

Licence Creative Commons  
MAJ: 18 août 2022

Une « piqure » de Python



1 Les fonctions

Une **fonction** est un **bloc d'instructions** qui a reçu un nom, dont le fonctionnement dépend d'un certain nombre de paramètres (les **arguments** de la fonction) et qui renvoie un résultat (au moyen de l'instruction **retourne**).

```
def ma_fonction():  
    instruction_1  
    instruction_2  
    ...  
    return
```

Annotations:

- mot réservé "def"
- nom bien choisi
- parenthèses
- deux points
- les instructions
- fin de la fonction

Exemple d'une fonction possédant un seul argument *a* :

```
1 def calcul_carre(a):  
2     cube = a * a      # ou bien a**2  
3     return cube
```

2 Les instructions conditionnelles

2.1 Si... alors

```
if condition:  
    instruction_1  
    instruction_2  
    ...  
instructions suivantes
```

Annotations:

- mot réservé "if"
- une condition
- deux points
- bloc d'instructions indenté sera exécuté uniquement si la condition est vérifiée
- suite du programme

Voici un exemple :

```
1 if vitesse > 110:  
2     print("Attention, tu roules trop vite.")
```

2 2 Si...alors...sinon ...

```
if condition:
    instruction
    instruction
    ...
else:
    instruction
    ...
instructions suivantes
```

bloc exécuté
si la condition est vérifiée

bloc exécuté
si la condition n'est pas vérifiée

3 Boucle bornée

3 1 Répéter n fois

Une **boucle** permet de répéter plusieurs fois de suite un même traitement. Lorsque le nombre n d'itérations est connu à l'avance, on parle de **boucle bornée**.

3 2 Compteur

Afin de compter le nombre d'itérations, on utilise un compteur initialisé par exemple à 1 qui s'incrémente automatiquement de 1 à chaque itération jusqu'à la valeur n .

Algorithme

```
Pour  $i$  allant de 1 à  $n$  :
    | Traitement
Fin Pour.
```

Langage Python

```
for  $i$  in range(1,  $n + 1$ ) :
    ...
    ...
```

```
for i in range(n+1):
    instruction_1
    instruction_2
    ...
instructions suivantes
```

mots réservés "for" et "in"

une variable

liste à parcourir, ici 0, 1, 2, ..., n

deux points

bloc d'instructions indenté
sera répété n fois
pour $i = 0$, puis $i = 1 \dots$ jusqu'à $i = n$

indentation

suite du programme

4 Boucle non bornée

4 1 Répéter tant que...

Dans une boucle, le nombre d'itérations peut dépendre d'une condition. Le traitement est alors répété tant que la condition est vraie. Lorsque la condition est fausse, on sort de la boucle et la suite des instructions des programmes s'applique.

4 2 Nombre d'itérations

Le nombre d'itérations n'est donc en général pas prévu à l'avance et on parle de **boucle non bornée**.

Algorithme

```
Tant que condition :
| traitement
Fin de Tant que.
```

Langage Python

```
While ...
...
...
```

Diagram illustrating the mapping between the algorithm and Python code:

- mot réservé 'while'**: Points to the `while` keyword in the Python code.
- une condition**: Points to the `condition` in the algorithm and the `condition` in the Python code.
- deux points**: Points to the colon `:` in the Python code.
- bloc d'instructions indenté sera exécuté tant que la condition est vérifiée**: Points to the indented block of instructions in the Python code.
- suite du programme**: Points to the `instructions suivantes` in the algorithm and the code following the `while` loop in the Python code.

```
while condition:
    instruction_1
    instruction_2
    ...
instructions suivantes
```

Voici un exemple :

Ce bout de code cherche la première puissance de 2 plus grande qu'un entier n donné. La boucle fait prendre à p les valeurs 2, 4, 8, 16, ... Elle s'arrête dès que la puissance de 2 est supérieure ou égale à n , donc ici ce programme affiche 128.

```
1 n = 100
2 p = 1
3 while p < n:
4     p = 2 * p
5 print(p)
```

Entrées : $n = 100$, $p = 1$

p	« $p < n$ » ?	nouvelle valeur de p
1
2
...
...
...
...
...
...

Affichage :