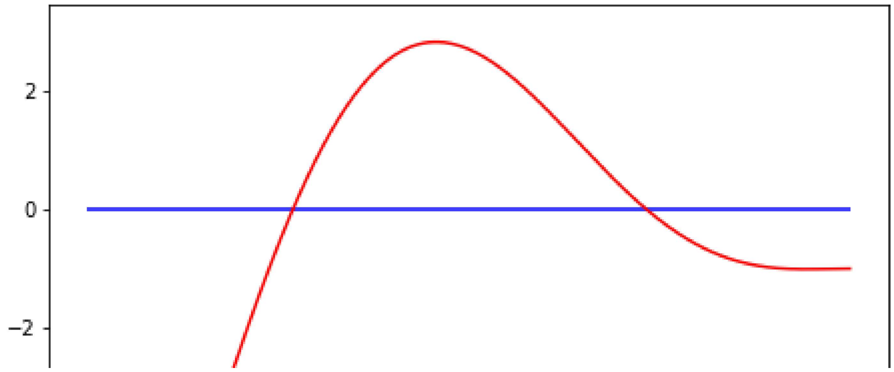



```
def f2(x):
    return (x*np.cos(x) - 2*(x**2) + 3*x - 1)
```

```
x = np.linspace(-0.5, 2, 1000)
plt.plot(x,f2(x), color='red')
plt.hlines(y=0,xmin=-0.5,xmax=2,color='blue')
plt.show()
```




```
def f3(x):
    return (2*x*np.cos(2*x) - (x+1)**2)

x = np.linspace(-3, 0, 1000)
plt.plot(x,f3(x), color='red')
plt.hlines(y=0,xmin=-3,xmax=0,color='blue')
plt.show()
```



Lab Task 2: Complete the missing code of bisection method accordding to the explained algorithm and find root of given problems by bisection method according to the instructions given in table.

1) $f1(x) = cos(x) - 1.3x$
2) $f2(x) = xcos(x) - 2x^2 + 3x - 1$
3) $f3(x) = 2xcos(2x) - (x+1)^2$

```
import numpy as np
from tabulate import tabulate

## module Bisection
''' root = bisection(func, x1, x2, tol=0.001, max_iter=100):.
    Finds a root of f(x) = 0 by bisection.
    The root must be bracketed in (x1,x2).

'''

def bisection(func, x1, x2, tol=0.00001, max_iter=100):
    if func(x1) * func(x2) >= 0:
        return  "Error: Choose different interval, function should have different signs at th
    data=[]
    iter = 0
    xr = x2
    error = tol + 1

    while iter < max_iter and error > tol:
        xrold = xr
        xr = ((x1+x2)/2)
        iter += 1
        error = abs((xr - xrold) )

        test = func(x1) * func(xr)
        # write your code here to replace value of x1 or x2 by xr
        if test <0:
          x2 = xr
        elif test>0:
          x1 = xr
        else:
          error=0
```

```
        data.append([iter+1,x1,func(x1),x2,func(x2),xr,func(xr),error])
    print(tabulate(data,headers=['#','x1','f(x1)','x2','f(x2)','xr','f(xr)',"error"],tablefmt
    print('\nRoot of given function is x=%.9f in n=%d number of iterations with a tolerence=%

    return
```

```
bisection(f1, 0, 1)
```

```
|   # |       x1 |        f(x1) |       x2 |         f(x2) |       xr |         f(xr) |
|-----|----------|--------------|----------|---------------|----------|---------------|----
|   2 | 0.5      | 0.227583     | 1        | -0.759698     | 0.5      |  0.227583     | 0.5
|   3 | 0.5      | 0.227583     | 0.75     | -0.243311     | 0.75     | -0.243311     | 0.2
|   4 | 0.5      | 0.227583     | 0.625    | -0.00153688   | 0.625    | -0.00153688   | 0.1
|   5 | 0.5625   | 0.114674     | 0.625    | -0.00153688   | 0.5625   |  0.114674     | 0.0
|   6 | 0.59375  | 0.0569735    | 0.625    | -0.00153688   | 0.59375  |  0.0569735    | 0.0
|   7 | 0.609375 | 0.0278184    | 0.625    | -0.00153688   | 0.609375 |  0.0278184    | 0.0
|   8 | 0.617188 | 0.0131656    | 0.625    | -0.00153688   | 0.617188 |  0.0131656    | 0.0
|   9 | 0.621094 | 0.00582059   | 0.625    | -0.00153688   | 0.621094 |  0.00582059   | 0.0
|  10 | 0.623047 | 0.0021434    | 0.625    | -0.00153688   | 0.623047 |  0.0021434    | 0.0
|  11 | 0.624023 | 0.000303648  | 0.625    | -0.00153688   | 0.624023 |  0.000303648  | 0.0
|  12 | 0.624023 | 0.000303648  | 0.624512 | -0.00061652   | 0.624512 | -0.00061652   | 0.0
|  13 | 0.624023 | 0.000303648  | 0.624268 | -0.000156412  | 0.624268 | -0.000156412  | 0.0
|  14 | 0.624146 | 7.36243e-05  | 0.624268 | -0.000156412  | 0.624146 |  7.36243e-05  | 0.0
|  15 | 0.624146 | 7.36243e-05  | 0.624207 | -4.13921e-05  | 0.624207 | -4.13921e-05  | 6.1
|  16 | 0.624176 | 1.61164e-05  | 0.624207 | -4.13921e-05  | 0.624176 |  1.61164e-05  | 3.0
|  17 | 0.624176 | 1.61164e-05  | 0.624191 | -1.26378e-05  | 0.624191 | -1.26378e-05  | 1.5
|  18 | 0.624184 | 1.73937e-06  | 0.624191 | -1.26378e-05  | 0.624184 |  1.73937e-06  | 7.6

Root of given function is x=0.624183655 in n=17 number of iterations with a tolerence=0
```

```
bisection(f2, 0, 0.6)
```

```
|   # |       x1 |          f(x1) |       x2 |        f(x2) |       xr |          f(xr) |
|-----|----------|----------------|----------|--------------|----------|----------------|----
|   2 | 0        | -1             | 0.3      | 0.00660095   | 0.3      |  0.00660095    | 0.3
|   3 | 0.15     | -0.446684      | 0.3      | 0.00660095   | 0.15     | -0.446684      | 0.1
|   4 | 0.225    | -0.206921      | 0.3      | 0.00660095   | 0.225    | -0.206921      | 0.0
|   5 | 0.2625   | -0.0968046     | 0.3      | 0.00660095   | 0.2625   | -0.0968046     | 0.0
|   6 | 0.28125  | -0.0442537     | 0.3      | 0.00660095   | 0.28125  | -0.0442537     | 0.0
|   7 | 0.290625 | -0.0186131     | 0.3      | 0.00660095   | 0.290625 | -0.0186131     | 0.0
|   8 | 0.295312 | -0.00595266    | 0.3      | 0.00660095   | 0.295312 | -0.00595266    | 0.0
|   9 | 0.295312 | -0.00595266    | 0.297656 | 0.000337524  | 0.297656 |  0.000337524   | 0.0
|  10 | 0.296484 | -0.00280422    | 0.297656 | 0.000337524  | 0.296484 | -0.00280422    | 0.0
|  11 | 0.29707  | -0.00123251    | 0.297656 | 0.000337524  | 0.29707  | -0.00123251    | 0.0
|  12 | 0.297363 | -0.000447286   | 0.297656 | 0.000337524  | 0.297363 | -0.000447286   | 0.0
|  13 | 0.29751  | -5.48292e-05   | 0.297656 | 0.000337524  | 0.29751  | -5.48292e-05   | 0.0
|  14 | 0.29751  | -5.48292e-05   | 0.297583 | 0.00014136   | 0.297583 |  0.00014136    | 7.3
|  15 | 0.29751  | -5.48292e-05   | 0.297546 | 4.32688e-05  | 0.297546 |  4.32688e-05   | 3.6
|  16 | 0.297528 | -5.77935e-06   | 0.297546 | 4.32688e-05  | 0.297528 | -5.77935e-06   | 1.8
|  17 | 0.297528 | -5.77935e-06   | 0.297537 | 1.87449e-05  | 0.297537 |  1.87449e-05   | 9.1

Root of given function is x=0.297537231 in n=16 number of iterations with a tolerence=0
```

```
bisection(f3, -2, -2.2)
```

```
|   # |       x1 |         f(x1) |       x2 |          f(x2) |       xr |          f(xr) |
|-----|----------|---------------|----------|----------------|----------|----------------|----
|   2 | -2.1     | 0.849095      | -2.2     | -0.0877354     | -2.1     |  0.849095      | 0.:
|   3 | -2.15    | 0.400936      | -2.2     | -0.0877354     | -2.15    |  0.400936      | 0.(
|   4 | -2.175   | 0.161489      | -2.2     | -0.0877354     | -2.175   |  0.161489      | 0.(
|   5 | -2.1875  | 0.0380755     | -2.2     | -0.0877354     | -2.1875  |  0.0380755     | 0.(
|   6 | -2.1875  | 0.0380755     | -2.19375 | -0.0245334     | -2.19375 | -0.0245334     | 0.(
|   7 | -2.19062 | 0.00684561    | -2.19375 | -0.0245334     | -2.19062 |  0.00684561    | 0.(
|   8 | -2.19062 | 0.00684561    | -2.19219 | -0.0088253     | -2.19219 | -0.0088253     | 0.(
|   9 | -2.19062 | 0.00684561    | -2.19141 | -0.000985195   | -2.19141 | -0.000985195   | 0.(
|  10 | -2.19102 | 0.00293137    | -2.19141 | -0.000985195   | -2.19102 |  0.00293137    | 0.(
|  11 | -2.19121 | 0.000973378   | -2.19141 | -0.000985195   | -2.19121 |  0.000973378   | 0.(
|  12 | -2.19121 | 0.000973378   | -2.19131 | -5.83576e-06   | -2.19131 | -5.83576e-06   | 9.7
|  13 | -2.19126 | 0.000483789   | -2.19131 | -5.83576e-06   | -2.19126 |  0.000483789   | 4.8
|  14 | -2.19128 | 0.000238981   | -2.19131 | -5.83576e-06   | -2.19128 |  0.000238981   | 2.4
|  15 | -2.1913  | 0.000116574   | -2.19131 | -5.83576e-06   | -2.1913  |  0.000116574   | 1.2
|  16 | -2.1913  | 5.53694e-05   | -2.19131 | -5.83576e-06   | -2.1913  |  5.53694e-05   | 6.1

Root of given function is x=-2.191302490 in n=15 number of iterations with a tolerence=0
```

Lab Task 3: Find root of given problems by Newton Raphson method according to the instructions given in table.

1) $f1(x) = cos(x) - 1.3x$
2) $f2(x) = xcos(x) - 2x^2 + 3x - 1$
3) $f3(x) = 2xcos(2x) - (x+1)^2$

```
import numpy as np
from tabulate import tabulate

## module Newton_Raphson
''' newton_raphson(func, dfunc, x0, tol=1e-4, max_iter=1000)
    Finds a root of f(x) = 0 by newton_raphson.
'''



def newton_raphson(func, dfunc, x0, tol=1e-5, max_iter=1000):
    xr = x0
    data=[]
    iter = 0
    error = tol + 1
    for i in range(max_iter):
        iter+=1
        fx = func(xr)
        dx = dfunc(xr)
```

```
        if abs(dx) < tol:
           raise Exception("Derivative is close to zero!")
        xrold=xr
        xr = xr - fx/dx
        error=abs(xr-xrold)
        data.append([iter,xr,func(xr),error])
        if error < tol:
           print(tabulate(data,headers=['Iteration','xr','f(xr)',"error"],tablefmt="github"))
           print('\nRoot of given function is x=%.9f in n=%d number of iterations with a tole
           return

    raise Exception("Max iterations reached")


def df1(x):
  return -np.sin(x) - 1.3


newton_raphson(f1, df1, -1)
```

```
|   Iteration |        xr |         f(xr) |       error |
|-------------|-----------|---------------|-------------|
|           1 |  3.01349  | -4.90935      | 4.01349     |
|           2 | -0.425025 |  1.46356      | 3.43852     |
|           3 |  1.22377  | -1.25079      | 1.64879     |
|           4 |  0.665474 | -0.0784929    | 0.558292    |
|           5 |  0.624538 | -0.000666076  | 0.0409365   |
|           6 |  0.624185 | -5.06647e-08  | 0.000353408 |
|           7 |  0.624185 | -3.33067e-16  | 2.68859e-08 |

Root of given function is x=0.624184578 in n=7 number of iterations with a tolerence=0.0
```

```
def df2(x):
  return (np.cos(x) -1 * (x * np.sin(x)) - 4 * x + 3)

newton_raphson(f2, df2, 0.2)
```

```
|   Iteration |       xr |         f(xr) |       error |
|-------------|----------|---------------|-------------|
|           1 | 0.290432 | -0.0191366    | 0.090432    |
|           2 | 0.297485 | -0.000120822  | 0.00705309  |
|           3 | 0.29753  | -4.9537e-09   | 4.51008e-05 |
|           4 | 0.29753  | -2.22045e-16  | 1.84928e-09 |

Root of given function is x=0.297530234 in n=4 number of iterations with a tolerence=0.0
```

```
def df3(x):
  return (2 * (np.cos(2*x) - 2 * x * np.sin(2*x))) -2 *(x+1)

newton_raphson(f3, df3, -2)
```

```
|   Iteration |       xr |         f(xr) |       error |
|-------------|----------|--------------|-------------|
|           1 | -2.2393  | -0.498306    | 0.239298    |
|           2 | -2.19285 | -0.0155015   | 0.0464458   |
|           3 | -2.19131 | -1.80995e-05 | 0.00154222  |
|           4 | -2.19131 | -2.48346e-11 | 1.80491e-06 |

Root of given function is x=-2.191308012 in n=4 number of iterations with a tolerence=0
```

Lab Task 4: Find root of given problems by using fsolve command of sympy.optimize

1) $f1(x) = cos(x) - 1.3x$
2) $f2(x) = xcos(x) - 2x^2 + 3x - 1$
3) $f3(x) = 2xcos(2x) - (x+1)^2$

```
from scipy import optimize
```

```
optimize.fsolve(f1, 0)
```

```
array([0.62418458])
```

```
optimize.fsolve(f2, 0)
```

```
array([0.29753023])
```

```
optimize.fsolve(f3, -2)
```

```
array([-2.19130801])
```

Lab Task 5: Write program of Secant and False Position method by altering above codes.

```
import numpy as np
from tabulate import tabulate

## module False Position
''' root = false_pos(func, x1, x2, tol=0.001, max_iter=100):.
    Finds a root of f(x) = 0 by False position.
    The root must be bracketed in (x1,x2).

'''

def false_pos(func, x1, x2, tol=0.00001, max_iter=100):
```

```
    if func(x1) * func(x2) >= 0:
        return  "Error: Choose different interval, function should have different signs at th
    data=[]
    iter = 0
    xr = x2
    error = tol + 1

    while iter < max_iter and error > tol:
        xrold = xr
        xr = x1 - ((func(x1)*(x2-x1))/(func(x2)-func(x1)))
        iter += 1
        error = abs((xr - xrold) )

        test = func(x1) * func(xr)
        # write your code here to replace value of x1 or x2 by xr
        if test <0:
          x2 = xr
        elif test>0:
          x1 = xr
        else:
          error=0
        data.append([iter+1,x1,func(x1),x2,func(x2),xr,func(xr),error])
    print(tabulate(data,headers=['#','x1','f(x1)','x2','f(x2)','xr','f(xr)',"error"],tablefmt
    print('\nRoot of given function is x=%.9f in n=%d number of iterations with a tolerence=%

    return


import numpy as np
from tabulate import tabulate

## module Secant
''' newton_raphson(func, dfunc, x0, tol=1e-4, max_iter=1000)
    Finds a root of f(x) = 0 by secant method.
'''



def secant_method(func, x1, x2, tol=1e-5, max_iter=1000):
    xr = x1
    data=[]
    iter = 0
    error = tol + 1
    for i in range(max_iter):
        iter+=1
        fx = func(xr)

        xrold=xr
        xr =  x1 - ((func(x1)*(x2-x1))/(func(x2)-func(x1)))
        x1= x2
        x2 = xr
        error=abs(xr-xrold)
        data.append([iter,xr,func(xr),error])
```

```
        if error < tol:
            print(tabulate(data,headers=['Iteration','xr','f(xr)',"error"],tablefmt="github"))
            print('\nRoot of given function is x=%.9f in n=%d number of iterations with a tole
            return

    raise Exception("Max iterations reached")
```

```
secant_method(f3,1,-2)
```

```
|   Iteration |       xr |          f(xr) |       error |
|-------------|----------|----------------|-------------|
|           1 | -1.24867 |   1.93491      | 2.24867     |
|           2 | -5.7869  | -29.2399       | 4.53823     |
|           3 | -1.53034 |   2.76941      | 4.25656     |
|           4 | -1.89861 |   2.20241      | 0.368272    |
|           5 | -3.32909 | -11.6202       | 1.43048     |
|           6 | -2.12654 |   0.616437     | 1.20256     |
|           7 | -2.18712 |   0.041876     | 0.0605806   |
|           8 | -2.19153 |  -0.0022666    | 0.00441532  |
|           9 | -2.19131 |   7.25471e-06  | 0.000226714 |
|          10 | -2.19131 |   1.24593e-09  | 7.2333e-07  |

Root of given function is x=-2.191308012 in n=10 number of iterations with a tolerence=6
```