```
############################################

# Name -> Aditya Sengupta
# Enrol No. -> A4455717017
#Program - B.Sc. (Hons.) Physics 2017-2020

############################################

#import the necessary libraries
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import spline
from Code2pdf.code2pdf import Code2pdf

#Define the function prefixes using NumPy
sin = np.sin
cos = np.cos
pi = np.pi
exp = np.exp
ln = np.log
log = np.log10

#Define the derivative function
def df(x):
    h = 0.000000001
    top = f(x + h) - f(x)
    bottom = h
    slope = top / bottom
    return slope

fx = input("Enter the function but not a logarithmic function :")      #take function as in
put from the user
f = lambda x: eval(fx)                          # usinf lamda for defining function

#Code to determine the points between which the root lies

for x in range(0, 100):
    if f(x)*f(x+1) < 0:
        print("The root lies between",int(x),"and",int(x+1))

# Newton Raphson Method

#Enter the initial approximation and Error as Input from User
p = float((input("Enter the initial approximation :")))
n = float((input("Input the number of iterations :")))
e = int((input("Input the decimal places upto which you want the answer :")))

#Define the Newton raphson function
def nr(x):
    return (x - (f(x) / df(x)))

# p - the initial point i.e. a value closer to the root
# n - number of iterations
# e - precision of the final answer

def iterate(p, n):
    x = 0
    for i in range(int(n)):
        if i == 0:                              #calculate first approximation
            x = nr(p)
        else:
            x = nr(iterate(x, n))               # Iterate the subsequent approximations
        n = n-1
    return x

a = iterate(p, n)
print(round(a, e))                              #Final Solution
```

```
#Code to plot the curve
x = np.linspace(-10,10,10)
xnew = np.linspace(x.min(), x.max(), 300)       #Interpolates extra values to make smooth c
urve
f_smooth = spline(x, f(x), xnew)
df_smooth = spline(x, df(x), xnew)

#Plot the curves
fig=plt.figure()
plt.plot(xnew, f_smooth, 'b', label = 'function')
plt.plot(xnew, df_smooth, 'g', label = 'derivative')
plt.xlabel('x')
plt.ylabel('f and df')
fig.savefig('Newton__Raphson_Exp_2.png')
fig.suptitle('Function and Derivative')
plt.legend()

#Show the curve
plt.show()
```