

# Les structures répétitives

Yao Aristide

# Table des matières



<b>I - Objectifs</b>	<b>3</b>
<b>II - Introduction</b>	<b>4</b>
<b>III - La structure while</b>	<b>5</b>
1. Forme de la structure while .....	5
2. Boucle finie .....	5
3. Boucle infinie .....	6
4. Arrêt instantané d'une boucle : break .....	6
5. L'instruction continue .....	6
6. Exercice .....	7
7. Exercice .....	7
8. Exercice .....	7
9. Exercice .....	8
<b>IV - La structure for</b>	<b>9</b>
1. la fonction range .....	9
2. Forme de la boucle for .....	9
3. Recommandation .....	10
4. Exercice .....	10
5. Exercice .....	10
<b>V - Conclusion</b>	<b>11</b>
<b>VI - Solutions des exercices</b>	<b>12</b>



# *Objectifs*

A la fin de cette leçon vous serez capable de :

- Écrire une structure répétitive
- Utiliser les structures répétitive pour simplifier des parties d'un programme
- Réaliser des programmes qui s'exécute en boucle

# Introduction



Un principe fondamental de la programmation est la factorisation du code. Factoriser un code consiste à l'écrire une seule fois et à le réutiliser autant de fois que possible. Une structure répétitive sert à répéter un certain nombre d'instructions d'un programme autant de fois que nécessaire de façon automatique.

Cette leçon présente les structures répétitives avec lesquelles l'apprenant pourra réaliser des programmes simplifiés et qui s'exécutent autant de fois que nécessaire.

# La structure while




## Objectifs

- Décrire la syntaxe de l'instruction répétitive while
- Créer des programmes contenant des boucles
- Créer des programmes qui s'exécute en boucle sans s'arrêter
- Créer des programmes qui s'exécute en boucle avec une condition d'arrêt

La structure while est utilisée pour faire des répétitions (boucles) sur une portion de code en fonction de la valeur de retour d'un prédicat. Cette partie introduit la notion de boucle et présente l'utilité de la structure while.


## 1. Forme de la structure while

 *Syntaxe : python 2 ou 3*

```
while condition :  
[indentation] bloc d'instruction
```

## 2. Boucle finie

Une structure while ("tant que" en Français) permet de *répéter un bloc d'instruction tant qu'une condition renvoie le booléen True*. Si la condition renvoie le booléen False, alors la répétition ou la boucle s'arrête et le bloc d'instruction n'est plus exécuté. On dit alors que la boucle est finie. chaque répétition d'un bloc d'instruction est appelée une *itération*.

 *Exemple : table de multiplication*

Ce programme a pour but de réaliser la table de multiplication d'un nombre donné

```
>>> nombre = 7  
>>> compteur = 0  
>>> while compteur <= 10:  
    print(nombre, "*", compteur, "=", nombre*compteur)  
    compteur = compteur + 1
```

*table de multiplication de la variable nombre avec while*

A la ligne 1, une variable est déclarée et initialisée en vue de réaliser sa table de multiplication, il est aussi possible d'utiliser la fonction de lecture *input* pour recueillir la valeur de cette variable.

Au niveau de la ligne 2, une variable compteur est initialisée à 0, elle va servir comme test d'arrêt pour la boucle while.

A la ligne 3, une boucle démarre en vérifiant la valeur renvoyé par le prédicat *compteur<=10* ; pour la première itération, *compteur vaut 0* alors le prédicat renvoie True car  $0 \leq 10$ .

La ligne 4 affiche la table de multiplication à l'écran et la ligne 5 augmente la valeur de la variable compteur de 1.

L'instruction de la ligne 3 est ré-exécutée avec la nouvelle valeur de la variable compteur qui vaut

maintenant 1. Les lignes 3, 4 et 5 sont ré-exécutée jusqu'à ce que le prédicat `compteur<=10` renvoie la valeur False ; c'est à dire que compteur prennent la valeur 11.

### 3. Boucle infinie

Une boucle infinie est une structure répétitive dont la condition renvoie toujours la valeur True.

☞ *Exemple : Python 2 ou 3*

---

cette instruction affiche de façon continue l'instruction "ca marche"

```
>>> while True:
    print("ca tourne")
```

*boucle infinie*

☞ *Remarque : Arrêter une boucle infinie*

---

Pour arrêter une boucle infinie dans un interpréteur interactif de python, il faudrait appuyer la combinaison de touche `ctrl+c`

### 4. Arrêt instantané d'une boucle : break

*l'instruction "break"*

Il est possible d'arrêter une boucle infinie sans appuyer la combinaison de touche `ctrl+c` grâce à l'instruction `break`. l'instruction `break` met fin immédiatement à une boucle lorsqu'elle est exécutée

☞ *Exemple : Python 2 ou 3*

---

```
>>> i=0
>>> while True:
    print('ca marche')
    i=i+1
    if i==6:
        print('stop')
        break
```

*instruction break dans une boucle infini*

Ce programme affiche "ca marche" a l'écran jusqu'à ce que la variable `i` soit égale à 6, ensuite la boucle infinie qui renvoie toujours la valeur True est stoppé par l'instruction `break`.

☞ *Remarque*

---

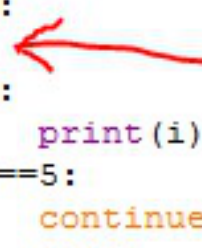
L'utilisation de l'instruction `break` en dehors d'une structure répétitive entraîne une erreur.

### 5. L'instruction continue

`continue` est une instruction qui à l'intérieur d'une structure répétitive termine l'itération courante et débute l'itération suivante.

### Exemple : python 2 ou 3

```
>>> i=0
>>> while i<10:
    i=i+1
    if i<5:
        print(i)
    elif i==5:
        continue
        i=i+4
    else:
        print(i)
```



*instruction continue dans une structure répétitive*

Ce code affichera les valeurs de 1 à 10 sauf la valeur 5. Lorsque  $i$  prend la valeur 5 le prédicat  $i==5$  renvoie la valeur `True`, alors le bloc d'instruction qui contient l'instruction `continue` est exécuté. Cependant l'instruction  $i=i+4$  n'est pas exécutée car `continue` termine l'itération courante et fait remonter l'exécution à la 1<sup>ère</sup> instruction de la boucle `while` qui est  $i=i+1$ , il débute ainsi l'itération pour  $i=6$ .

### Remarque

L'utilisation de l'instruction `continue` en dehors d'une structure répétitive entraîne une erreur.

## 6. Exercice

[Solution n°1 p 12]

ce code exécute l'instruction `print` combien de fois ?

```
>>> i=3
>>> while i>=0:
    print(i)
    i=i-1
```

## 7. Exercice

[Solution n°2 p 12]

Complétez ce code afin de créer une boucle qui n' affiche que les valeurs paires

```
>>>x=0
```

```
>>> x<=20 :
```

```
[indentation] (x)
```

```
[indentation] x = x+
```

## 8. Exercice

[Solution n°3 p 12]

Quelles instruction termine instantanément une boucle?

## 9. Exercice

[Solution n°4 p 12]

Que fait ce programme ?

```
>>> i=1
>>> while i==1:
        print("c'est parti")
```

- ☐ Il génère un message d'erreur
- ☐ Il crée une boucle finie sans condition d'arrêt
- ☐ Il crée une boucle infinie avec condition d'arrêt
- ☐ Il crée une boucle infinie



# La structure for



## Objectifs

- Décrire la syntaxe de l'instruction répétitive for
- Utiliser la structure for pour parcourir des séquences
- Créer des programmes simplifiés et moins complexe à lire

La structure for est utilisée pour parcourir des *séquences*. Une *séquence* est un ensemble fini d'élément indexé de p à n-1 où p est le 1er élément et n le n<sup>ème</sup> élément. Cette leçon présente la structure répétitive for et son utilisation sur les séquences de nombre entier.

## 1. la fonction range

*range*

la fonction *range* est une fonction qui crée une séquence contenant des éléments entier d'un intervalle donné en fonction des paramètres fournis à la fonction.

### Exemple

```
>>>range(10) -> contient tous les entiers de 0 inclus à 10 exclus ; i.e. [0, 10[ ou [0, 9]
>>>range(1, 11) -> contient tous les entiers de 1 inclus à 11 exclus ; i.e. [1, 11[ ou [1, 10]
>>>range(0, 10, 2) -> contient un ensemble d'entiers de 0 inclus à 10 exclus avec un pas de 2 ; i.e. {0, 2, 4, 6, 8}
```

## 2. Forme de la boucle for

### Syntaxe

```
>>>for element in sequence :
```

```
[indentation] bloc d'instruction
```

*element* est une variable que le programmeur appelle, mais la structure for se charge de la créer et de lui affecter successivement chaque élément de la séquence. Ainsi le programmeur n'a pas besoin de déclarer auparavant la variable *element*.

Le mot clé *in* permet de faire un *test d'appartenance* d'un élément à une séquence. Le test d'appartenance renvoie la valeur True si l'élément est dans la séquence ou il renvoie la valeur False si l'élément n'est pas dans la séquence. *Il peut être utilisé ailleurs que dans une structure for*

### Exemple

Reprenons l'exemple de la table de multiplication d'un nombre donné

```
>>> nombre = 7
>>> for compteur in range(11):
    print(nombre, "*", compteur, "=", nombre*compteur)
```

*table de multiplication avec la boucle for*

On remarque d'abord qu'il n'y a pas de déclaration et d'initialisation de la variable compteur. Ensuite, il n'y a pas d'*incrément* de la variable compteur ; c'est-à-dire l'instruction compteur=compteur+1 n'existe pas. Enfin le code avec la boucle for *est beaucoup plus court* que celui avec la boucle while.

### 3. Recommandation

#### Remarque : Imbrication

Il est possible de faire une imbrication de structures répétitives; i.e. qu'on peut écrire un while à l'intérieur d'un while, un for à l'intérieur d'un for, un for à l'intérieur d'un while et vice-versa.

#### Conseil

La boucle while est parfaitement adaptée pour : réaliser des boucles finies sur un bloc d'instruction en fonction d'un prédicat bien défini, réaliser des programmes qui se répètent jusqu'à ce que l'utilisateur décide d'y mettre fin, faire des programmes qui s'exécutent sans fin.

La boucle for est beaucoup plus adaptée pour parcourir des séquences d'éléments. Elle rend le code moins surchargé et beaucoup plus lisible. Parcourir une séquence avec une boucle while demande beaucoup de ligne de code.

### 4. Exercice

[Solution n°5 p 12]

*Laquelle de ces structures est utilisée pour réaliser des itérations sur une séquence en python*

- ☐ foreach
- ☐ fore
- ☐ forever
- ☐ for
- ☐ whole

### 5. Exercice

[Solution n°6 p 12]

*Complétez ce code afin de créer une boucle qui n'affiche que les valeurs impaires de la séquence.*

```
for i in range( , 20, ):
[indentation] print( )
```

# Conclusion



Cette leçon a présenté la notion de structure répétitive et leur utilité. L'apprenant pourra s'aider de ces structures pour factoriser son code, créer des boucles ou parcourir des séquences d'éléments. La prochaine leçon s'intéressera à la factorisation de code avec les fonctions.

# Solutions des exercices

## > Solution n° 1

*Exercice p. 7*`4`

## > Solution n° 2

*Exercice p. 7*

```
>>>x=0
>>> while x<=20 :
[indentation] print (x)
[indentation] x = x+ 2
```

## > Solution n° 3

*Exercice p. 7*`break`

## > Solution n°4

*Exercice p. 8*

- ☐ Il génère un message d'erreur
- ☐ Il crée une boucle finie sans condition d'arrêt
- ☐ Il crée une boucle infinie avec condition d'arrêt
- ☒ Il crée une boucle infinie

## > Solution n°5

*Exercice p. 10*

- ☐ foreach
- ☐ fore
- ☐ forever
- ☒ for
- ☐ whole

## > Solution n° 6

*Exercice p. 10*

```
for i in range( 1 , 20, 2 ) :
[indentation] print( i )
```