Python la boucle for

Indentation, itération

La boucle for en résumé

- Dans un programme, on peut avoir besoin de répéter des actions similaires (calculs, affichages, etc.) un nombre connu de fois. On utilise alors une instruction appelée boucle bornée ou boucle for.

L'instruction `for` est une instruction composée, c'est-à-dire une instruction dont l'en-tête se termine par deux-points `:`, suivie d'un bloc indenté qui constitue le **corps de la boucle**.

On dit que l'on réalise une **itération** de la boucle à chaque fois que le corps de la boucle est exécuté.

Dans l'en-tête de la boucle, on précise après le mot-clé `for` le nom d'une variable (`i` dans l'exemple ci-dessus) qui prendra successivement toutes les valeurs qui sont données après le mot-clé `in`. On dit souvent que cette variable (ici `i`) est un **compteur** car elle sert à numéroter les itérations de la boucle.

La boucle « for *i* in range(*n*) » prend toutes les valeurs entières de 0 à *n* – 1 inclus. La syntaxe pour écrire en langage Python une telle boucle est :
for *i* in range (*n*):
<Instructions>

- La boucle « for *i* in range(*n*,*m*) » prend toutes les valeurs entières de *n* à *m* – 1 inclus. La syntaxe pour écrire en langage Python une telle boucle est :
  for *i* in range (*n*,*m*):
  <Instructions>
- La boucle « for *i* in range(*n*,*m*,*p*) » prend toutes les valeurs entières de *n* à *m* – 1 inclus, avec un pas de *p*. La syntaxe pour écrire en langage Python une telle boucle est :
  for *i* in range (*n*,*m*,*p*):
  <Instructions>

Boucle for in range (n) :

a. Principe
Dans un programme, on peut avoir besoin de répéter des actions similaires (calculs, affichages, etc.) un nombre connu de fois. On utilise alors une instruction appelée **boucle bornée** ou **boucle for**.

La boucle **« for *i* in range(*n*) »** prend toutes les valeurs entières de 0 à *n* – 1 inclus. La syntaxe pour écrire une telle boucle est :

Langage naturel	Langage Python

$a \leftarrow 0$
Pour *i* allant de 0 à 2
  $a \leftarrow a + 2$
Fin Pour
	`a=0`
	`for i in range(3):`
	`    a=a+2`
	`print(a)`

Remarques
La variable *i* prend successivement toutes les valeurs entières de 0 à *n* – 1 inclus, soit *n* valeurs différentes.
Pour chaque valeur de *i*, la boucle « for *i* in range(*n*) » exécute les instructions.

b. Exemple

Langage naturel	Langage Python

$a \leftarrow 0$
Pour i allant de 0 à 2
$a \leftarrow a + 2$
Fin Pour
	`a=0`
	`for i in range(3):`
	`  a=a+2`
	`print(a)`

L'instruction conditionnelle permet d'ajouter, trois fois de suite, le nombre 2 à la variable *a*. On peut résumer les étapes dans un tableau :

Valeurs prises par *i*	0	1	2
Valeurs prises par *a*	0	2	4 **6**

Le résultat affiché est alors 6 (0 + 2 + 2 + 2).

Remarque

Voici les erreurs de saisie les plus classiques :

```
for i in range(3)
    a=a+2
```

Oublier les deux points.

```
for i in range(2):
    a=a+2
```

Se tromper sur le choix de la valeur de n (ici i ne prend que les valeurs 0 et 1).

```
for i in range(3):
a=a+2
```

Oublier ou effacer par mégarde la tabulation (indentation) de la ligne 3.

Boucle for in range (n, m) :

2. Boucle du type « for i in range(n,m) »

a. Principe

La boucle « for *i* in range(*n*,*m*) » prend toutes les valeurs entières de *n* à *m* – 1 inclus. La syntaxe pour écrire une telle boucle est :

Langage naturel	Langage Python
Pour *i* allant de *n* à *m* – 1 <Instructions> Fin Pour	for *i* in range(*n*,*m*): <Instructions>

Remarques

La variable *i* prend successivement toutes les valeurs entières de *n* à *m* – 1 inclus, soit (*m* – *n*) valeurs différentes.

Pour chaque valeur de *i*, la boucle « for *i* in range(*n*,*m*) » exécute les instructions. Ces dernières sont donc exécutées (*m* – *n*) fois.

b. Exemple

Langage naturel

$a \leftarrow 0$
Pour i allant de 4 à 6
  $a \leftarrow a + i$
Fin Pour

Langage Python

```
a=0
for i in range(4,7):
    a=a+i
print(a)
```

La boucle permet d'ajouter à la variable a les valeurs prises par la variable i, c'est-à-dire les nombres 4, puis 5 et enfin 6. On peut résumer les étapes dans un tableau :

Valeurs prises par i		4	5	6
Valeurs prises par a	0	4	9	15

Le résultat affiché est alors 15 (0 + 4 + 5 + 6).

Remarque
Les instructions « for i in range(n) » et « for i in range(0,n) » sont équivalentes.

Boucle for in range (n, m, p) :

a. Principe
La boucle « for i in range(n,m,p) » prend toutes les valeurs entières de n à $m - 1$ inclus, avec un pas de p. La syntaxe pour écrire une telle boucle est :

Langage naturel	Langage Python
Pour i allant de n à $m - 1$ avec un pas de p <Instructions> Fin Pour	for i in range(n,m,p): <Instructions>

Remarque
La variable i prend successivement toutes les valeurs entières de n, $n + p$, $n + 2p$, $n + 3p$, etc. à $m - 1$.

b. Exemple

Langage naturel

$a \leftarrow 0$
Pour i allant de 1 à 5 avec un pas de 2
  $a \leftarrow a + i$
Fin Pour

Langage Python

```
a=0
for i in range(1,6,2):
    a=a+i
print(a)
```

La boucle permet d'ajouter à la variable a les valeurs prises par la variable i, c'est-à-dire les nombres 1, puis 3 et enfin 5. On peut résumer les étapes dans un tableau :

Valeurs prises par i		1	3	5
Valeurs prises par a	0	1	4	9

Le résultat affiché est alors 9 (0 + 1 + 3 + 5).

Remarque
Les instructions « for i in range(n) » et « for i in range(0,n,1) » sont équivalentes.