Chapitre 2: Fonctions et récursivité

brice.mayag@dauphine.fr

M1 SIREN: maj. SIEE/GSI

Dauphine | PSL
UNIVERSITÉ PARIS

Sommaire

Sommaire

Objectifs

- Découper un programme en plusieurs parties
- Rendre un programme plus lisible et efficace

Approche fonctionnelle

```
Fonction f(param_entree):
Instruction f1
Instruction f2
Retourne val_sortie

Programme principal:
Instruction 1
Instruction 2
Instruction 3
Si condition a vérifiée :
Appelle f(param_entree)
FinSi
Instruction 4
```

Syntaxe d'une fonction en Python

```
def fonction_nom(par_1,...,par_k):
    instruction_1
    ...
    intruction_m
    return res_1,...,res_j
```

- `fonction_nom` est l'identifiant ou le nom de la fonction, il suit les mêmes règles que les identifiants des variables ;
- `par_1 à par_k` sont les noms des paramètres en entrée, ou les noms des arguments de la fonction ;
- `res_1 à res_j` sont les résultats retournés par la fonction ;
- les instructions associées à une fonction doivent être indentées par rapport au mot-clé `def` ;
- s'il n'y a aucun résultat, l'instruction `return` est facultative, ou peut être utilisée seule sans être suivie par une valeur ou une variable.

Définition et appel d'une fonction

- Attention, définir et appeler une fonction sont des actions totalement différentes !
- Définir une fonction ne suffit pas pour qu'elle renvoie un résultat. Une fonction doit être d'abord définie, et ensuite appelée.
- Attention à l'indentation quand vous tapez un programme python.

Appel d'une fonction

L'exécution d'une fonction A se fait par un appel dans un sous-programme B.

- l'appel s'effectue par une instruction en utilisant le nom de la fonction A
- lors de l'appel, une valeur est affectée à chaque paramètre d'entrée de A à la fin de A , l'exécution reprend dans le sous-programme B
- la valeur retournée peut être utilisée dans B pour une affectation, un calcul ...

```
fonction_nom(val_1,...,val_k)

res_1,...,res_j = fonction_nom(val_1,...,val_k)
```

```
#Un exemple de fonction en Python
def minimum(a,b):
  if (a >= b):
     res = b
  else :
     res = a
  return res

m1 = minimum(2,3)
m2 = minimum(5,10)
print(minimum(m1,m2))

m1 = minimum(a=2,b=3)
m2 = minimum(a=5,b=10)
print(minimum(m1,m2))
```

Les variables a et b sont appelées des paramètres de la fonction minimum, alors que les valeurs $2, 3$ respectivement $5, 10$ sont appelées des arguments.

Variables locales et variables globales

Lorsqu'une fonction utilise une variable, elle est toujours "locale" à cette fonction.

```
def triple (n) :
resultat = n * 3
return resultat
>>> triple(4)
12
>>> resultat              #variable locale n'est pas connue
Traceback (most recent call last): ...
>>> resultat =128 #variable globale
>>> resultat(2)
Traceback (most recent call last): ....
>>> triple(2)
6
>>> resultat #variables globale inchang\'{e}e m\^{e}me si
#utilis\'{e}e entretemps comme variable locale
128
```

Sommaire

Algorithmes récursifs

Définition

Un algorithme est dit récursif s'il s'appelle lui même.

Récusivité

Principe qui consiste à décrire les étapes nécéssaires à la résolution de problèmes en utilisant la résolution du même problème sur des entrées plus petites.

Algorithmes récursifs : exemple

Factorielle d'un nombre

On peut définir la factorielle d'un nombre de la manière suivante :

$$n! = \begin{cases} 1 & \text{si } n \leq 1 \\ n(n-1)! & \text{sinon} \end{cases}$$

Algoritmes récursifs : exemple

Factorielle : algorithme itératif

Algorithm 1 FACTORIEL-ITERATIF (n : entier positif)

1: $x \leftarrow 1$
2: **for** $i \leftarrow 1$ to n **do**
3: $\quad x \leftarrow i * x$
4: **end for**
5: return x

Algoritmes récursifs : exemple

Factorielle :algorithme récursif

Algorithm 2 FACTORIEL-RECURSIF (n : entier positif)

1: **if** $n = 1$ **then**
2: return 1
3: **else**
4: return n FACTORIEL-RECURSIF($n - 1$)
5: **end if**