PRÉVISION SAISONNIÈRE

MODULE 4 – STRUCTURES DE CONTRÔLE EN PYTHON

Mandela HOUNGNIBO **Arsène KIEMA**

mandela.houngnibo@cilss.int / arsene.kiema@cilss.int

2025-06-21

Les conditions : if, elif, else

Les conditions permettent d'exécuter certaines instructions seulement si une condition est vraie

- `if` vérifie une première condition
- `elif`signifie "sinon si", pour une autre condition
- `else` est exécuté si aucune condition précédente n'est vraie

Les conditions : if, elif, else

Syntaxe générale

```
if condition1:
    # Bloc exécuté si condition1 est vraie
elif condition2:
    # Bloc exécuté si condition2 est vraie
else:
    # Bloc exécuté si aucune condition ci-dessus n'est vraie
```

NB : Les blocs d'instructions sont indentés (en général 4 espaces) pour indiquer qu'ils dépendent de la condition.

Les conditions : if, elif, else

Comment ça fonctionne ?

1. `if` : Python vérifie d'abord si la condition après if est vraie. Si oui, il exécute ce bloc et ignore le reste.
2. `elif (else if)` : Si la première condition est fausse, Python vérifie cette condition.
3. `else` : Si aucune condition n'est remplie, ce bloc est exécuté.

Les conditions : if, elif, else

Conditions simple

```
age = 20

if age < 18:
    print("Mineur")
elif age == 18:
    print("Tout juste majeur")
else:
    print("Majeur")
```

```
Majeur
```

Les conditions : if, elif, else

Conditions combinées

```
precip_mm = 2
temp_max = 42

if precip_mm < 5 and temp_max > 40:
    print("  Alerte sécheresse extrême")
else:
    print("Conditions normales")
```

```
   Alerte sécheresse extrême
```

Les boucles

Les boucles permettent de répéter un bloc d'instructions plusieurs fois, soit un nombre connu de fois (`for`), soit tant qu'une condition est vraie (`while`).

La boucle for

Utilisée pour répéter une action un nombre défini de fois, ou parcourir une séquence (liste, chaîne, etc.).

```
# Syntaxe générale
for variable in séquence:
    # bloc de code à répéter
```

Les boucles

La boucle `for`

```
for i in range(5):
  print("Iteration : " + str(i))
```

```
Iteration : 0
Iteration : 1
Iteration : 2
Iteration : 3
Iteration : 4
```

Les boucles

La boucle `while`

Répéter un bloc tant qu'une condition est vraie. Très utile lorsqu'on ne connaît pas à l'avance le nombre d'itérations.

```
# Syntaxe générale
while condition :
    # bloc à exécuter tant que la condition est vraie
```

Les boucles

La boucle `while`

```
compteur = 1

while compteur <= 3:
    print("Compteur :", compteur)
    compteur += 1
```

```
Compteur : 1
Compteur : 2
Compteur : 3
```

Contrôle de boucle : break et continue

break : Interrompt la boucle

Utilisé lorsqu'on souhaite sortir d'une boucle avant sa fin naturelle, par exemple si une condition critique est rencontrée.

```
stations = ["Niamey", "bamako", "Ouagadougou", "cotonou","lome"]

for station in stations:
    if station == "Ouagadougou":
        print("Station Ouagadougou. Arrêt du traitement.")
        break
    print(f"Traitement des données pour la station : {station}")
```

```
Traitement des données pour la station : Niamey
Traitement des données pour la station : bamako
```

Contrôle de boucle : break et continue

Utilisé pour ignorer certaines conditions sans interrompre toute la boucle.

continue : Saute à l'itération suivante

```
pluies = [0, 10, 0, 5, 0, 12]  # précipitations en mm

for jour, val in enumerate(pluies, start=1):
    if val == 0:
        continue  # ignore les jours sans pluie
    print(f"  Jour {jour} : {val} mm de pluie")
```

```
    Jour 2 : 10 mm de pluie
    Jour 4 : 5 mm de pluie
    Jour 6 : 12 mm de pluie
```

Les Fonctions en Python

Une fonction est un bloc de code qui exécute une tâche spécifique. Elle permet de réutiliser du code, de le structurer, et de le rendre plus lisible.

```
def nom_de_la_fonction(parametre1, parametre2, ...):
    """
    (Optionnel) Description de la fonction : ce qu'elle fait,
    les paramètres attendus, et la valeur retournée.
    """
    # Bloc d'instructions (indenté)
    # Traitement éventuel
    return valeur_ou_resultat  # (optionnel)
```

Les Fonctions en Python

Fonction sans paramètre ni return

```
def presentation():
    print("Je m'appelle GUIDO VAN ROSSUM et j'apprends Python.")

presentation()
```

```
Je m'appelle GUIDO VAN ROSSUM et j'apprends Python.
```

Les Fonctions en Python

Fonction avec paramètre et return

```
def multiplier(a, b):
    produit = a * b
    return produit

résultat = multiplier(4, 5)
print("Le produit est :", résultat)
```

```
Le produit est : 20
```

Les Fonctions en Python

Fonction avec paramètres et documentation

```
def calcul_moyenne(note1, note2, note3):
    """
    Calcule la moyenne de trois notes.
    Paramètres : note1, note2, note3 (float ou int)
    Retour : moyenne (float)
    """
    moyenne = (note1 + note2 + note3) / 3
    return moyenne

print(calcul_moyenne(12, 15, 17))
```

```
14.666666666666666
```

Opérateurs arithmétiques

```
# Données climatiques d'une journée
tmax = 40.0          # température maximale en °C
tmin = 28.0          # température minimale en °C
pluie = 12.5         # précipitations en mm
vent = 4.0           # vitesse du vent (arbitraire)

# Calculs avec opérateurs
tmean = (tmax + tmin) / 2            # moyenne journalière
amplitude = tmax - tmin              # écart de température
cumul_3jours = pluie * 3             # cumul sur 3 jours simulé
force_vent = vent ** 2               # force du vent au carré
```

Opérateurs arithmétiques

```
#Résultats du calcul
```

```
Température moyenne : 34.0 °C

Amplitude thermique : 12.0 °C

Cumul sur 3 jours estimé : 37.5 mm

Force du vent (exposant 2) : 16.0
```

Opérateurs de comparaison

Les opérateurs de comparaison permettent de **comparer deux valeurs** et renvoient un résultat **booléen** : `True` ou `False`.

Opérateur	Signification	Exemple	Résultat
==	Égal à	`22 == 30`	False
!=	Différent de	`45 != 30`	True
<	Strictement inférieur	`30 < 30`	False
>	Strictement supérieur	`52 > 40`	True
<=	Inférieur ou égal	`28 <= 28`	True
>=	Supérieur ou égal	`5**2 >= 40`	False

Opérateurs de comparaison

```
# Données
ville = "Kaya"
tmax = 41.3
tmin = 29.5
tmean = (tmax + tmin) / 2
pluie = 0.0
```

```
# Comparaison
print(tmax > 40)
```

```
True
```

```
print(tmean == 35)
```

```
False
```

Opérateurs logiques : and, or, not

Ces opérateurs permettent de combiner des conditions multiples

Opérateur	Signification	Exemple
`and`	Vrai si **toutes les conditions** sont vraies	`tmean > 35 and pluie == 0`
`or`	Vrai si **au moins une** condition est vraie	`tmax > 40 or pluie > 50`
`not`	Inverse la condition	`not pluie == 0` $\rightarrow$ pluie différente de 0

THANK YOU FOR YOUR ATTENTATION

AGRHYMET - CCR AOS