UNIVERSITÉ Larbi Ben Mhidi

Faculté des Sciences de la Terre, de la Géographie et de l'Aménagement du Territoire

Département Géographie et aménagement du territoire

Matière : Programmation

Niveau : 1ère année Licence— Semestre 2

Crédits : 2 | **Coefficient :** 1

Enseignant : Bezzaz Soumia

Année universitaire : 2025–2026

Structures de Contrôle : Conditions et Boucles

PARTIE I — LES STRUCTURES CONDITIONNELLES

1.1. Introduction

Dans un programme, il est souvent nécessaire d'exécuter certaines instructions uniquement si une condition est satisfaite. Les **structures conditionnelles** (ou structures de contrôle de flux) permettent de faire des choix et d'orienter l'exécution du programme en fonction de conditions logiques.

Définition formelle :

Une structure conditionnelle est un mécanisme de programmation qui permet d'exécuter un bloc d'instructions si et seulement si une expression booléenne (condition) est évaluée à **True**.

Exemple concret:

Un sismologue souhaite classifier un séisme en fonction de sa magnitude :

• Si magnitude $< 4.0 \rightarrow$ séisme faible
• Si $4.0 \leq$ magnitude $< 6.0 \rightarrow$ séisme modéré
• Si magnitude $\geq 6.0 \rightarrow$ séisme majeur

1.2. La structure if (si)

Syntaxe générale :

if condition :

```
    # Bloc d'instructions exécuté si la condition est vraie
    instruction1
    instruction2
```

Exemple 1 :

```
magnitude = 6.5

if magnitude >= 6.0:
    print("Séisme majeur détecté !")
```

Sortie :

```
Séisme majeur détecté !
```

1.3. La structure if...else (si...sinon)

Syntaxe générale :

```
if condition :
    # Exécuté si condition = True
    instructions_si_vrai
else:
    # Exécuté si condition = False
    instructions_si_faux
```

Exemple 2 : Détection de pluie

```
precipitation = 15  # mm de pluie

if precipitation > 0:
    print("Il pleut.")
else:
    print("Temps sec.")
```

Sortie :

```
Il pleut.
```

1.4. La structure if...elif...else

Lorsqu'on a plusieurs conditions à tester, on utilise **elif** (contraction de else if).

Syntaxe générale :

```
if condition1 :
    bloc1
elif condition2 :
```

```
    bloc2
elif condition3 :
    bloc3
else:
    bloc_par_defaut
```

Exemple 3 : Classification sismique

```
magnitude = 5.2


if magnitude < 4.0:
    print("Séisme faible")
elif magnitude < 6.0:
    print("Séisme modéré")
else:
    print("Séisme majeur")
```

Sortie :

Séisme modéré

PARTIE II — LES BOUCLES (LOOPS)

2.1. Introduction

Une boucle est une structure de contrôle qui permet de répéter l'exécution d'un bloc d'instructions tant qu'une condition est vérifiée ou pour chaque élément d'une séquence.

Pourquoi utiliser des boucles ?

• Automatiser des tâches répétitives
• Parcourir des listes de données
• Effectuer des calculs itératifs

Python propose deux types de boucles principaux :

1. **La boucle while** (tant que) : répète tant qu'une condition est vraie
2. **La boucle for** (pour) : parcourt une séquence d'éléments

2.2. La boucle while (tant que)

Syntaxe générale :

```
while condition :
    # Bloc répété tant que condition = True
```

```
    instructions
```

Attention : Il faut toujours s'assurer que la condition finira par devenir **False**, sinon la boucle sera infinie !

Exemple 5 : Compteur simple

```
i = 1
while i <= 5:
    print(f"Itération {i}")
    i += 1  # Équivalent à i = i + 1
```

Sortie :

```
Itération 1
Itération 2
Itération 3
Itération 4
Itération 5
```

Exemple 6 :

```
# Simulation de forage : profondeur croissante
profondeur = 0  # mètres
profondeur_cible = 50  # mètres


while profondeur < profondeur_cible:
    profondeur += 10
    print(f"Forage à {profondeur} m")
print("Profondeur cible atteinte !")
```

2.3. La boucle for (pour)

La boucle **for** permet de parcourir une séquence (liste, chaîne, intervalle de nombres, etc.) élément par élément.

Syntaxe générale :

```
for element in sequence :
    # Bloc exécuté pour chaque élément
    instructions
```

Exemple 7 : Parcourir une liste de minéraux

```
mineraux = ["quartz", "calcite", "feldspath", "mica"]


for mineral in mineraux:
```

```
    print(f"Minéral détecté : {mineral}")
```

Sortie :

```
Minéral détecté : quartz
Minéral détecté : calcite
Minéral détecté : feldspath
Minéral détecté : mica
```

2.4. Les mots-clés break et continue

break : Interrompt la boucle immédiatement

continue : Passe à l'itération suivante sans exécuter le reste du bloc

Exemple 10 : Utilisation de break

```
# Recherche d'un séisme majeur
magnitudes = [3.5, 4.2, 6.8, 5.0, 7.2]

for mag in magnitudes:
    if mag >= 6.0:
        print(f"Alerte ! Séisme majeur de magnitude {mag}")
        break  # Sort de la boucle dès qu'un séisme majeur est trouvé
    print(f"Magnitude {mag} - pas d'alerte")
```

Sortie :

```
Magnitude 3.5 - pas d'alerte
Magnitude 4.2 - pas d'alerte
Alerte ! Séisme majeur de magnitude 6.8
```

Exemple 11 : Utilisation de continue

```
# Ignorer les valeurs négatives (données erronées)
temperatures = [15.2, -999, 18.5, 20.1, -999, 22.0]

for temp in temperatures:
    if temp < 0:
        continue  # Passe à la température suivante
    print(f"Température valide : {temp}°C")
```

Sortie :

```
Température valide : 15.2°C
Température valide : 18.5°C
Température valide : 20.1°C
Température valide : 22.0°C
```

PARTIE III — EXERCICES

Exercice 1 — Classification de température

Écrire un programme qui demande une température en degrés Celsius et affiche :

• « Très froid » si $T < 0$
• « Froid » si $0 \leq T < 15$
• « Modéré » si $15 \leq T < 25$
• « Chaud » si $T \geq 25$

Solution

```
temp = float(input("Température (°C) : "))

if temp < 0:
    print("Très froid")
elif temp < 15:
    print("Froid")
elif temp < 25:
    print("Modéré")
else:
    print("Chaud")
```

Exercice 2 — Calcul de la somme des 10 premiers entiers

Écrire un programme qui calcule la somme : 1 + 2 + 3 + ... + 10 en utilisant une boucle **for**.

Solution

```
somme = 0

for i in range(1, 11):
    somme += i

print(f"La somme est : {somme}")  # Résultat : 55
```

Exercice 3 — Ondes sismiques

Un géophysicien mesure les temps d'arrivée des ondes sismiques à différentes stations. Écrire un programme qui :

1. Crée une liste de temps d'arrivée (en secondes) : [12.5, 15.3, 18.7, 22.1, 25.8]

2. Parcourt cette liste et affiche chaque temps
3. Si un temps dépasse 20 secondes, affiche une alerte

Solution

```
temps_arrivee = [12.5, 15.3, 18.7, 22.1, 25.8]
for temps in temps_arrivee:
    print(f"Temps d'arrivée : {temps} s")
    if temps > 20:
        print("Alerte : Temps inhabituellement élevé !")
```