TP N°11 : Les fonctions récursives

Définition

La *récursivité*, ce n'est pas autre chose que la récurrence mathématique transposée dans la programmation. Par exemple, le calcul d'un terme de la suite suite U définie par récurrence par

$$U_0 = 2, \forall n \in \mathbb{N}^*, U_n = \frac{1}{2}(U_{n-1} + \frac{3}{U_{n-1}})$$

se fera par une fonction récursive :

```
def u(n):
    if n == 0 :
        return 2
    else:
        res = (u(n-1) + 3/u(n-1)) / 2
        return res
for k in range (10) :
    print(u(k))
```

Remarques :

- La fonction u est récursive parce que dans sa définition même, elle est utilisée.
- On distinguera l'appel principal de u des appels récursifs :
  - L'*appel principal* se fait lorsque u n'est pas encore en cours d'exécution.
  - Les *appels récursifs* sont les appels provoqués par l'exécution de u.
- Une fonction récursive peut avoir un ou plusieurs *cas de bases* et un ou plusieurs *cas de propagation*

  Le cas de base C'est un cas particulier pour lequel la valeur à retourner ne nécessite pas d'appel à la fonction u. Sans sa présence, la fonction ne peut pas se terminer (garantie l'arrêt du programme).

  Le cas de propagation C'est lui qui contient l'appel récursif et dont l'un ou plusieurs des arguments qui lui sont transmises doivent converger et arriver à la condition d'arrêt ($n = 0$).

Exercice 1

```
def  f(a,b):
    if b == 1 :
        return a
    else:
        return a + f(a,b-1)
print(f(3,5))
```

1. Quel est le résultat affiché par ce programme ? Donner le schéma d'exécution de l'appel f(3,5).
2. Quel est le cas de base dans cette fonction récursive ?
3. Qu'est ce qui garantit dans les appels récursifs que le programme finira par s'arrêter ?
4. Que retourne `f(a,b)` (a et b étant des entiers naturels non nuls) ?

Remarques :

- A chaque nouvel appel récursif, des nouvelles variables sont crées qui portent les mêmes noms mais pas les mêmes valeurs
- L'ensemble des variables avec leurs valeurs de chaque appel récursif composent le *contexte* de l'appel récursif principal

Exercice 2

Définir les fonctions récursives suivantes :

1. `fact(n)` : la fonction factorielle, définie par $fact(0) = 1, \forall n \in \mathbb{N}^*, fact(n) = nfact(n-1)$
2. `somme(n)` : calcule la somme des n premiers entiers.
3. `syracuse(n)` : calcule et affiche les termes de la suite de Syracuse. Rappelons que selon la conjecture de Syracuse, toute itération de cette fonction à partir d'un entier naturel non-nul aboutit à la valeur 1.

$$C_n = \begin{cases} n/2 & \text{si n est pair} \\ 3n+1 & \text{sinon} \end{cases}$$

Exercice 3

1. Écrire une fonction récursive `somme_termes` qui prend en argument une liste. Elle renvoie une liste de même longueur dont la i-ème élément est la somme des i premiers éléments de la liste.
   Par exemple : `somme_termes([1,4,3,-1])` renvoie `[1,5,8,7]`
2. Écrire une fonction récursive `somme_rec` qui renvoie la somme d'une liste passée en argument.

Exercice 4

Écrire une fonction récursive permettant de calculer des termes de la suite U_n définie par :
$U_0 = U_1 = U_2 = 1$ et $\forall n \geq 3$, $U_{n+3} = U_n + U_{n+1} + U_{n+2}$