```
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize

#deformed shape of beam
def y(x):
    #Beam properties:
    E = 29 * 10**4  #psi
    I = 723         #in^4
    w0 = 3000       #lbs/ft
    L = 15          #ft
    return (w0/(120*E*I*L)) * (-x**5 + (2 * L**2 * x**3) - (L**4 * x))

#analytically computed derivative of beam
def dydx(x):
    #Beam properties:
    E = 29 * 10**4  #psi
    I = 723         #in^4
    w0 = 3000       #lbs/ft
    L = 15          #ft
    return (w0/(120*E*I*L)) * (-5 * x**4 + (6 * L**2 * x**2) - (L**4))

#analytically computed second derivative of beam
def d2ydx2(x):
    #Beam properties:
    E = 29 * 10**4  #psi
    I = 723         #in^4
    w0 = 3000       #lbs/ft
    L = 15          #ft
    return (w0/(120*E*I*L)) * (-20 * x**3 + (12 * L**2 * x))

#One iteration of Newton-Raphson method
def newtIter( func, dfunc, x1):
    return x1 - (func(x1)/dfunc(x1))

def newtonRaphson( func, dfunc, x1, tolerance= 1 * 10**-5, maxIterations=10):
    nIterations = 0
    xn = x1
    while (np.abs(func(xn)) > tolerance) and (nIterations < maxIterations):
         xn = newtIter( func, dfunc, xn)
         nIterations += 1
    return [xn, nIterations]

x = np.linspace(start=0, stop=15, num=100)
plt.figure()
plt.plot(x, y(x),label="Profile" )
#plt.plot(x, dydx(x),label="First derivative" )
plt.legend()
plt.xlabel("x")
plt.ylabel("y")
plt.title("Profile of rod")
plt.grid(True)
#plt.show()

# Newton-Raphson result
tolerance = 1 * 10**-8
root, iterations = newtonRaphson(dydx, d2ydx2, 7.5, tolerance)
print(f'Zero estimated at: {root}\" in {iterations} iterations with tolerance of {tolerance}')

# Zero using scipy
zero_scipy = scipy.optimize.fsolve(dydx, 7.5)
print(f'Zero found using scipy: {zero_scipy[0]}\"')

#The Newton-Raphson result matches the result using scipy
```