Chapitre 3 : Les structures de contrôle en Python

Par Robert DIASSÉ

Dans ce chapitre, nous allons explorer les structures de contrôle essentielles pour la programmation en Python. Ces outils permettent de contrôler l'exécution du programme en fonction de conditions ou en répétant certaines actions.

Plan du chapitre

---

1. Introduction aux structures de contrôle

Les structures de contrôle permettent de :

- **Exécuter une ou plusieurs instructions** seulement si une condition est vraie.
- **Répéter une série d'instructions** tant qu'une condition est remplie.

---

2. Les conditions (`if`, `elif`, `else`)

2.1. Structure de base

La structure conditionnelle permet d'exécuter un bloc d'instructions selon le résultat d'une expression booléenne (vrai ou faux).

Syntaxe :

```
if condition:
    # Instructions si la condition est vraie
elif autre_condition:
    # Instructions si l'autre condition est vraie
else:
    # Instructions si aucune condition n'est vraie
```

Exemple :

```
age = int(input("Entrez votre âge : "))

if age < 18:
    print("Vous êtes mineur.")
elif age >= 18 and age < 65:
    print("Vous êtes adulte.")
else:
    print("Vous êtes retraité.")
```

2.2. Opérateurs de comparaison

Opérateur	Description	Exemple
`==`	Égal à	`x == y`
`!=`	Différent de	`x != y`
`<`	Inférieur à	`x < y`
`>`	Supérieur à	`x > y`
`<=`	Inférieur ou égal à	`x <= y`
`>=`	Supérieur ou égal à	`x >= y`

2.3. Opérateurs logiques

Opérateur	Description	Exemple
`and`	Les deux conditions sont vraies	`x > 0 and x < 10`
`or`	Une des deux conditions est vraie	`x < 0 or x > 10`
`not`	Inverse la condition	`not(x > 0)`

---

3. Les boucles

3.1. La boucle `for`

Une boucle `for` permet d'itérer sur une séquence (liste, chaîne de caractères, range, etc.).

Syntaxe :

```
for element in sequence:
    # Instructions
```

Exemple :

```
for i in range(5):  # Itère de 0 à 4
    print("Itération :", i)
```

3.2. La boucle `while`

Une boucle `while` exécute un bloc tant qu'une condition est vraie.

Syntaxe :

```
while condition:
    # Instructions
```

Exemple :

```
compteur = 0

while compteur < 5:
    print("Compteur :", compteur)
    compteur += 1
```

3.3. Les mots-clés `break` et `continue`

- `break` : Permet de **sortir immédiatement** d'une boucle.
- `continue` : Permet de **passer directement à l'itération suivante**.

Exemple :

```
for i in range(10):
    if i == 5:
        break  # Quitte la boucle si i vaut 5
    if i % 2 == 0:
```

```
        continue  # Saute les nombres pairs
    print(i)
```

3.4. La clause `else` avec les boucles

Une clause `else` peut être utilisée avec une boucle pour exécuter un bloc lorsque la boucle se termine **normalement** (sans `break`).

Exemple :

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

---

4. Les compréhensions de listes

Les compréhensions de listes permettent de créer des listes en une seule ligne, avec une syntaxe compacte.

Exemple :

```
nombres = [x for x in range(10) if x % 2 == 0]
print(nombres)  # [0, 2, 4, 6, 8]
```

---

5. Instructions conditionnelles sur une ligne (opérateur ternaire)

L'opérateur ternaire permet d'écrire une condition simple sur une seule ligne.

Syntaxe :

```
variable = valeur_si_vrai if condition else valeur_si_faux
```

Exemple :

```
age = int(input("Entrez votre âge : "))
categorie = "mineur" if age < 18 else "majeur"
print("Vous êtes", categorie)
```

---

6. Exercices pratiques

Exercice 1 : Nombre pair ou impair

- Demandez à l'utilisateur d'entrer un nombre.
- Indiquez s'il est pair ou impair.

Exercice 2 : Boucle avec `while`

- Écrivez un programme qui continue à demander un mot de passe jusqu'à ce qu'il soit correct (par exemple, "admin123").

Exercice 3 : Table de multiplication

- Affichez la table de multiplication d'un nombre donné par l'utilisateur (de 1 à 10).

Exercice 4 : Trouver un nombre dans une liste

- Créez une liste de nombres.
- Demandez à l'utilisateur d'entrer un nombre et affichez s'il est présent dans la liste.

Exercice 5 : Compréhensions de listes

- Créez une liste contenant les carrés des nombres de 1 à 10.

Résumé

Ce chapitre fournit une base solide pour maîtriser les structures de contrôle en Python. Vous devrais maintenant être capables de :

1. Prendre des décisions conditionnelles avec `if`, `elif`, `else`.
2. Répéter des actions avec `for` et `while`.
3. Utiliser efficacement `break`, `continue` et la clause `else`.
4. Créer des listes dynamiques avec les compréhensions.