Codes du TP 7

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

#Exercice 1.1

#Question 1
def f1(x):
    return x**2-2


x1=np.arange(1,4,0.01)
plt.plot(x1,f1(x1))

#Question 4
def dicho(n):
    a=1
    b=4
    for k in range(n):
        m=(a+b)/2
        if f1(a)*f1(m)<=0:
            b=m
        else:
            a=m
    return a,b

#Question 6
def dichobis(eps):
    n=0
    a=1
    b=4
    while np.abs(a-b)>eps:
        n=n+1
        m=(a+b)/2
        if f1(a)*f1(m)<=0:
            b=m
        else:
            a=m
    return a,b,n
```

```
#Exercice 1.2

#Question 1
def g(x):
    return (-x+2)*np.exp(x)-2

x2=np.arange(0,3,0.01)
plt.plot(x2,g(x2))

#Question 6
def dichoter(eps,f):
    n=0
    a=1
    b=4
    while np.abs(a-b)>eps:
        n=n+1
        m=(a+b)/2
        if f(a)*f(m)<=0:
            b=m
        else:
            a=m
    return a,b,n

#Exercice 2.2

#Question 2
def newton1(n):
    u=3
    for k in range(n):
        u=(1/2)*(u+2/u)
    return u

#Question 4
def newton2(eps):
    n=0
    u0=3
    u1=(1/2)*(u0+2/u0)
    while np.abs(u0-u1)>eps:
        n=n+1
        u0=u1
        u1=(1/2)*(u1+2/u1)
    return u0,u1,n
```

#Exercice 2.3

#Question 1, fonction à utiliser ci-dessous, il faut ensuite adapter l'algorithme de la méthode de Newton à cette fonction particulière

```
def f2(x):
    return x**2-3
```

#Question 2, fonction à utiliser ci-dessous, il faut ensuite adapter l'algorithme de la méthode de Newton à cette fonction particulière

```
def f3(x):
    return x**3-2
```