Python: Les boucles while

Les boucles `while` sont une des structures de contrôle de Python. Elles permettent d'exécuter un bloc de code **tant qu'une condition spécifiée est vraie**. Dans cet article, nous explorerons en détail la syntaxe des boucles `while` en Python, leur fonctionnement, leurs avantages et leurs inconvénients, ainsi que des exemples pratiques pour illustrer leur utilisation.

5TTR 6TTR 3TTR Exploration

Introduction aux boucles `while`

Les boucles `while` sont utilisées lorsque vous souhaitez répéter l'exécution d'un bloc de code **aussi longtemps qu'une condition spécifiée reste vraie**. Autrement dit, **les boucles `while` permettent d'exécuter un bloc de code tant qu'une condition spécifiée est vraie**.

> Les boucles `for`, quant à elles, sont utilisées pour parcourir des listes d'éléments (comme une liste, une chaîne de caractères ou une plage de nombres) et répéter l'exécution d'un bloc de code un certain nombre de fois.

> 💡 **1 ITÉRATION** = **1** passage dans la boucle.

> 💡 En Python on utilisera de préférence:
> - les boucles ´for´ quand on connait le nombre d'itérations à l'avance
> - les boucles `while` si on ne connait pas le nombre d'itérations à l'avance

La syntaxe générale d'une boucle `while` en Python est la suivante :

```
1 while condition:
2     # Bloc de code à exécuter tant que la condition est vraie
```

Dans cette structure, le bloc de code indenté sous la déclaration `while` est répété tant que la condition spécifiée reste **vraie** (`True`). La condition est évaluée **avant** chaque itération de la boucle. Si la condition devient fausse à un moment donné, l'exécution de la boucle `while` se termine à la fin du bloc de coade et le programme passe à l'instruction suivante après la boucle.

Exemples d'utilisation des boucles `while`

Voyons maintenant quelques exemples pratiques pour illustrer l'utilisation des boucles `while` en Python :

Exemple 1 : Comptage jusqu'à un certain nombre

```python
# Comptage jusqu'à 5
count = 1
while count <= 5:
    print(count)
    count += 1
```

Dans cet exemple, la boucle `while` est utilisée pour compter jusqu'à 5 en affichant chaque nombre à l'écran. La variable `count` est initialisée à 1, puis la boucle continue d'itérer tant que `count` est inférieur ou égal à 5. À chaque itération, la valeur de `count` est augmentée de 1 à l'aide de l'opérateur `+=`.

> L'erreur la plus courante avec une boucle `while` consiste à oublier de mettre à jour la variable qui est testée dans la condition d'entrée. Dans le cas d'une boucle comme celle ci-dessus qui permet de compter jusqu'à un certain nombre, il est fréquent d'oublier d'incrémenter la variable (ici: `count`). Résultat? Une **boucle infinie**, c'est-à-dire qui ne se termine jamais...

Exemple 2 : Demande de saisie utilisateur

```python
# Demande de saisie utilisateur jusqu'à ce qu'un mot spécifique soit entré
password = ""
while password != "motdepasse":
    password = input("Entrez le mot de passe : ")
print("Accès autorisé !")
```

Dans cet exemple, la boucle `while` est utilisée pour demander à l'utilisateur de saisir un mot de passe jusqu'à ce qu'il entre le mot de passe correct, qui est "motdepasse" dans cet exemple. La boucle continue de demander à l'utilisateur de saisir le mot de passe tant que la valeur entrée ne correspond pas au mot de passe attendu.

Avantages et inconvénients des boucles `while`

Les boucles `while` offrent plusieurs avantages, mais elles peuvent également présenter certains inconvénients :

Avantages :

- Flexibilité : Les boucles `while` permettent une **répétition basée sur une condition**, ce qui les rend flexibles et polyvalentes.

- Convivialité : Elles sont souvent utilisées lorsque **le nombre d'itérations n'est pas connu** à l'avance ou lorsqu'une condition spécifique doit être vérifiée avant chaque itération.

Inconvénients :

- **Risque de boucle infinie** : Si la condition spécifiée dans une boucle `while` reste toujours vraie, la boucle continuera à s'exécuter indéfiniment, ce qui peut entraîner un gel de l'exécution du programme ou un épuisement des ressources système.
- Complexité potentielle : Les boucles `while` peuvent rendre le code plus difficile à comprendre si la condition et les étapes d'itération ne sont pas clairement définies.

Différence entre `while` et `for`

Les boucles `while` et `for` sont deux structures de contrôle couramment utilisées en programmation pour répéter une série d'instructions.

La boucle `while` exécute un bloc de code **tant qu'une condition donnée est vraie**. Elle est particulièrement utile lorsque **le nombre d'itérations n'est pas connu à l'avance** et dépend d'une condition qui peut changer à chaque itération. Par exemple, une boucle `while` est idéale pour lire les entrées utilisateur jusqu'à ce qu'une entrée valide soit fournie.

En revanche, la boucle `for` est utilisée pour **itérer sur une séquence** (comme une **liste**, un tuple ou une **chaîne de caractères**) ou sur une **plage de valeurs prédéfinie** avec `range()`. Elle est particulièrement adaptée lorsque **le nombre d'itérations est connu à l'avance** ou lorsque l'on doit **parcourir chaque élément d'une collection**. Par exemple, une boucle `for` est idéale pour parcourir tous les éléments d'une liste et effectuer une opération sur chacun d'eux.

> En résumé, utilisez une boucle `while` lorsque vous ne savez pas combien de fois vous devez itérer et une boucle `for` lorsque le nombre d'itérations est déterminé ou que vous itérez sur une collection.

Conclusion

Dans cet article, nous avons exploré les boucles `while` en Python, leur syntaxe, leur fonctionnement, ainsi que leurs avantages et leurs inconvénients. Les boucles `while` sont des outils puissants pour contrôler le flux d'exécution dans un programme et sont largement utilisées dans une variété de situations.

En comprenant comment utiliser efficacement les boucles `while`, vous pouvez écrire du code plus lisible, plus maintenable et plus robuste dans vos projets Python. N'oubliez pas de pratiquer et d'expérimenter avec des exemples concrets pour maîtriser pleinement les concepts discutés dans cet article.