TP n° 10 : Dichotomie et méthode de Newton

1.
```python
from math import *
def g(x):
    return sqrt(sin(x)) + x - 1
```

2.
```
>>> g(0)
-1.0
>>> g(pi)
2.1415926646561694
```

1 Dichotomie

3.
```python
def Dichotomie(f, a, b, eps):
    while abs(b-a)>eps:
        milieu = (a+b)/2
        if f(a)*f(milieu)<0:
            b = milieu
        else:
            a = milieu
    return (a+b)/2
```

4.
```
>>> Dichotomie(g, 0, pi, 10**(-3))
0.3861796633502102
```

5.
```python
def Dichotomie(f, a, b, eps):
    nb_ite = 0
    while abs(b-a)>eps:
        milieu = (a+b)/2
        nb_ite = nb_ite + 1
        if f(a)*f(milieu)<0:
            b = milieu
        else:
            a = milieu
    return (a+b)/2, nb_ite
```

6.
```
>>> Dichotomie(g, 0, pi, 10**(-3))
(0.3861796633502102, 12)
>>> Dichotomie(g, 0, pi, 10**(-9))
(0.3862368703959704, 32)
>>> Dichotomie(g, 0, pi, 10**(-15))
(0.3862368706447995, 52)
```

7. On commence par définir une fonction correspondant à ce problème.
```python
def h(x):
    return cos(x) - x**3
```
puis on utilise la fonction précédente :
```
>>> Dichotomie(h, 0, 1, 10**(-15))
(0.8654740331016142, 50)
```

2 Méthode de Newton

8. L'équation de la tangente à la courbe représentative de f au point d'abscisse x_n est
$$y = f'(x_n)(x - x_n) + f(x_n).$$
9. Comme x_{n+1} est l'abscisse du point d'intersection de cette tangente avec l'axe des abscisses, on a
$$0 = f'(x_n)(x_{n+1} - x_n) + f(x_n)$$
d'où
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$
10.
```python
def Newton(f,fp,x0,eps):
    x1 = x0 - f(x0)/fp(x0)
    while abs(x1-x0)>eps:
        x0 = x1
        x1 = x0 - f(x0)/fp(x0)
    return x1
```
11. On calcule la dérivée avec les formules mathématiques usuelles $g'(x) = \dfrac{\cos(x)}{2\sqrt{\sin(x)}} + 1$, d'où
```python
def gp(x):
    return cos(x)/(2*sqrt(sin(x))) + 1
```
12.
```
>>> Newton(g,gp,1,10**(-3))
0.386236870636691
```
13.
```python
def Newton(f,fp,x0,eps):
    x1 = x0 - f(x0)/fp(x0)
    nb_ite = 0
    while abs(x1-x0)>eps:
        x0 = x1
        x1 = x0 - f(x0)/fp(x0)
        nb_ite += 1
    return x1, nb_ite
```
14.
```
>>> Newton(g,gp,1,10**(-3))
(0.386236870636691, 3)
>>> Newton(g,gp,1,10**(-9))
(0.3862368706447994, 4)
>>> Newton(g,gp,1,10**(-15))
(0.3862368706447994, 5)
```
15.
```python
def derivee(f,x):
    h = 10**(-3)
    return (f(x+h)-f(x))/h
```
16.
```python
def Newton2(f,x0,eps):
    x1 = x0 - f(x0)/derivee(f,x0)
    nb_ite = 0
    while abs(x1-x0)>eps and nb_ite<1000:
        x0=x1
        x1 = x0 - f(x0)/derivee(f,x0)
        nb_ite += 1
    return x1, nb_ite
```