

Vidéo Fonction

Une **fonction** est un **ensemble d'instructions**, dépendant généralement d'un ou de plusieurs paramètres, qui effectue une **action** et cette fonction peut être appelée depuis n'importe où dans le programme.

Ecrire une **fonction** permet:

1. **d'éviter les répétitions** de code;
2. **décomposer un programme complexe** en une série de sous-programmes plus simples
3. de **mieux structurer un programme**;
4. **permet la réutilisation**. Une même fonction peut être utilisée dans différents programmes.

Définition et appel

Il est nécessaire de bien comprendre la différence entre l'écriture de la **définition** d'une fonction et l'**appel** de cette fonction.

The diagram shows two parts of Python code. Part 1, labeled with a red circle containing the number 1, is a function definition: `def un_a_nb(mon_nombre):` followed by an indented `for i in range(1, mon_nombre+1):` loop and a `print(i)` statement. A red bracket to the right of this code block is labeled "définition de la fonction". Part 2, labeled with a red circle containing the number 2, is a function call: `un_a_nb(5)`. A red arrow points from this call to a yellow box labeled "appel de la fonction". A callout box with a pointer to the function definition contains the text: "Ce code n'est pas interprété par Python tant que la fonction n'est pas appelée ! Par contre, il met le code en mémoire."

```
1 def un_a_nb(mon_nombre):  
    for i in range(1, mon_nombre+1):  
        print(i)  
  
2 un_a_nb(5)
```

>>> 1
>>> 2
>>> 3
>>> 4
>>> 5

définition de la fonction

appel de la fonction

Ce code n'est pas interprété par Python tant que la fonction n'est pas appelée ! Par contre, il met le code en mémoire.

Types des fonctions

Fonctions qui renvoient une valeur

Fonction qui calcule le cube d'un nombre

```
def cube(mon_nombre):
    return mon_nombre**3
```

```
a = cube(2)
b = cube(5)
print(a+b)
```

- La valeur renvoyée par la fonction peut-être mémorisée par exemple `a = cube(2)`.
- La fonction peut être appelée plusieurs fois.

[Vidéo nombre premier](#)

fonction qui vérifie si un nombre est premier

```
def premier(nb):
    for d in range(2,nb):
        if nb % d == 0:
            return False
    return True
```

```
a = premier(7)
```

[Vidéo nombre premier avec exit\(\)](#)

- Il peut y avoir plusieurs instructions **return** mais chacune interrompt l'exécution de la fonction !

```
def trois_au_cube():
    return 3**3
```

```
resultat = trois_au_cube()
print(resultat)
>>> 27
```

fonction
sans
paramètre
et avec
un renvoi
(return) d'une valeur

Fonctions qui ne renvoient rien

```
def compteur3():
    i = 0
    while i < 3:
        print(i)
        i = i + 1
```

```
compteur3()
>>> 0
>>> 1
>>> 2
```

fonction
sans
paramètre
et sans
renvoi
(pas de
return)

Espace et portée des variables

variable locale

```
def test():  
    b = 5  
    print(a, b)
```

variable globale

```
a = 2  
b = 7  
test()  
print(a, b)
```

```
>>> 2 5  
>>> 2 7
```

- Dans la fonction `test`, la variable `b` est **accessible uniquement** à l'intérieur du corps de la fonction. On dit que cette variable est une variable "locale à la fonction".
- Elle peut porter le **même nom** qu'une variable **globale**.
- Le contenu d'une variable **locale** est stocké dans un espace **inaccessible** depuis l'extérieur de la fonction !
- Une variable **globale** est visible à l'intérieur d'une fonction mais la fonction **ne peut pas la modifier** !