CHAPITRE 5 : RECURSIVITE

Introduction

On considère la suite numérique (u_n) définie par $\begin{cases} u_0 = 1 \\ u_{n+1} = u_n^2 + 2 \; pour \; n \geq 1 \end{cases}$.

1. Calculer u_1, u_2, u_5.

2. Comment doit-on faire pour calculer u_{10} ?

3. Proposer plusieurs programmes permettant de calculer un terme u_n de cette suite.

I. DEFINITION

Une fonction récursive est une fonction qui s'appelle elle-même.

Exemple :

On souhaite écrire de manière récursive le calcul de la puissance d'un nombre a.

On sait que $a^0 = 1$, et, pour tout entier n, $a^n = a \times a^{n-1}$.

On obtient alors l'implémentation récursive suivante :

```
def puissance(a, n):
    if n == 0:
        return 1
    else:
        return a * puissance(a , n-1)
```

La fonction commence par une condition d'arrêt, qui traite le cas de base (valeur pour laquelle la fonction s'arrête immédiatement).

II. PILE D'EXECUTION

L'appel d'une fonction récursive utilise une structure de pile.

Dans l'exemple précédent, l'appel de puissance(7 , 4) va engendrer une suite d'appels « en cascade » que l'on peut représenter par l'arbre suivant :

```
puissance(7,4) =
 return 7 * puissance(7,3)
                   |
        return 7 * puissance(7,2)
                          |
               return 7 * puissance(7,1)
                                 |
                      return 7 * puissance(7,0)
                                        |
                                     return 1
```

On peut représenter cela de manière schématisée :

On empile :

$puissance(7,4)=7*puissance(7,3)$

$puissance(7,3)=7*puissance(7,2)$
$puissance(7,4)=7*puissance(7,3)$

$puissance(7,2)=7*puissance(7,1)$
$puissance(7,3)=7*puissance(7,2)$
$puissance(7,4)=7*puissance(7,3)$

$puissance(7,1)=7*puissance(7,0)$
$puissance(7,2)=7*puissance(7,1)$
$puissance(7,3)=7*puissance(7,2)$
$puissance(7,4)=7*puissance(7,3)$

$puissance(7,0)=7$
$puissance(7,1)=7*puissance(7,0)$
$puissance(7,2)=7*puissance(7,1)$
$puissance(7,3)=7*puissance(7,2)$
$puissance(7,4)=7*puissance(7,3)$

Puis on dépile :

$puissance(7,1)=7*7$
$puissance(7,2)=7*puissance(7,1)$
$puissance(7,3)=7*puissance(7,2)$
$puissance(7,4)=7*puissance(7,3)$

....

La pile utilisée est de taille limitée : 1000 en Python. Au-delà de cette limite, on a une erreur de dépassement de pile : ***maximum recursion depth exceeded***.

III. ECRITURE D'UNE FONCTION RECURSIVE

Pour écrire une fonction récursive, on doit :

- Déterminer le type de données à renvoyer
- Déterminer la condition d'arrêt (cas de base) : pour quelle valeur de l'argument le problème est-il résolu immédiatement, et écrire cette condition
- Déterminer de quelle manière la taille du problème est réduite : quel argument décroît, quelle liste a une taille qui diminue …
- Ecrire l'appel récursif en veillant à ce qu'on arrive bien à la condition d'arrêt après un certain nombre d'appels.

Exemple : On peut écrire la multiplication d'un nombre a par un entier n à l'aide d'une fonction récursive faisant appel à l'addition.

- *La donnée à renvoyer est de type flottant (ou entier si le nombre est entier)*
- *La condition d'arrêt est : pour $n=1$, renvoie a.*
- *A chaque appel récursif, la valeur de n décroit de 1*
- *L'appel récursif correspond à : $a\times n=a+a\times(n-1)$*

On obtient alors le script suivant :

```
def produit(a, n):
    if n == 1:
        return a
    else:
        return a + produit(a , n-1)
```

Application : Ecrire une fonction récursive permettant de calculer la factorielle d'un entier n.

On rappelle que $n!=1\times2\times3\times\ldots\times(n-1)\times n$

IV. COMPLEXITE

On note T_n le nombre d'opérations nécessaires pour évaluer une fonction récursive sur un problème de taille n. Pour évaluer T_n, on cherche une relation de récurrence sur T_n.

Par exemple, pour calculer de manière récursive la puissance d'un nombre, on a $T_n = T_{n-1} + 1$: A chaque appel récursif, on effectue une multiplication.

On sait alors que $T_n = n+1$ $(suite\ arithmétique)$. La complexité est donc ici $O(n)$.

Le tableau ci-dessous donne quelques complexités classiques :

Relation de récurrence	Complexité
$T_n = T_{n-1} + \Theta(1)$	$\Theta(n)$
$T_n = T_{n-1} + \Theta(n)$	$\Theta(n^2)$
$T_n = 2T_{n-1} + \Theta(1)$	$\Theta(2^n)$

Remarque : La récursivité a un coût élevé. Il est parfois possible de réduire ce coût en limitant le nombre d'appels récursifs.

Exemple : on peut définir autrement la fonction puissance(x,n) sous la forme suivante, qui utilise la parité de n :

$$puissance(x,n) = \begin{cases} 1 & \text{si } n = 0, \\ x & \text{si } n = 1, \\ carre(puissance(x, n/2)) & \text{si } n > 1 \text{ et } n \text{ est pair,} \\ x \times carre(puissance(x, (n-1)/2)) & \text{si } n > 1 \text{ et } n \text{ est impair.} \end{cases}$$

Quel est le nombre d'appels nécessaires avec cette définition pour déterminer $puissance(7,28)$ *?*

Exercices d'application : exercices 1 à 5