Fondements informatiques I

Cours 4: Fonctions

Sorina Ionica sorina.ionica@uvsq.fr

Sandrine Vial sandrine.vial@uvsq.fr

Les fonctions

- Il est très fréquent d'utiliser la même fonctionnalité plusieurs fois.
- La librairie standard de Python ou des librairies externes vont vous fournir des fonctions déjà codées.

Exemples

```
import math

x=2
math.sqrt(x)
math.cos(x)
print(x)
chaine="hello"
x=len(chaine)
```

Les fonctions

En mathématique une fonction s'écrit $f : x \to f(x)$.

Il y a trois parties dans la fonction : f son *nom*, x son *argument* et $f(x)$ son image (valeur/résultat).

Quelques exemples que vous connaissez bien :

$$\begin{aligned} f(x) &= ax + b \\ f(x, y) &= x + y \end{aligned}$$

$$\begin{aligned} f(x) &= x^2 \\ f(x) &= 2^x \\ f(x) &= |x| \end{aligned}$$

Définir une fonction en Python

Syntaxe

```
def nom_fonction(arg 1, arg2, ..., argn)
        instruction1
        instruction 2
        ...
        instruction n
```

- Une fonction est introduite grâce au mot clé `def`
- On indique ensuite son nom et on donne sa liste d'arguments entre parenthèses
- Le corps de la fonction est un *bloc d'instructions*, donc indenté, qui vient après sa déclaration.

Définir une fonction en Python

Un premier exemple

```
def multiplie(x):
    """ Cette fonction multiplie son argument par 2 et l'affiche"""
    print(x*2)

multiplie(7)
multiplie("aaa")
```

- Le bloc de la fonction peut être précédé d'un commentaire, appelé **docstring** qui sert à générer la documentation automatique de `help()`. Un docstring est entouré de trois guillemets : `""" docstring """`.

Écrire une fonction

- Les fonctions retournent une **valeur**, donnée après le mot clé `return`.
- Même les fonctions qui n'ont pas d'instruction `return`, ou qui terminent en finissant d'exécuter le bloc de code sans atteindre un `return`, retournent la valeur `None`.

Syntaxe

```
def nom_fonction(arg 1, arg2, ..., argn):
        instruction1
        instruction 2
        ...
        return objet
```

Avec return

```
def multiplie(x):
        return x*2

print(type(multiplie(2)))
```

Sans return

```
def multiplie(x):
        print(x*2)

print(type(multiplie(2)))
```

Signature d'une fonction

- En programmation, la *signature d'une fonction* est donnée par son nom, les paramètres, leur type et le type renvoyé.
- En Python, pas besoin de donner le type renvoyé lors de la déclaration, ni le type des paramètres.

Que se passe-t-il lors de l'appel de la fonction ?

```
def somme(x,y):
    return x+y

print(somme("hello", 1))
```

Sortir de la fonction

- Attention, une fonction termine dès qu'elle atteint le mot clé `return`.
- C'est utile pour gérer le flot d'exécution d'une fonction.

Exemple

```
def un():
    return 1
    print("je ne serai jamais atteint")
un()

def cherche(x,l):
    for elem in l:
        1 / elem #erreur quand elem vaut 0
        if(x == elem):
            return 1
    return 0

cherche(10, [1,10,0]) #la valeur 0  pas atteinte car on trouve 10 avant
```

Type de return

- En Python, on peut utiliser des types de retour complexes (contrairement à C par exemple).
- On peut par exemple renvoyer des tuples de valeurs, ce qui est souvent pratique.

Exemple

```
def ordonne(a, b):
    if a < b:
        return a, b
    else:
        return b, a

print(ordonne(10, 5))
```