Boucles inconditionnelles

L’objectif des boucles est de répéter plusieurs fois un ensemble d’instructions. Chaque répétition de ces instructions est appelé itération.

Il existe deux types de boucle :

Une boucle "Pour" (ou boucle inconditionnelle) qui sert à répéter une nombre de fois fini plusieurs procédures.

Une boucle "Tant que" (ou boucle conditionnelle) est une boucle qui se répète tant que la condition donnée reste vrai. Si cette condition devient fausse, la boucle s'arrête.

La boucle inconditionnelles.

Syntaxe python :

```
for i in range(...) :
     # bloc de la boucle pour
     # Les instructions écrites ici seront exécuté plusieurs fois
     ...
# Fin du bloc de boucle
```

La variable de boucle (ci-dessus nommé i mais que l’on peut nommer autrement) est locale au bloc de la boucle, c’est à dire qu’elle est définie uniquement dans le contexte des instructions indentées sous la ligne de code du for et plus après.

```
for i in range(...) :
     # i existe ici, nous sommes dans le bloc de boucle
     ...
# i n’existe plus ici, nous sommes sorti de la boucle
```

Lorsque l'on crée une boucle Pour, il est très important de réfléchir au nom de la variable de boucle. Une erreur classique consiste à utiliser le même nom qu'une variable déjà existante. Cela aura pour effet de supprimer l'ancienne valeur associée à la variable déjà utilisée, et peut causer un bug dans le programme.

Il se peut aussi que l’on n’ai pas besoin d’une variable de boucle, seulement que les instructions se répètent un certains nombre de fois. Dans ces cas là, la convention est de nommer la variable de boucle _. On l’appelle la variable anonyme.

```
for _ in range(...) :
     ...
```

A chaque itération de la boucle, la variable de boucle prend la valeur suivante défini par la fonction range.

`for n in range(3) :`
`    print (n)`
	`0`
	`1`
	`2`

Il existe 3 façon d'utiliser la fonction range :

- avec un seul paramètre : c'est la valeur de fin. Les valeurs de la variable de boucle iront de 0 à fin-1 avec un pas de 1 en 1.

`for a in range(3) :`
`    print(a)`
	`0`
	`1`
	`2`

- deux paramètres : ceux sont les valeurs de début et de fin. Les valeurs de la variable de boucle iront du début à la fin avec un pas de 1 en 1.

`for b in range(2, 5) :`
`    print(b)`
	`2`
	`3`
	`4`

- trois paramètres : ceux sont les valeurs de début, de fin et le pas. Les valeurs de la variable de boucle iront de début à fin-1 de pas en pas. Il est possible de boucler avec un pas positif ou négatif.

`# ici le pas est négatif`
`for c in range (0, -6, -2) :`
`    print(c)`
	`0`
	`-2`
	`-4`

Nous verrons plus tard dans l'année que la boucle pour sert aussi pour parcourir un à un les éléments de différents ensembles de valeurs.

Exercice 1

Écrire un programme qui affiche la table de 2 avec une boucle for.

Exercice 2

Écrire un programme qui demande à l'utilisateur un entier n affiche la table de n avec une boucle for.

Exercice 3

Écrivez un programme qui affiche tous les nombres entre 1 et 10, et indique pour chacun si celui-ci est pair ou impair.

Exercice 4

Écrire le code permettant d'obtenir les sorties suivantes avec des boucles (dans des boucles).

```
# Sortie 1
*
**
***
****
*****
```

```
# Sortie 2
*****
****
***
**
*
```

```
# Code d'aide

txt=""
for _ in range(...):
    txt = txt + "*"
```

Exercice 5

Écrire un programme qui calcule la somme des entiers de 1 à 100 à l'aide d'une boucle for puis affiche le résultat.

Exercice 6

En mathématiques, la suite de Fibonacci est une suite d'entiers dans laquelle chaque terme est la somme des deux termes qui le précèdent. Elle commence généralement par les termes 0 et 1 (parfois 1 et 1) et ses premiers termes sont 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, etc.

Écrivez un programme qui demande à l'utilisateur un nombre n et qui affiche les valeurs des n premiers termes de la suite de Fibonacci.