Correction du DS 3

Informatique pour tous, première année

Julien Reichert

Exercice 1

Attention à ne pas indexer par un couple, on a affaire à des listes de listes. Il faut absolument les deux parenthèses au niveau de la méthode `append`.

```
def positions(LL, x):
    reponse = []
    for i in range(len(LL)):
        for j in range(len(LL[i])):
            if LL[i][j] == x:
                reponse.append((i, j))
    return reponse
```

Exercice 2

Une variante de mes notes de cours, qui utilise `numpy` et a l'air un peu moins lourde...

```
import numpy as np

def euler(G, y0, X):
    Y = np.zeros(len(X))
    Y[0] = y0
    for i in range(1, len(X)):
        Y[i] = Y[i-1] + (X[i]-X[i-1])*G(Y[i-1], X[i-1])
    return Y
```

Exercice 3

```
def newton(f, df, x0, eps):
    xn = x0
    xnp1 = xn - f(xn) / df(xn)
    while abs(xnp1 - xn) > eps: # ou, au choix,  while abs(f(xnp1)) > eps
        xn = xnp1
        xnp1 = xn - f(xn) / df(xn)
    return xnp1
```

Exercice 4

Question 4.1 : Soit $f(x) = \alpha x^2 + \beta x + \gamma$. Alors $f'(x) = 2\alpha x + \beta$ et $f''(x) = 2\alpha$. L'évaluation de ces trois fonctions en x_0 nous donne un système triangulaire permettant de trouver $\alpha = \frac{c}{2}$, $\beta = b - cx_0$ et $\gamma = a - bx_0 + \frac{cx_0^2}{2}$. Pour ceux qui préfèrent l'analyse, cela va plus vite avec les formules de Taylor.

Question 4.2 : Le discriminant de la fonction polynomiale est alors $(b - cx_0)^2 - c(2a - 2bx_0 + cx_0^2)$, qui se simplifie (heureusement !) en $b^2 - 2ac$. Si celui-ci est strictement positif, alors les deux solutions à $f(x) = 0$ sont $\frac{b - cx_0 \pm \sqrt{b^2 - 2ac}}{c}$, s'il est nul on a seulement $\frac{b}{c} - x_0$ et s'il est négatif il n'y a pas de solution.

Question 4.3 : Mathématiquement, s'il y a deux solutions on est bien en peine de déterminer la plus proche, donc on laissera ce soin à Python...

```
def zero(a, b, c, x0):
    discriminant = b**2 - 2*a*c
    if discriminant == 0:
        return (b - c*x0)/2
    if discriminant > 0:
        r1 = (b - c*x0 - np.sqrt(discriminant))/c
        r2 = (b - c*x0 + np.sqrt(discriminant))/c
        if abs(r1 - x0) < abs(r2 - x0):
            return r1
        else:
            return r2
```

Si le discriminant est négatif, la fonction ne retourne donc rien.

Question 4.4 :

```
def newtonbis(f, df, ddf, x0, eps):
    xn = x0 - 2 * eps
# ainsi on entre forcément dans la boucle
# et on ne fait pas deux fois le long travail
    xnp1 = x0
    while abs(xnp1 - xn) > eps:
        xn = xnp1
        prochain_point = zero(f(xn), df(xn), ddf(xn), xn)
        if prochain_point == None:
            xnp1 = xn - f(xn) / df(xn)
        else:
            xnp1 = prochain_point
    return xnp1
```