

Récursivité

J. Boucher

Lycée Paul Riquet, T^{le} NSI

24 septembre 2024

Plan

1 I. Les fonctions récursives

2 II. Différentes formes de récursivité

Définition

Fonction récursive

Une fonction f est dite **récursive** lorsque la définition de f fait elle-même appelle à f .

Exemple : Calcul de la somme des $n + 1$ premiers entiers

```
1 def somme(n):  
2     if n == 0:  
3         return 0  
4     else:  
5         return n + somme(n - 1)
```

Définition informatique

$$\text{pour tout } n \in \mathbb{N}, \text{ somme}(n) = \begin{cases} 0 & \text{si } n = 0, \\ n + \text{somme}(n - 1) & \text{si } n > 0. \end{cases}$$

Définition mathématique

Terminaison d'une fonction récursive

Tout comme une boucle non bornée peut se répéter indéfiniment, une fonction récursive peut donner lieu à un algorithme dont l'exécution se poursuit indéfiniment.

Terminaison d'une fonction récursive

Pour qu'une fonction récursive se termine, il faut s'assurer :

- qu'elle possède au moins un cas de base,
- qu'un de ses cas de base soit systématiquement atteint.

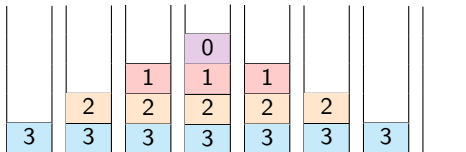
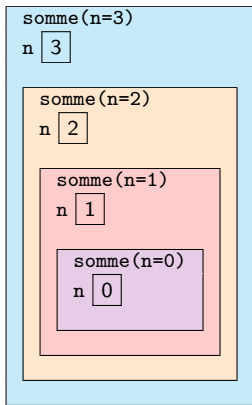
```
1 def somme(n):  
2     |   if n == 0:  
3     |       return 0  
4     |   else:  
5     |       return n + somme(n - 1)  
6 somme(-1)
```

En appelant somme avec -1 pour argument, le cas de base n'est jamais atteint.

→ **Spé. Maths** : Plus formellement, on peut généralement prouver la terminaison, la complexité et la correction par récurrence.

Pile d'appels

L'appel récursif de `somme(3)` va engendrer une suite d'appels « en cascade » à la fonction `somme`.



- Pour chacun de ces appels, un **contexte d'exécution** contenant une nouvelle variable locale `n` est créé en mémoire ; il est ajouté à une **pile d'appels** ou « empilé ».
- Quand une fonction termine son exécution, le contexte correspondant est retiré du sommet de la pile ou « dépilé ».

Différentes formes de récursivité

Récursion simple

Cas de base et récursif uniques

```
1 def puissance(x, n):  
2     if n == 0:  
3         return 1  
4     else:  
5         return x * puissance(x, n-1)
```

Cas de base multiples

```
1 def puissance(x, n):  
2     if n == 0:  
3         return 1  
4     if n == 1:  
5         return x  
6     else:  
7         return x * puissance(x, n-1)
```

Cas récursifs multiples

```
1 def puissance(x, n):  
2     if n == 0:  
3         return 1  
4     if n > 1 and n%2 == 0:  
5         return puissance(x, n//2)**2  
6     else:  
7         return x *  
           puissance(x, (n-1)//2)**2
```

Différentes formes de récursivité

Récursion imbriquée

Dans la définition d'une fonction on trouve plusieurs appels à elle-même de manière imbriqué.

```
1 def f91(n):  
2     if n > 100:  
3         return n - 10  
4     else:  
5         return f91(f91(n+11))
```

Récursion mutuelle

Quand des fonctions récursives font référence les unes aux autres, on parle alors de définitions récursives mutuelles.

```
1 def pair(n):  
2     if n == 0:  
3         return True  
4     else:  
5         return impair(n-1)  
6 def impair(n):  
7     if n == 0:  
8         return False  
9     else:  
10        return pair(n-1)
```