PYTHON AU LYCÉE (3) : LES BOUCLES

COURS

Python au lycée (3) : Les boucles

DURÉE ESTIMÉE

15 minutes

OBJECTIFS DU CHAPITRE

- Parcourir des entiers avec for et range
- Calculer une somme avec une boucle for
- Compter avec un compteur dans une boucle
- Utiliser une boucle while pour s'arrêter à un seuil
- Choisir entre une boucle for et une boucle while

L'un des intérêts de la programmation est de pouvoir faire exécuter facilement à une machine des tâches **répétitives**.

Le langage Python propose deux instructions : « for » et « while » qui permettent de répéter automatiquement l'exécution de certains blocs de code.

1. Les boucles « for » (Boucles bornées)

Une boucle « for » (ou boucle « pour » ou boucle bornée) est généralement utilisée lorsque l'on connaît le nombre de répétitions que l'on souhaite exécuter.

La syntaxe de cette instruction est la suivante :

```
for variable in [liste de valeurs] :
    # bloc d'instructions à répéter
    # instructions à exécuter une fois la boucle terminée
```

Ce programme se déroule de la manière suivante :

- les « instructions à répéter » sont exécutées en donnant à la « variable » chacune des valeurs de la « liste de valeurs » ; c'est l'indentation (écriture décalée vers la droite) qui détermine la taille du bloc d'instructions à répéter
- une fois que tous les items de la liste ont été parcourus, le programme passe au « instructions à exécuter une fois la boucle terminée ».

Par exemple, le programme Python ci-dessous :

```
for i in [1, 2, 5, 11] :
    j = i**2   # calcul du carré de i
    print(j, end=' - ') # affichage; on sépare les valeurs par de
print("fin")
```

affichera :
1 - 4 - 25 - 121 - fin
ce qui correspond aux carrés des nombres de la liste.
Remarque : on aurait pu faire l'économie de la variable j, en écrivant plus simplement « print(i**2, end=' - ') » mais le but, ici, était de montrer qu'un bloc pouvait comporter plusieurs lignes.

Avec l'instruction *for* on utilise fréquemment la fonction *range* ; en effet, *range(a,b)* permet de parcourir successivement les entiers de *a* à *b - 1* (inclus).

> ⚠ **Attention**
>
> L'instruction *range(a,b)* parcourt les entiers de *a* à *b - 1* et non à l'entier *b* !

Par exemple, le programme suivant affiche les doubles des nombres entiers compris entre 3 et 5 :

```
for i in range(3, 6) :
   print(2*i, end=' - ') # affiche 6 - 8 - 10 -
```

Remarque : si l'on utilise l'instruction *range* avec un seul paramètre *b*, celle-ci parcourt les entiers de 0 à *b - 1*:

```
for i in range(4):
   print(i, end=' - ') # affiche 0 - 1 - 2 - 3 -
```

Application : calcul d'une somme

L'utilisation la plus fréquente de la boucle *for* en mathématiques est le calcul d'une somme. Par exemple, pour calculer $S = 1 + 2 + 3 + \ldots + 100$, on écrit :

```
S = 0                      # on initialise la somme à 0
for i in range(1, 101) :   # i prend les valeurs 1, 2, 3, ..., 1(
    S = S + i              # on ajoute i à la somme
print(S)                   # affiche 5050
```

Pour bien comprendre ce programme, on peut représenter les valeurs successives prises par *i* et *S* dans un tableau :

Étape	1	2	3	4	5	...	100
i	1	2	3	4	5	...	100
S	1	3	6	10	15	...	5050

En adaptant ce schéma, on peut aussi calculer une somme de carrés, un produit, un maximum, etc.

2. Les boucles « while » (Boucles non bornées)

On utilise une boucle *while* (ou boucle « Tant que » ou boucle non bornée) lorsque l'on doit répéter l'exécution d'un bloc d'instructions **tant qu'une condition est vérifiée** (mais en général, on ne sait pas au préalable le nombre de répétitions que l'on devra effectuer). Là encore, c'est l'indentation qui détermine la fin du bloc d'instructions à répéter.

La syntaxe de l'instruction « while » est :

```
while condition :
    # bloc d'instructions à répéter
    # instructions à exécuter une fois la boucle terminée
```

Le programme se déroule alors de la façon suivante :

- tant que la « condition » de la ligne 1. est vraie, les « instructions à répéter » de la ligne 2. sont exécutées
- dès que la « condition » de la ligne 1. devient fausse, le programme passe aux « instructions à exécuter une fois la boucle terminée » (ligne 3.).

Par exemple, le programme ci-dessous affiche la plus petite puissance de 2 qui est supérieure ou égale à 1 000 :

```
i = 1 # initialisation de i
while i < 1000 :
   i = i * 2
print(i) # affiche 1024
```

À chaque passage à la ligne 3, on multiplie la valeur précédente de *i* par 2. On obtient ainsi les puissances successives de 2.
Pour bien comprendre comment fonctionne ce programme, il peut être utile de représenter les différentes valeurs de *i* et de la condition *i < 1000* dans un tableau :

i	1	2	4	8	16	32	64	128	256	512	1024
i < 1000	vrai	vrai	vrai	vrai	vrai	vrai	vrai	vrai	vrai	vrai	faux

L'utilisation de l'instruction *while* peut être assez délicate ; si, suite à une erreur de programmation, la condition figurant dans le *while* est toujours vérifiée, le programme ne sortira jamais de la boucle et tournera indéfiniment (sauf si un intervenant extérieur met fin à son exécution...)

En particulier, il faut faire attention aux points suivants :

- penser à initialiser les variables avant l'instruction *while* : dans notre exemple, python provoquerait une erreur si la ligne 1. (initialisation de i) était manquante.
- la condition figurant après l'instruction *while* est celle qui permet de **rester** dans la boucle ; c'est donc le **contraire** de la condition de **sortie** de boucle. Dans l'exemple précédent, on souhaitait **sortir** de la boucle lorsque la valeur de *i* était **supérieure ou égale à 1000** ; il fallait donc coder *i < 1000* à l'intérieur de l'instruction *while*.

Les questions essentielles

1. Comment parcourir des entiers avec for et range ?

On écrit *for i in range(a, b) :* pour que *i* prenne les valeurs a, $a+1$, ..., $b-1$. La borne supérieure b est **exclue**.

Voir la fiche méthode : Parcourir des entiers avec for et range

2. Comment calculer une somme avec une boucle for ?

On initialise une variable *S = 0* **avant** la boucle, puis à chaque tour on écrit *S = S + terme*. Après la boucle, *S* contient la somme.

Voir la fiche méthode : Calculer une somme avec une boucle for

3. Comment compter combien d'éléments vérifient une condition ?

On initialise un compteur *c = 0*, puis dans la boucle on teste la condition avec *if* : si elle est vraie, on fait *c = c + 1*.

Voir la fiche méthode : Compter avec un compteur dans une boucle

4. Comment utiliser une boucle while pour atteindre un seuil ?

On met dans le *while* la condition pour **rester** dans la boucle (contraire de la sortie). On initialise les variables avant le *while* et on les fait évoluer à chaque tour.

Voir la fiche méthode : Utiliser une boucle while pour s'arrêter à un seuil

5. Comment choisir entre for et while ?

Si le nombre de répétitions est connu à l'avance, utiliser *for*. Sinon (condition à vérifier, seuil à dépasser), utiliser *while*.

Voir la fiche méthode : Choisir entre une boucle for et une boucle while

Télécharger en PDF