Les boucles `for` en Python : Guide complet

Les boucles `for` en Python sont utilisées pour itérer sur une séquence (comme une liste, un tuple, une chaîne de caractères, un dictionnaire ou même une plage de nombres). Elles permettent d'exécuter un bloc de code plusieurs fois, de manière simple et efficace.

1. Syntaxe de base

```
for élément in séquence:
    # Bloc de code à exécuter
```

- `élément` : représente chaque élément de la séquence à chaque itération.
- `séquence` : peut être une liste, une chaîne, un tuple, un dictionnaire, etc.

2. Exemples simples

a. Parcourir une liste

```
fruits = ["pomme", "banane", "cerise"]
for fruit in fruits:
    print(fruit)
```

Sortie :

```
pomme
banane
cerise
```

b. Parcourir une chaîne de caractères

```
mot = "Python"
for lettre in mot:
    print(lettre)
```

Sortie :

```
P
y
```

```
t
h
o
n
```

c. Utiliser `range()` pour générer une séquence de nombres

```python
for i in range(5):  # Génère les nombres de 0 à 4
    print(i)
```

Sortie :

```
 0
1
2
3
4
```

3. Utilisation avancée

a. Boucle avec un intervalle personnalisé

```python
for i in range(2, 10, 2):  # De 2 à 10 (exclu), avec un pas de 2
    print(i)
```

Sortie :

```
 2
4
6
8
```

b. Parcourir un dictionnaire

```python
étudiants = {"Alice": 18, "Bob": 20, "Claire": 19}
for nom, âge in étudiants.items():
    print(f"{nom} a {âge} ans")
```

Sortie :

```
 Alice a 18 ans
Bob a 20 ans
Claire a 19 ans
```

c. Boucle avec une condition

```
nombres = [1, 2, 3, 4, 5]
for nombre in nombres:
    if nombre % 2 == 0:
        print(f"{nombre} est pair")
```

Sortie :

```
 2 est pair
4 est pair
```

4. Instructions utiles dans une boucle `for`

a. `break` : arrêter la boucle

```
for i in range(10):
    if i == 5:
        break
    print(i)
```

Sortie :

```
 0
1
2
3
4
```

b. `continue` : passer à l'itération suivante

```
for i in range(5):
    if i == 2:
```

```
        continue
    print(i)
```

Sortie :

```
0
1
3
4
```

c. `else` avec une boucle `for`

L'instruction `else` est exécutée si la boucle se termine normalement (sans `break`).

```
for i in range(5):
    print(i)
else:
    print("Boucle terminée")
```

Sortie :

```
0
1
2
3
4
Boucle terminée
```

5. Cas particuliers

a. Boucle sur une liste vide

Si la séquence est vide, le bloc de code ne sera pas exécuté.

```
for élément in []:
    print("Cela ne s'affichera pas")
```

b. Boucle imbriquée

Vous pouvez imbriquer des boucles pour parcourir des structures complexes.

```
matrice = [[1, 2], [3, 4], [5, 6]]
for ligne in matrice:
    for élément in ligne:
        print(élément)
```

Sortie :

```
1
2
3
4
5
6
```

6. Avantages des boucles `for`

- Simples à utiliser pour parcourir des séquences.
- Compatibles avec de nombreux types de données.
- Puissantes lorsqu'elles sont combinées avec des fonctions comme `range()`, `enumerate()`, ou des compréh