TP 2 - éléments de correction

```python
import math
import matplotlib.pyplot as plt
import numpy as np

def f(x):
    # Attention, si on veut pouvoir rentrer des "vecteurs", il faut travailler avec la bibliothèque numpy pour sin
    return x/2 - np.sin(x) + np.pi/6 - np.sqrt(3)/2

def df(x):
    return 0.5 - np.cos(x)

xpt = np.linspace(-5,5,10000)
ypt = f(xpt)
plt.plot(xpt,ypt,label='f')
plt.grid()
plt.legend()
plt.show()

# Méthodes ____________________________________

def newton(f,df,x0,tol,prec,n_max):
    x_n = x0
    x_np1 = x_n - f(x_n)/df(x_n)
    n = 0
    while n < n_max and abs(x_np1 - x_n) > prec and abs(f(x_n))>tol:
        n = n + 1
        x_n = x_np1
        x_np1 =  x_np1 - f(x_np1)/df(x_np1)
    return [n,x_n]

def dicho(f,a,b,tol,prec,n_max):
    a_n = a
    b_n = b
    n= 0
    while n < n_max and abs(b_n - a_n) >= prec and abs(f((a_n+b_n)/2)) >= tol :
        c_n = (a_n + b_n)/ 2
        if f(c_n) * f(a_n) > 0 :
            a_n = c_n
        else :
            b_n = c_n
        n = n + 1
    return [n,a_n,b_n]

# Exercice 2

# racine négative :
print("Racine négative\n")
x0 = -1
tol = 10**(-10)
prec = 10**(-10)
nmax = 100
[n,x_n] = newton(f,df,x0,tol,prec,nmax)
print(f" l'algorithme de Newton s'arrête après {n} étapes et nous donne comme solution approchée alpha = {x_n:.10f}.\n")

# dichotomie
a = -1.1
b = -1
[n,a_n,b_n] = dicho(f,a,b,tol,prec,nmax)
print(f"Pour dichotomie:\nnb_itérations = {n}\n[a_n,b_n]=[{a_n},{b_n}]\n")

# racine positive :
print("\nRacine positive\n")
x0 = 2
[n,x_n] = newton(f,df,x0,tol,prec,nmax)
print(f" l'algorithme de Newton s'arrête après {n} étapes et nous donne comme solution approchée alpha = {x_n:.10f}.\n")

# dichotomie
a = 2
b = 3
[n,a_n,b_n] = dicho(f,a,b,tol,prec,nmax)
```

```python
print(f"Pour dichotomie:\nnb_itérations = {n}\n[a_n,b_n]=[{a_n},{b_n}]\n")

## Exercice 3
def f3(x):
    return math.cos(2*x)**2 - x**2

def df3(x):
    return -4 * math.cos(2*x) * math.sin(2*x) - 2 * x

f = f3
df = df3

N = 60
l = 0.5149332646611294

abscisse_vecteur = np.arange(0,N+1)

# méthode courte
erreur_dicho_vecteur = np.array([abs(dicho(f,0,1.5,-1,-1,n)[1] - l) for n in range(N+1)])
erreur_newton_vecteur = np.array([abs(newton(f,df,0.75,-1,-1,n)[1] - l) for n in range(N+1)])

plt.semilogy(abscisse_vecteur, erreur_dicho_vecteur,label='dicho')
plt.semilogy(abscisse_vecteur, erreur_newton_vecteur,label='newton')

# méthode boucle
erreur_dicho_liste = []
erreur_newton_liste = []

for n in range(N+1):
    erreur_dicho_n = abs(dicho(f,0,1.5,-1,-1,n)[1] - l)
    erreur_dicho_liste.append(erreur_dicho_n)

    erreur_newton_n = abs(newton(f,df,0.75,-1,-1,n)[1] - l)
    erreur_newton_liste.append(erreur_newton_n)

erreur_dicho = np.array(erreur_dicho_liste)
erreur_newton = np.array(erreur_newton_liste)

plt.legend()
plt.show()
```