Python
Récursivité

1 Fonction récursive simple

Définir une fonction récursive en Python ne pose aucun problème : le nom de la fonction est défini dès le `def`, on peut donc l'utiliser dans sa propre définition. Par exemple factorielle définie par (avec $n \in \mathbb{N}$) :

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n \times (n-1)! & \text{sinon.} \end{cases}$$

s'écrit directement :

```
def fact_rec(n):
    if n == 0:
        return 1
    else:
        return n * fact_rec(n-1)
```

On utilise la fonction `trace` (dont le code est donné en annexe) pour voir les différents appels et les `return` exécutés.

```
>>> fact_rec = trace(fact_rec)
>>> fact_rec(3)
|-- fact_rec 3
|   |-- fact_rec 2
|   |   |-- fact_rec 1
|   |   |   |-- fact_rec 0
|   |   |   |   |-- return 1
|   |   |   |-- return 1
|   |   |-- return 2
|   |-- return 6
6
```

On voit bien ici les différents appels à la fonction qui sont d'abord faits, jusqu'au cas d'arrêt : de 3 jusqu'à 0. Une fois au cas d'arrêt, on peut commencer à "remonter" : chaque appel est remplacé par sa valeur, et donc l'appel du "dessus" peut se terminer et à son tour retourner sa valeur. Et on remonte comme cela jusqu'à l'appel initial qui peut enfin se terminer et retourner la valeur finale.

2 Éviter la récursion infinie

On rappelle les règles pour l'écriture d'une fonction récursive :
Le schéma d'une fonction récursive simple :

```
            if <condition arrêt>:
               {\it <instructions arrêt>}
            else:
               {\it <instructions comportant un appel récursif>}
```

Règle 1 : Un algorithme récursif comporte toujours un cas d'arrêt.

Règle 2 : L'appel récursif doit toujours être fait sur des données différentes évoluant vers le cas d'arrêt.

Il faut parfois vérifier les paramètres pour éviter une récursion infinie, comme le paramètre de la fonction précédente qui doit être un entier naturel. Cela ne doit pas être fait dans la fonction récursive (pas très optimal). On peut soit inclure notre fonction dans une fonction "chapeau", soit écrire une fonction d'appel :

```
def fact(n):
    assert n >= 0   # erreur sinon
    return fact_rec(n)
```

3 Python et la récursivité

Python n'est pas un langage qui gère bien la récursivité, les raisons seront vues dans un cours plus avancés.

Une singularité : le nombre d'appels récursifs est limité par une valeur (en général 1000) que l'on peut modifier (déconseillé...). Voir les fonctions `sys.getrecursionlimit` et `sys.setrecursionlimit`.