

Στην αρχή έχω τα απαραίτητα imports για numpy, scipy και matplotlib:

```
from numpy import *
import numpy

from scipy import *
import scipy

import matplotlib.pyplot as plt
```

Η συνάρτηση και οι 2 πρώτες παράγωγοι ορίζονται και γίνεται το plot της $f(x)$:

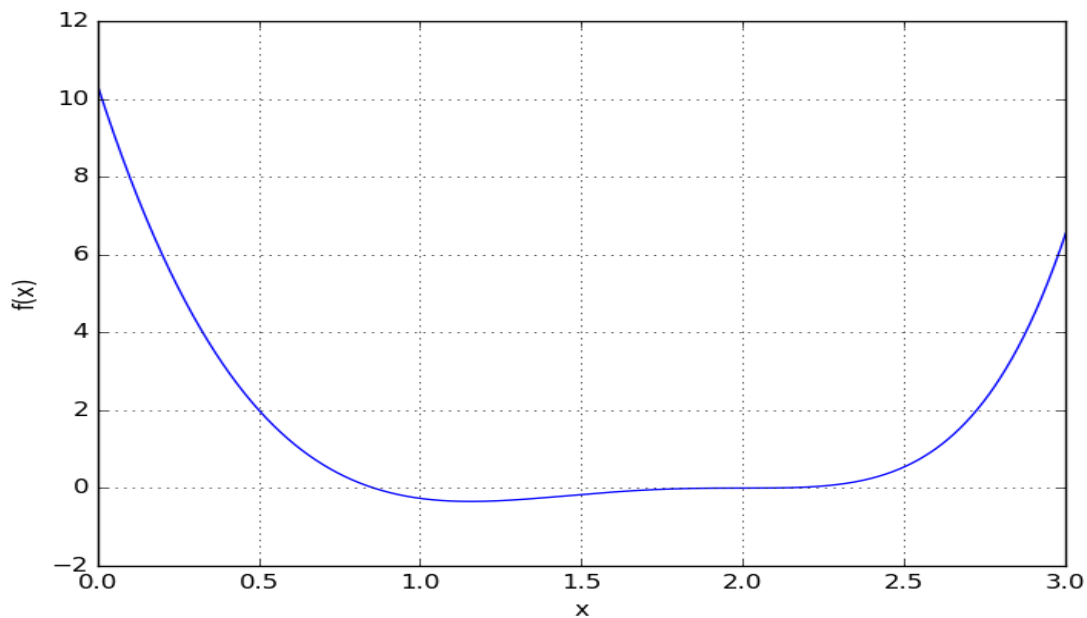
```
t = arange(0,3.001,step=.001)

def f(t):
    return (14*t)*(e**(t-2)) - 12*(e**(t-2)) - 7*(t**3) + 20*(t**2) - 26*t + 12

def f_der(t):
    return (14*t)*(e**(t-2)) + 2*(e**(t-2)) - 21*(t**2) + 40*t - 26

def f_der_2(t):
    return (14*t)*(e**(t-2)) - 16*(e**(t-2)) - 42*t + 40

plt.xlabel("x")
plt.ylabel("f(x)")
plt.plot(t, f(t))
plt.grid(True)
plt.show()
```



Καθώς η άσκηση δεν το απαιτεί δεν υπάρχουν έλεγχοι για σφάλματα ούτε συνθήκες τερματισμού σε περίπτωση που δεν υπάρχουν ρίζες.

Ακολουθεί η συνάρτηση για διχοτόμηση:

```
def bisect(a, b):
    a_1, b_1 = a, b
    root = (a + b) / 2

    # number of times to repeat to achieve 6 points accuracy
    N = int(ceil((log(b - a) - log(0.0000005)) / log(2)))

    for i in range(0, N):
        if (f(a) < 0 and f(root) > 0) or (f(a) > 0 and f(root) < 0):
            b = root
        elif (f(b) < 0 and f(root) > 0) or (f(b) > 0 and f(root) < 0):
            a = root
        root = (a + b) / 2

    return root, N, a_1, b_1
```

Απλά γίνετε η εφαρμογή του θεωρήματος για $N = \frac{\ln b - a - \ln error}{\ln 2}$ φορές
Στην συνέχεια έχω τη συνάρτηση για την μέθοδο Newton-Raphson:

```
def new_raph(start):
    temp_l = [start, start - (f(start)/f_der(start))]

    N = 1
    while 0.0000005*abs(temp_l[N]) < abs(temp_l[N-1] - temp_l[N]):
        temp = temp_l[N] - (f(temp_l[N])/f_der(temp_l[N]))
        temp_l.append(temp)
        N = N + 1

    root = temp_l[N]

    #N-1 because the N=N+1 happens at the very end of the while loop
    return root, N - 1, start
```

Εδώ αρχικοποιώ τον πίνακα temp_l με την τιμή εισόδου της συνάρτησης και το αποτέλεσμα μιας πρώτης εφαρμογής της αναδρομικής συνάρτησης

$$f(x_n) = f(x_{n-1}) - \frac{f(x_{n-1})}{f'(x_{n-1})}$$

Στην συνέχεια με τον έλεγχο στο while η συνάρτηση θα τρέχει μέχρι να επιτευχθεί η επιθυμητή ακρίβεια (6 δεκαδικά ψηφία)

Επιστρέφω την ρίζα, τον αριθμό επαναλήψεων και το σημείο εκκίνησης. Ο αριθμός επαναλήψεων είναι N-1 γιατί η αύξηση του δείκτη γίνεται στο τέλος του while ενώ ξεκινάει με τον δείκτη στο 1 αντι του 0 για να μπορεί να γίνει ο έλεγχος

Ακολουθεί η συνάρτηση της μεθόδου της τέμνουσας:

```
def intersection(a, b):  
    temp_l = [a, b, b - ((f(b)*(b - a))/f(b) - f(a))]  
  
    N = 1  
    while 0.0000005*abs(temp_l[N]) < abs(temp_l[N-1] - temp_l[N]):  
        temp = temp_l[N] - (f(temp_l[N])*  
                             (temp_l[N] - temp_l[N-1]))/(f(temp_l[N]) - f(temp_l[N-1]))  
        temp_l.append(temp)  
        N = N + 1  
  
    root = temp_l[-1]  
  
    #N-1 for the same reason as the N-R method  
    return root, N - 1, a, b
```

```
$ python ex1.py  
$ f(0.952374) = 0.000000
```