

LA RÉCURSIVITÉ

On parle de récursivité lorsqu'une fonction **s'appelle elle-même**. C'est une *technique* de programmation qui s'appuie sur les *pires d'appel de fonctions*:

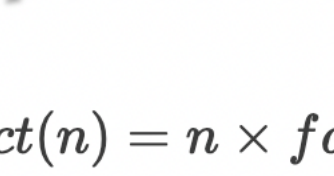
- Lorsqu'une fonction est appelée, elle est placée au sommet de la pile d'exécution.
- Lorsque l'exécution d'une fonction est terminée, elle est dépilée.
- Une fonction qui n'est pas au sommet de la pile a interrompu son exécution, elle reprendra lorsqu'elle se trouvera au sommet de la pile.

Il est nécessaire que les appels récursifs se terminent. Il est donc toujours nécessaire d'avoir un **cas de base**.

La force de cette technique de programmation est qu'elle permet de faire des retours en arrière lors de l'exécution d'un algorithme, nous le verrons par exemple lors du parcours des arbres binaires. Notons qu'un algorithme **récursif** peut toujours s'écrire également de manière **itérative**.

Dans les cas les plus simples, un algorithme récursif s'écrit à partir de ses primitives contenant au moins un **cas de base** et une relation de **réurrence** :

Exemple du calcul d'une factorielle :

$$\begin{cases} fact(0) = 1 \\ \forall n \in \mathbb{N} \quad fact(n) = n \times fact(n - 1) \end{cases}$$


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