

La boucle bornée

Les boucles, en programmation, servent à répéter plusieurs fois les mêmes instructions.

On a vu que la boucle non bornée (`while` en Python) permettait de répéter un bloc d'instruction tant qu'une condition n'était pas atteinte : on peut effectuer une répétition même lorsqu'on ne connaît pas à l'avance le nombre exact de tours de boucle qu'il faudra.

Une boucle bornée est utile lorsqu'on connaît **à l'avance** le nombre de tours de boucle à effectuer.

Exercice 1

On utilise une boucle `for` pour répéter l'affichage de la chaîne de caractères "NSI". Le nombre de fois que l'instruction est répétée est définie par les paramètres de la fonction `range`. Cette dernière fait partie de la bibliothèque standard Python.

1. `for _ in range(5):
 print("NSI")`

(a) Combien de fois cette chaîne apparaît-elle dans la console ?

(b) Modifier le code pour que "SNT" s'affiche 10 fois.

2. `for i in range(5):
 print(i)`

(a) Qu'est-ce qui a changé dans la syntaxe de la boucle par rapport à la question précédente ?

(b) Le paramètre du `range` est toujours 5. Noter la liste des valeurs qui s'affichent à l'exécution du script.

3. `for i in range(1, 5):
 print(i)`

(a) Qu'est-ce qui a changé dans la syntaxe de la boucle par rapport à la question précédente ?

(b) De combien à combien vont les valeurs qui s'affichent ?

4. `for i in range(5, 1, -1):
 print(i)`

(a) Qu'est-ce qui a changé dans la syntaxe de la boucle par rapport à la question précédente ?

(b) En se basant sur l'affichage correspondant à ce script, à quoi peut correspondre le -1 écrit dans le range ?

5. Utiliser ce que vous avez vu dans les questions précédentes pour afficher les entiers de 1 à 10 compris.

6.

```
for i in range(1, 10, 2):
    print(i)
```

(a) Quelles valeurs le programme affiche-t-il ?

(b) Le modifier pour afficher cette fois-ci les multiples de 3 de 1 à 10 compris.

7. Afficher tous les entiers de 0 à 40 compris.

8. Afficher tous les entiers de 40 à 0 compris.

9. Afficher les multiples de 5 en partant de 40 et en allant jusqu'à 0 compris.

Définition

Une **boucle bornée** (qui a des bornes, de départ et d'arrivée) permet de répéter des instructions un nombre de fois déterminé à l'avance. On utilise souvent une variable stockant une valeur différente à chaque nouvelle itération, dite variable de boucle.

Un boucle bornée s'écrit en Python à l'aide du mot-clef `for`.

→ De manière générale, la valeur contenue dans la **variable de boucle** est prise dans une **séquence** (une suite de valeurs) qui peut être créée grâce à la fonction `range`.

Exemple : Pour afficher tous les entiers de 1 à 10 avec une boucle bornée, l'algorithme est le suivant :

```
for i in range(1, 11):
    print(i)
```

→ Le `i` est une variable créée avec la boucle, dont la valeur change automatiquement à chaque nouveau passage (à chaque nouvelle **itération**).

→ La façon dont `i` varie est déterminée par la **fonction** `range` qui permet d'obtenir une séquence de nombres à partir d'un nombre de **départ**, **d'arrivée** et éventuellement d'un **pas** (c'est-à-dire la différence entre deux nombres successifs de la séquence).

Voici un extrait de la documentation de la fonction `range` :

```
range(start, stop[, step])
[...]
start
    Valeur du paramètre start (ou 0 si le paramètre n'a pas été fourni)

stop
    Valeur du paramètre stop

step
    Valeur du paramètre step (ou 1 si le paramètre n'a pas été fourni)
```

Exercice 2

Concernant la fonction `range` :

1. S'il n'y a qu'un paramètre, que désigne-t-il ?

2. S'il y a 2 paramètres, que désignent-ils ?

3. Même question s'il y en a 3.

4. Quelle est la valeur de début par défaut ? La valeur du pas par défaut ?

Exercice 3

Traduire en Python les algorithmes suivant écrits en pseudo-code (langage naturel).

Pseudo-code (langage naturel)	Python
Pour <code>i</code> dans séquence instructions	
Pour <code>j</code> variant de 0 à <code>fin-1</code> instructions	
Pour <code>k</code> variant de <code>début</code> à <code>fin-1</code> instructions	
Pour <code>l</code> variant de <code>début</code> à <code>fin-1</code> en variant de <code>pas</code> instructions	

Exercice 4 – (Boucles sur des entiers)

Soit le script suivant.

```
for i in range(1,11):
    print(2*i)
```

1. Combien de fois l'instruction de la boucle `for` va-t-elle être exécutée ?

2. Compléter la table de trace suivante. Attention, la première colonne correspond ici au numéro d'itération !

Itération	i	2*i
1
2
...

3. En déduire ce qu'affiche ce programme.

4. Qu'affichent les programmes suivants ? Vous appliquerez au besoin la même méthode que précédemment (attention aux paramètres du `range`!).

```
for i in range(0,10):    for i in range(0,5):    for i in range(1,100,10):
    print(3*i)              print(i**2)          print(i//10)
```

Exercice 5

1. Transformer le script suivant pour utiliser une boucle bornée à la place d'une boucle non-bornée. Quelle valeur contient la variable `somme`, dans les deux cas, après exécution du code ?

```
c = 0
somme = 0
while c < 5:
    somme = somme + c
    c = c + 1
```

2. Transformer le script suivant pour utiliser une boucle non-bornée à la place d'une boucle bornée. Quelle valeur contient la variable total, dans les deux cas, après exécution du code ?

```
total = 0
for i in range(5):
    total = total + i**2
```

Exercice 6

On a utilisé dans la fonction ci-dessous une boucle while.

```
1 def truc(n):
2     x = 1
3     k = 1
4     while k < n:
5         x = 2*x
6         k = k+1
7     print(x)
```

1. Quelle est la valeur affichée lors de l'appel truc(5) ?
2. Proposer une version de la fonction truc utilisant une boucle for.

Exercice 7

Compléter la fonction suivante pour qu'elle affiche la somme de tous les entiers naturels inférieurs ou égaux au paramètre n.

```
1 def somme(n):
2     s = ...
3     for ...
4         s = s+k
5     print(s)
```

Exercice 8

En théorie, une boucle bornée se termine toujours (contrairement aux boucles non bornées, d'où la terminologie).

En pratique, cela n'est pas forcément le cas, comme dans l'exemple (stupide) ci-dessous.

```
1 n = 10**12
2 k = 0
3 for i in range(n):
4     k = k+1
5 print(k)
```

Quelle doit être la valeur affichée par l'instruction ligne 5 ? Combien de tours de boucle sont nécessaires ?