Chapitre 2
Récursivité

I. Définition et premier exemple

Définition 2.1.
La récursivité est une démarche qui fait référence à l'objet même de la démarche à un moment du processus. En d'autres termes, c'est une démarche dont la description mène à la répétition d'une même règle. Par exemple :
- écrire un algorithme qui s'invoque lui-même
- définir une structure à partir de l'une au moins de ses sous-structures.

▪ **Exemple 2.1.**
On définit factorielle n, $n!$ par : $n! = n \times (n-1) \times \ldots 2 \times 1$. On peut voir la factorielle de la façon suivante :

$$\begin{aligned} 5! &= 5 \times 4! \\ &= 5 \times 4 \times 3! \\ &= 5 \times 4 \times 3 \times 2! \\ &= 5 \times 4 \times 3 \times 2 \times 1 \qquad\qquad = 120 \end{aligned}$$

▪

```python
def fact(n):
        if n <= 1:
                return 1
        else:
                return n * fact(n-1)

fact(5) = 120
```

II. Déroulé d'un programme récursif

Chaque appel récursif s'ajoute à la pile des appels successifs de la fonction. Chaque appel possède son propre environnement, donc ses propres variables. La pile est nécessaire pour mémoriser les valeurs propre à chaque appel.

R **Attention!!!** En python, la taille de la pile des appels récursifs est limitée. Si la récursivité est trop profonde et que l'on atteint cette limite, on déclenche une *RecursionError*

III. Eléments Caractéristiques

1. Cas d'arrêt

Il faut que dans la procédure, il y ait au moins une situation qui ne consiste pas en un appel récursif (Sinon la procédure ne s'arrêterait jamais). Dans la fonction Factorielle le cas d'arrêt est :

```
1 if n <= 1:
2         return 1
```

Cette situation est appelé **situation de terminaison**,**situation d'arrêt**,**cas d'arrêt** ou **cas de base**.

2. Il faut se rapprocher du cas d'arrêt

Là où la récursivité et la récurrence sont vraiment **DIFFÉRENTES** est le cas suivant. Une récurrence (classique) part d'un cas d'initialisation et va s'en éloigner pour aller vers $+\infty$. Visuellement on déplace le même problème avec juste des nombres plus gros. Une récursivité a plus un effet de loupe, à chaque appel on rapproche la loupe du cas d'arrêt.
Plus simplement, l'appel récursif doit modifier l'argument **vers** le cas d'arrêt.

```
1 return n*fact(n-1)
```

Il faut s'assurer que la situation de terminaison est atteinte après un nombre fini d'appels récursifs.

La preuve de terminaison d'un algorithme récursif se fait en identifiant la construction d'une suite strictement décroissante d'entiers positifs ou nuls. Dans le cas de la factorielle, il s'agit simplement de la suite des valeurs du paramètre.