Récursivité

1 Usage récursif d'une fonction

1.1 Rappel : usage d'une fonction

En programmation, il est nécessaire de distinguer deux manipulations de fonction :

1. La définition de la fonction :

2. L'appel de la fonction :

1.2 Definition d'une fonction récursive

Une fonction récursive peut s'appeler elle-même. Ce qui entraine une succession d'appel de la même fonction.
Intérêt : exprimer simplement le traitement nécessaire (applicable à certains types de problèmes)
La résolution de problème par récursivité peut aussi être menée par récurrence.

1.3 Exemple : calcul de factorielle $n!$

Notons f la fonction factorielle de n : $f(n) = n!$. On peut exprimer $f(n)$ de deux manières :

$$\begin{aligned} f(n) &= 1 \times 2 \times \cdots \times n \\ f(n) &= n \times f(n-1) \end{aligned}$$

```python
def facto1(n):
    y = 1
    for k in range(1,n+1):
        y = k*y
    return(y)
```

```python
def facto2(n):
    if n <= 1:
        return(1)
    else:
        return(n*facto2(n-1))
```

2 Règle d'usage

Une fonction récursive présente un comportement alternatif :

- soit la fonction s'appelle elle-même, mais avec des arguments plus faciles à traiter (pour simplifier le problème) ;
- soit la fonction ne s'appelle pas elle-même, car les arguements sont suffisamment simples pour être traitées directement (on parle du cas trivial).

La condition d'arrêt de récursivité est la partie de la fonction récursive qui fait qu'elle ne va plus s'appeler elle-même. Cette partie est indispensable, sinon le programme s'exécute indéfiniment.

2.1 Synthèse

2.2 Code

```
def fonction_recursive(x):
    if cas trivial :
            return constant_simple
    else :
            return(fonction_recursive(x-1))
```

3 Applications

3.1 Application 1 : programme mystère

On donne le programme suivant :

```
def mystere(C):
    if C == '':
        return(C)
    else:
        return(mystere(C[1:]) + C[0])

print(mystere('python'))
```

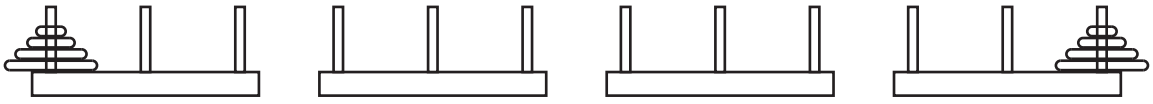
Quel est le résultat de ce programme ?
Proposer une version récurrente.

3.2 Application 2 : compléter une liste

Une liste `A` est en mémoire, elle a une longueur inférieure ou égale à 100. Ecrire un programme récursif qui complète la liste A par des 0, jusqu'à ce que la liste est une longueur d'au moins 100.

3.3 Application 3 : tours de Hanoï

Le problème des tours de Hanoï est un jeu dont l'objectif est de déplacer une pile de disque d'un piquet de départ vers un piquet d'arrivée. Il n'est possible de déplacer qu'un disque à la fois, et on ne peut poser un disque que sur un disque de diamètre supérieur.



Mise en place informatique : chacun des disques est représenté par un entier (n entiers croissants), la valeur de l'entier correspondant au diamètre. Trois listes A , B, C représentent les piquets accueillant les disques. A l'initialisation :

```
A=[6,5,4,3,2,1]
B=[]
C=[]
```

1. Expliciter la solution du problème sous forme récursive ;
2. Proposer le programme recursif correspondant.

4 Limites de la programmation récursive

Lors d'un appel récursif de fonction, un mécanisme de pile est mis en place (avec des variables locales et des transfert de valeur au niveau supérieur). La mémoire d'un ordinateur n'étant pas infinie, il n'est pas possible de faire une infinité d'appel récursif.

Un programme peut donc atteindre la limite en mémoire de la pile de récursivité. En python, cette limite est d'à peu près 1000 appel récursif. Cette limite est réglable si besoin.