Les conditions

Une condition permet à un programme de prendre des décisions en fonction des valeurs des variables.

Les structures conditionnelles exécutent du code seulement si une condition donnée est vraie.

La Condition if

```
port = 22
if port == 22:
    print("Le port 22 est ouvert : SSH activé.")
```

Condition else

Utilisée pour spécifier un bloc de code à exécuter si la condition if est fausse.

```
protocole_connexion = "HTTP"

if protocole_connexion == "HTTPS":
    print("Connexion sécurisée via HTTPS.")
else:
    print("Alerte : Connexion non sécurisée via HTTP. Veuillez passer à HTTPS.")
```

Condition elif

Utilisée pour tester plusieurs conditions successives

```
tentatives_connexion = 4
tentatives_max = 3

if tentatives_connexion == 0:
    print("Aucune tentative de connexion.")
elif tentatives_connexion <= tentatives_max:
    print("Connexion autorisée.")
else:
    print("Alerte : Trop de tentatives de connexion, compte bloqué.")
```

Opérateurs Logiques

- **and** : Toutes les conditions doivent être vraies.
- **or** : Au moins une des conditions doit être vraie.
- **not** : Inverse la condition.

```python
pare_feu = True
antivirus = True

if pare_feu and antivirus:
    print("Le serveur est sécurisé.")
else:
    print("Le serveur est vulnérable.")
```

match-case

match-case est une structure introduite dans Python 3.10 qui permet de simuler le comportement du **switch** présent dans d'autres langages.

Elle est particulièrement utile pour tester une variable contre plusieurs valeurs, comme dans les contrôles de réseaux ou de protocoles.

```
protocol = "HTTP"

match protocol:
    case "HTTP":
        print("Connexion non sécurisée : HTTP détecté.")
    case "HTTPS":
        print("Connexion sécurisée : HTTPS détecté.")
    case "FTP":
        print("Connexion FTP détectée.")
    case _:
        print("Protocole inconnu.")
```

Conditions imbriquées

Parfois, il est nécessaire d'avoir des conditions à l'intérieur d'autres conditions.

```python
age = 20
citoyen = True

if age >= 18:
    if citoyen:
        print("Vous pouvez voter.")
    else:
        print("Vous devez être citoyen pour voter.")
else:
    print("Vous êtes trop jeune pour voter.")
```

Opérateurs ternaires

Python permet d'écrire des conditions simples en une seule ligne avec une **expression ternaire.**

```
age = 20
statut = "majeur" if age >= 18 else "mineur"
print(statut)  # Affiche "majeur"
```

Gestion des erreurs dans les conditions

Lorsqu'on utilise des conditions, il peut être utile d'ajouter des blocs try-except pour gérer les erreurs potentielles (comme les erreurs de type ou d'index)

```
try:
    age = int(input("Entrez votre âge : "))
    if age >= 18:
        print("Vous êtes majeur.")
    else:
        print("Vous êtes mineur.")
except:
    print("Erreur : Veuillez entrer un nombre valide.")
```

Exemple d’utilisation des conditions

```python
utilisateur = "John Doe"
role = "admin"  # Peut être "admin", "moderateur", ou "utilisateur"
mot_de_passe = "12345"
mot_de_passe_saisi = input("Entrez le mot de passe : ")

# Vérification du mot de passe
if mot_de_passe_saisi == mot_de_passe:
    # Vérification du rôle utilisateur
    if role == "admin":
        print(f"Bienvenue {utilisateur}, vous avez un accès complet en tant qu'administrateur.")
    elif role == "moderateur":
        print(f"Bienvenue {utilisateur}, vous avez un accès limité en tant que modérateur.")
    else:
        print(f"Bienvenue {utilisateur}, vous avez un accès restreint en tant qu'utilisateur.")
else:
    print("Erreur : Mot de passe incorrect.")
```