

## La boucle non bornée: while

La boucle *while* est plus puissante que la boucle *for*, elle permet de créer des répétitions, même si l'on ne connaît pas à l'avance le nombre de répétitions.

Contrairement à la boucle *for*, le nombre d'itérations n'est pas fixé à l'avance, mais il est testé dynamiquement au début de chaque boucle: On parle de boucle non bornée.

À chaque tour, on teste une condition, si cette condition est vérifiée, on exécute le bloc d'instructions de la boucle, sinon, l'exécution de la boucle est arrêtée et le programme se poursuit «normalement».

L'écriture d'une boucle non bornée nécessite trois étapes:

- Initialisation de la variable de contrôle.

- Condition d'arrêt.

- Mise à jour de la variable de contrôle.

### Réécriture d'une boucle for

Nous allons traduire une boucle *for* par un *while* (**même si cela n'est pas conseillé**).

```
for i in range(1, 7, 2):  
    print(i)
```

```
1  
3  
5
```

Dans une boucle *for* toutes les affectations de la variable *i* sont gérées par la fonction *range()*. Avec le *while* c'est à nous de nous en occuper.

```
i = 1                # initialisation  
while i < 7:         # condition d'arrêt  
    print(i)  
    i = i + 2        # Mise à jour de la variable de contrôle
```

```
1  
3  
5
```

## Contrôle par variable booléenne

Nous créons ici une variable booléenne `repeat` pour bien comprendre si la boucle *while* doit se poursuivre ou non.

```
repeat = True;
while repeat:
    resultat = int(input("Combien font 2 + 5? "))
    if resultat == 7:
        print("Bravo vous avez trouvé.")
        repeat = False
```

Il serait cependant plus élégant de faire passer la condition du *if* dans le *while*.

```
resultat = None
while resultat != 7:
    resultat = int(input("Combien font 2 + 5? "))
print("Bravo vous avez trouvé.")
```

## Remarques

- La variable compteur est souvent notée `i`;
- Faire **attention à ne pas programmer une boucle infinie** qui bloquerait votre programme en lui faisant répéter l'instruction une infinité de fois et saturerait ainsi votre processeur et votre mémoire.

```
i = 0
while i >= 0:
    print(i)
    i = i + 1
    # i sera toujours positif, la condition d'arrêt ne sera jamais
    vraie
```

### Exercice 1 :

Écrivez un programme qui fait jouer l'utilisateur au ni oui, ni non : il rentre un texte jusqu'à saisir "oui" ou "non", ce qui déclenche la fin du jeu.

### Exercice 2 :

a) Écrivez un programme qui fait saisir un nombre à l'utilisateur jusqu'à ce que ce nombre soit inférieur ou égal à 100.

b) Ensuite, améliorez votre programme pour que le nombre saisi soit compris entre 50 et 100.

*Attention Réfléchissez bien à la condition de votre boucle : quel est l'inverse de "compris entre 50 et 100" ?*

### Exercice 3 :

Écrire un programme qui trouve le plus petit entier  $n$  tel que  $n^3 > 10000$ .

### Exercice 4 :

On dispose d'une feuille de papier d'épaisseur 0,1 mm. Combien de fois doit-on la plier au minimum pour que l'épaisseur dépasse la hauteur de la tour Eiffel 324 m. Écrire un programme en Python pour résoudre ce problème.

### Exercice 5 : Comprendre la différence entre for et while!

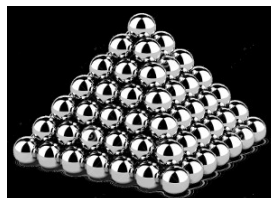
Aujourd'hui un appartement vaut 100 000€. Sa valeur augmente de 1% chaque année.

a) Écrire un programme en Python pour connaître sa valeur au bout de 10ans.

b) Écrire un programme en Python pour savoir au bout de combien d'années sa valeur aura doublé.

### Exercice 6 :

Inès veut construire une pyramide à base carrée comme sur la photo. La pyramide sur la photo a 7 étages.



Inès a 1000 billes. Combien d'étages au maximum aura sa pyramide? Écrire un programme en Python pour répondre au problème.