05 - Récursivité

1) Principe & Définitions

> Dire d'une fonction qu'elle est récursive signifie qu'elle s'appelle elle-même. Ce principe en informatique est basée sur la notion mathématique de récurrence.

Exemple : Calcul de la factorielle

$3! = 1 * 2 * 3 = 6$

```
3 != 6 # Réponse True
```

Par convention :

$0! = 1$

On peut calculer les factorielles de deux façons différentes :

- Méthode itérative :

```
def fact_iter(n):
        reponse = 1
            for i in range(1, n+1):
                reponse *= i
            return reponse
```

- La méthode récursive :

```
def fact_rec(n):
        if n <= 1:
            return 1
```

```
        else:
                return n*fact_rec(n-1)
```

A retenir : Une fonction récursive est toujours constituée de deux parties :

- Cas de base
- Pas de récursion

▼ 2) Terminaison d'une fonction récursive

> Pour garantir la terminaison d'une fonction récursive il faut s'assurer que l'on "retombe" en un nombre fini d'étapes sur un des cas de bases

En python un garde fou est prévu, on ne peut pas dépasser 999 étapes récursives `RecursionError: Maximum recursion depth excedeed`

Pourquoi on fixe une limite : La RAM, on stock l'information dans la RAM et donc pour éviter le crash ou le freeze a cause du remplissage de la RAM.

> Récurrence non terminale fait de proche en proche et attends pas de tout calculer avant de propager l'information

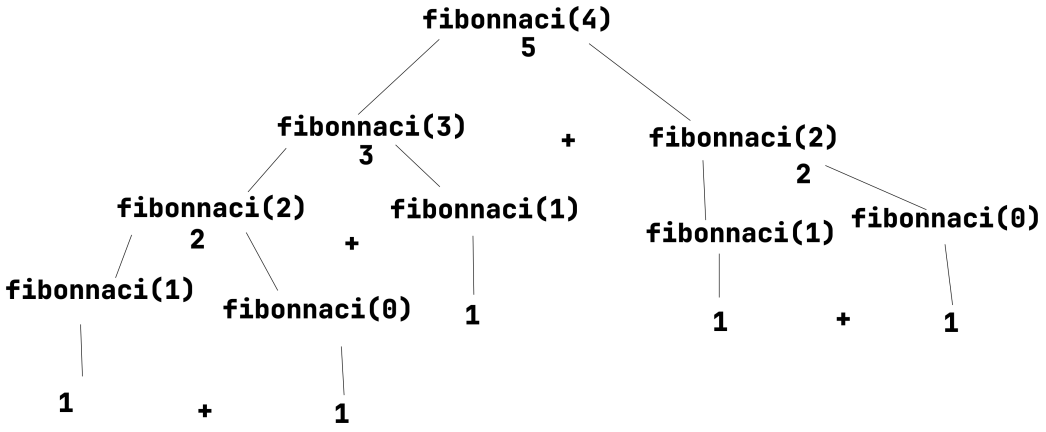
▼ 3) Représentation d'une exécution récursive

> On peut représenter des appels récursifs à l'aide d'un arbre.

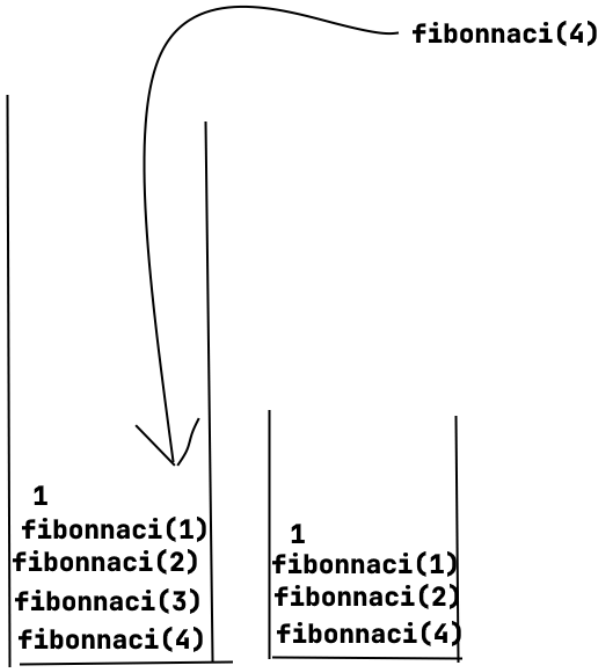
Exemple : Exécution de `fibonnaci(4)`

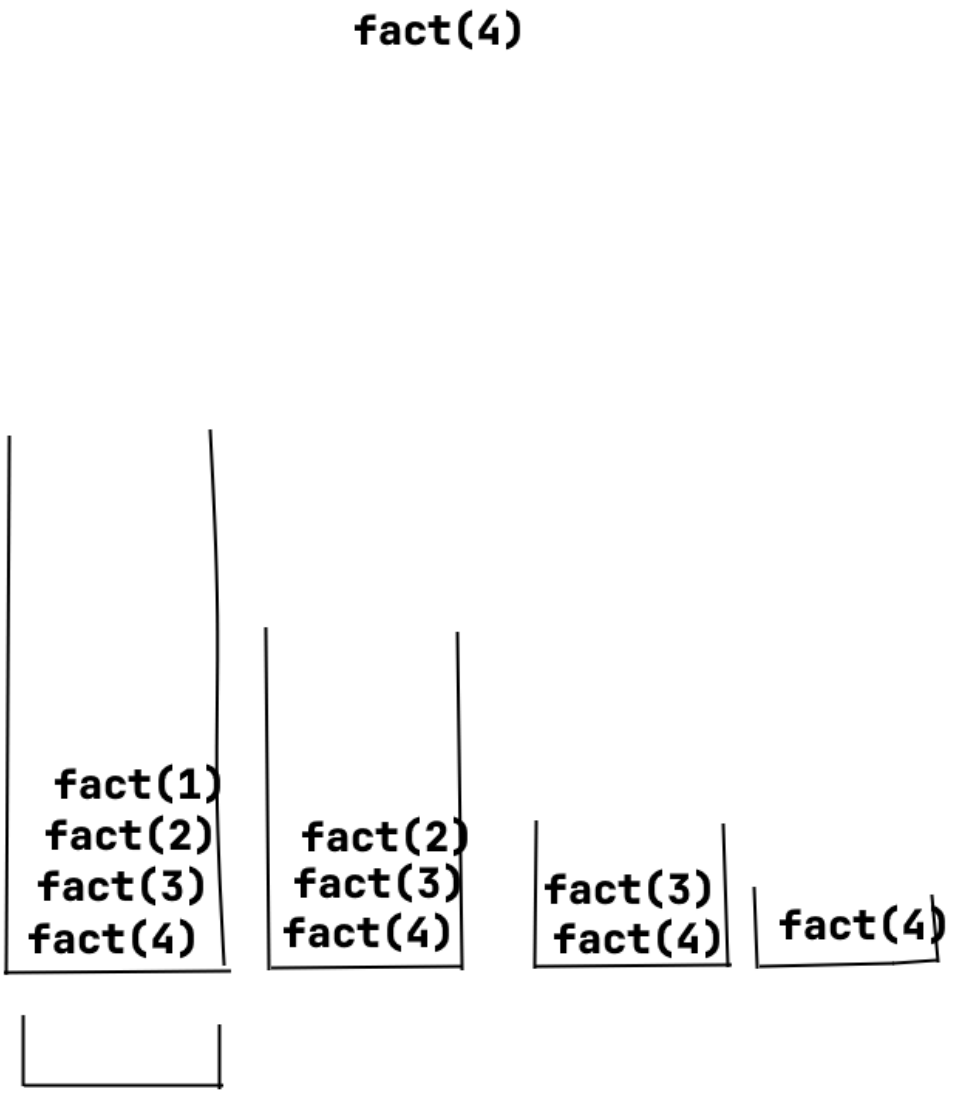
```
def fibonnaci(n):
        #cas de base
```

```
        if n <= 1:
            return 1
        else:
            return fibonnaci(n-1) + fibonnaci(n-2)
```



Pile d’exécution d’appels (Call Stack) : A chaque appel de la fonction le résultat est empilé, il est dépilé lorsque l’on atteint le(s) cas de base.


fibonnaci(4)
1
fibonnaci(1)
fibonnaci(2)
fibonnaci(3)
fibonnaci(4)
1
fibonnaci(1)
fibonnaci(2)
fibonnaci(4)



Exercice 1 :

Exercice une fonction récursive qui renvoie le maximum d’un tableau d’entiers.

```
from icecream import ic #permet le debug de ouf
def rechercheMaxRec(tab):
    #cas de base
    if len(tab) == 1:
        return tab[0]
    #recursion step
```

```
    else:
                ic(tab[1:])
        #le slicing n'est pas au programme mais c'est plus
        return max(tab[0], rechercheMaxRec(tab[1:]))
```