

```
# -*- coding: utf-8 -*-  
"""
```

```
Created on Sat Apr 12 14:56:01 2014
```

```
@author: dconduche  
"""
```

```
"""
```

```
DÉRIVÉE :
```

Le calcul de la dérivée vient des formules de Taylor :

$$f(x+h) = f(x) + h f'(x) + \frac{h^2}{2} f''(x) + o(h^2)$$

$$f(x-h) = f(x) - h f'(x) + \frac{h^2}{2} f''(x) + o(h^2)$$

donc on gagne un ordre dans le DL par rapport à $f(x+h) - f(x)$:

$$\frac{f(x+h) - f(x-h)}{2h} = f'(x) + o(h)$$

```
NEWTON:
```

* Newton est d'ordre 2 (on double le nombre de décimales à chaque étape), mais la convergence est délicate (conditions sur f' et f'').

Rmq : On calcule les termes d'une suite récurrente, mais sans tous les retenir Mathématiquement, $u = u_n$, et $v = u_{n+1}$.

On doit retenir u_n pour mesurer l'écart entre u_n et u_{n+1} .

```
"""
```

```
def derive(f, x):
```

```
    """Calcul approché de la dérivée de f en x"""
```

```
    h = 10**(-5)
```

```
    return (f(x+h) - f(x-h)) / (2*h)
```

```
def newton(f, fprime, u0, eps):
```

```
    """Recherche la solution  $x_0$  de  $f(x)=0$  par Newton
```

```
    Keyword arguments:
```

```
    f -- fonction suffisamment régulière
```

```
    fprime -- sa dérivée
```

```
    u0 -- float représentant le point de départ de la méthode
```

```
    eps -- float représentant la précision cherchée
```

```
    retourne  $x_0$  à eps près, ou part en boucle infinie
```

```
    """
```

```
    u = u0
```

```
    v = u - f(u) / fprime(u)
```

```
    while abs(u-v) > eps:
```

```
        u = v
```

```
        v = u - f(u) / fprime(u)
```

```
    return v
```

```
f = lambda x: x**2-2
```

```
fprime = lambda x: derive(f, x)
```

```
solN = newton(f, fprime, 2, 10**(-10))
```

```
solP = 2**(1/2)
```

```
print(solN-solP)
```