

Les fonctions

Alexandre Benoit

BCPST

I Introduction

On souhaite calculer l'âge de 3 personnes.

```
annee = 2015
date = int(input("Entrez votre annee"))
print("Vous avez " + str(annee-date) + " ans")
date = int(input("Entrez votre annee"))
print("Vous avez " + str(annee-date) + " ans")
date = int(input("Entrez votre annee"))
print("Vous avez " + str(annee-date) + " ans")
```

On répète 3 fois la même ligne.

On va écrire une fonction pour encapsuler les lignes redondantes.

```
def calcul_age():  
    annee = 2015  
    date = int(input("Entrez votre annee"))  
    print("Vous avez "+str(annee-date)+" ans")
```

Une fonction est une suite d'instructions.

On appellera la fonction de la manière suivante :

```
calcul_age()  
calcul_age()  
calcul_age()
```

Premier paramètre

Pour ne pas avoir une redondance de l'année, on peut la mettre en paramètre

```
def calcul_age(annee):  
    date = int(input("Entrez votre annee"))  
    print("Vous avez "+str(annee-date)+" ans")
```

On appellera la fonction de la manière suivante :

```
annee=2015  
calcul_age(annee)  
calcul_age(annee)  
calcul_age(annee)
```

II Retour de valeur

On se donne le code suivant :

```
def carre(x):  
    x*x  
y = carre(5)  
print(y)
```

Qu'affiche ce programme quand il est lancé ?

La bonne solution : **return**

On retourne la valeur de x pour obtenir l'affichage

```
def carre(x):  
    return x*x  
y = carre(5)  
print(y)
```

III Variable locale contre globale

Lorsque l'on initialise une variable dans une fonction, celle-ci existe localement à la fonction.

```
def carre(x):  
    y = x*x  
carre(5)  
print(y)
```

Qu'affiche ce programme quand il est lancé ?

Une solution est de déclarer globalement la fonction :

```
def carre(x):  
    global y  
    y = x*x  
carre(5)  
print(y)
```

De façon générale, une bonne pratique consiste à utiliser les variables globales pour représenter les **constantes** du problème. En pratique, on ne devrait pas recourir souvent à la construction global de Python.

Comme pour les fonctions, il est préférable de donner aux variables globales des noms longs et explicites, ce qui les distinguera de fait des variables locales qui portent habituellement des noms courts (comme les paramètres formels).

IV Passage par valeur

Évolution d'une variable après un passage par valeur

On se donne la fonction f suivante :

```
def f(x):  
    x = x+1  
    return x
```

Qu'affiche ce programme ?

```
a = 4  
print(f(a))  
print(a)
```

Lors de l'appel de la fonction f , une variable x contenant la valeur 4 s'est créée.

Lorsque la fonction finie, la variable disparaît.

Et avec le même nom

Et si on avait

```
def f(x):  
    x = x+1  
    return x
```

Avec le programme ?

```
x = 4  
print(f(x))  
print(x)
```

Le même procédé existe avec les listes. Sauf que lors de l'appel de la fonction, on ne copie pas le contenu de la liste mais l'adresse des contenues.

```
def f(l):  
    l[0] = 5  
    return l
```

Qu'affiche ce programme ?

```
L = [4]  
print(f(L))  
print(L)
```

V Fonctions de bibliothèque

Ne jamais réinventer la roue

Les fonctions les plus utiles sont déjà écrites en Python. On ne cherche donc pas à les réécrire.

On a déjà vu par exemple la fonction **len**

Beaucoup de ces fonctions sont regroupées dans des bibliothèques.

- **math** qui contient toutes les fonctions habituellement utilisées en analyse ;
- **random** qui calcule des nombres pseudo-aléatoires ;
- **fractions** qui sert à manipuler des nombres rationnels en valeur exacte ;
- **numpy** qui fournit des outils variés pour le calcul scientifique.

Appeler une fonction d'une bibliothèque

Pour importer une fonction, on utilise la commande :

```
import random
```

Pour appeler une fonction, on utilise la commande :

```
random.randint(0,5)
```

On peut aussi utiliser un alias :

```
import random as rand  
rand.randint(0,5)
```

Ou encore :

```
from random import randint  
randint(0,5)
```

- Définir une fonction, savoir utiliser `def`
- Retourner une valeur
- Appeler une fonction et afficher la valeur retournée par celle-ci.
- Connaître la différence entre variable globale et locale.
- Comprendre ce qu'il se passe avec une liste dans une fonction
- Savoir faire les exercices 1 à 8

Ce cours est issu en grande partie du livre *Informatique pour tous* de Wack et autres aux éditions **Eyrolles**