Python au lycée (4) : Les fonctions

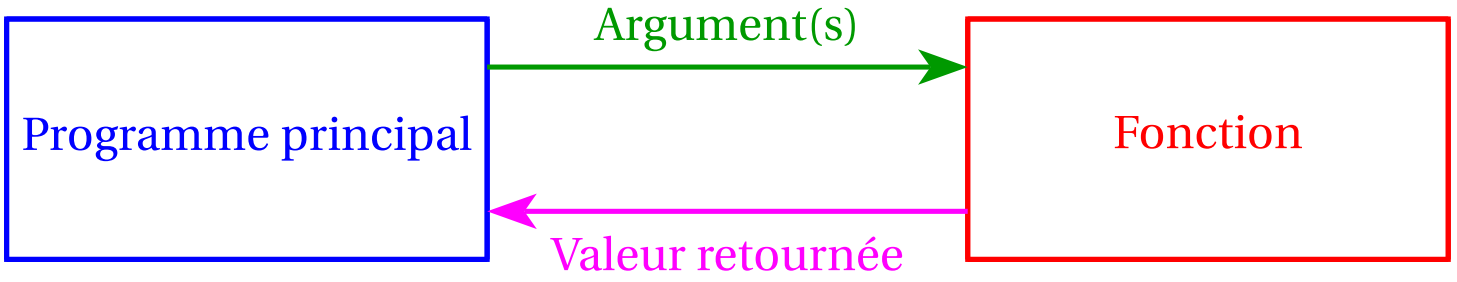
Fonctions en programmation

Lorsqu'un groupe d'instructions se répète plusieurs fois dans un programme, il peut être utile de regrouper ces instructions à l'intérieur de *fonctions.*

Cela permet de diminuer la taille du programme, de faciliter sa maintenance et de le rendre plus lisible en fractionnant le code en blocs indépendants. Par ailleurs, un programmeur peut ainsi utiliser un code écrit par une autre personne.

Dans les chapitres précédents, nous avons déjà utilisé des fonctions comme print() ou len() qui sont des fonctions internes à Python.

L'appel à une fonction peut être schématisé de la manière suivante :

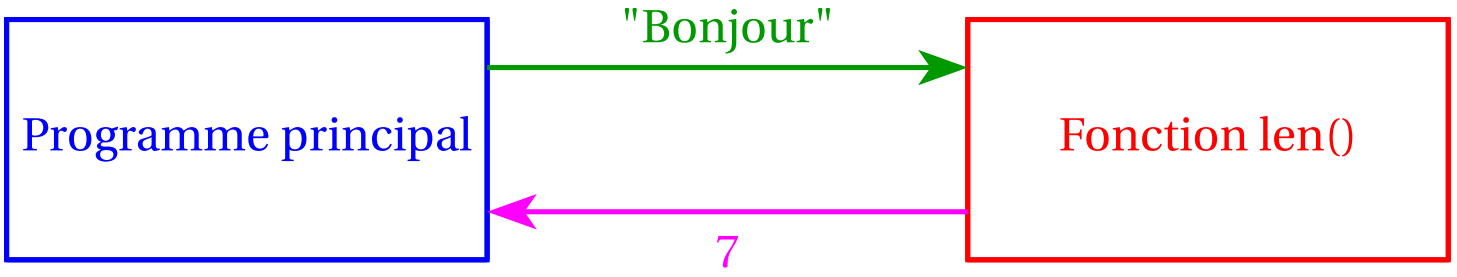


- Le programme appelle la fonction en lui passant éventuellement certaines valeurs appelées *arguments* ou *paramètres*. Il peut y avoir zéro, un ou plusieurs paramètre(s) de n'importe quel type.
- La fonction effectue certaines opérations en utilisant, au besoin, les valeurs passées en paramètre.
- La fonction peut ensuite retourner un résultat au programme qui l'a appelé ; le retour d'une valeur n'est toutefois pas obligatoire.

Considérons, par exemple, l'instruction :

```
x=len("Bonjour")
```

L'exécution de cette commande se déroule de la manière suivante :



- le programme appelle la fonction Python len() en lui passant le paramètre « Bonjour » de type chaîne de caractères ;
- la fonction len() calcule la longueur de cette chaîne ;
- la fonction renvoie l'entier 7 qui correspond à cette longueur ;
- le résultat est stocké dans la variable x qui pourra être utilisé par la suite.

Dans les exemples précédents, les fonctions ont été appelées à partir du programme principal. Cependant, une fonction peut être appelée à partir de n'importe où ; en particulier, une fonction peut très bien être appelée par une autre fonction.

Définition et appel d'une fonction en Python

En plus des fonctions prédéfinies par Python, le programmeur a la possibilité de définir ses propres fonctions.

On utilise le mot-clé def pour définir une nouvelle fonction avec la syntaxe suivante :

```
def nom_de_la_fonction(liste_des_paramètres):
    # bloc d'instructions
    # calculant le résultat
    return resultat # facultatif
```

Par exemple :

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

Les paramètres passés à la fonction sont placés entre parenthèses. Même si la fonction n'admet pas d'argument, il faut utiliser des parenthèses ; dans ce cas on ne place rien entre les parenthèses.

Lorsque l'on définit une fonction en Python, aucune instruction n'est exécutée. L'exécution des instructions situées dans le corps de la fonction n'a lieu que lors de l'**appel** de la fonction.

L'appel d'une fonction en Python obéit à la syntaxe suivante :

```
nom_de_la_fonction(valeurs_des_paramètres)
```

Le résultat est alors souvent stocké dans une variable :

```
resultat=nom_de_la_fonction(valeurs_des_paramètres)
```

ou affiché à l'écran :

```
print(nom_de_la_fonction(valeurs_des_paramètres))
```

Par exemple, pour utiliser la fonction ajoute définie précédemment :

```
x=ajoute(4,3) # x contient la valeur 7
```

Les modules Python

En Python, il est possible de regrouper des fonctions (ou d'autres objets) dans des fichiers appelés *modules* afin de pouvoir les réutiliser par la suite dans d'autres programmes.

Un certain nombre de modules sont intégrés à Python mais il est nécessaire de les charger dans son programme afin de pouvoir les utiliser. Pour charger un module on peut utiliser la syntaxe suivante :

```
import nom_du_module
```

Par la suite on peut utiliser une fonction de ce module en la préfixant par le nom du module de la manière suivante :

```
resultat = nom_du_module.nom_de la fonction # Noter la présence du point
```

Par exemple pour utiliser la fonction sqrt() (racine carrée) du module math :

```
import math
print(math.sqrt(9)) # affiche 3.0
```

Pour éviter de devoir préfixer la fonction à chaque fois qu'on l'utilise, on peut choisir de charger cette fonction lors de l'import en utilisant la syntaxe suivante :

```
from math import sqrt
print(sqrt(9)) # affiche 3.0
```

On peut également importer tous les objets d'un module en utilisant une * :

```
from math import *
print(sqrt(9)) # affiche 3.0
print(cos(pi)) # affiche -1.0
```

Parmi les nombreux modules intégrés à Python, les modules math et random seront fréquemment utilisés au lycée :

- **Le module « math »** :

  Ce module contient différentes fonctions et constantes mathématiques : pi, sqrt(), exp(), ln(), sin(), cos(), ...

- **Le module « random »** :

  Ce module, utile en statistiques et en probabilités, permet de génères des nombres aléatoires.

  - random() renvoie un nombre réel aléatoire compris entre 0 et 1.
  - randint(min, max) renvoie un nombre entier aléatoire compris entre les entiers min et max (bornes incluses).

Par exemple on peut simuler dix lancers d'un dé à six faces grâce au programme suivant :

```
from random import randint
for i in range (10):
    print(randint(1,6))
```