# Les structures conditionnelles


**Résumé**

Un programme qui fait toujours la même chose n'est pas très utile. Pour qu'il puisse réagir différemment selon la situation (par exemple, afficher « majeur » ou « mineur » selon l'âge de l'utilisateur), il a besoin de **prendre des décisions**. C'est le rôle des **structures conditionnelles** : elles permettent au programme de tester une condition et de choisir quoi faire en fonction du résultat.


**Prérequis :** Les types de base.

## ① Mémo

**Définition 1 (La structure `if / elif / else`)**

```
if condition1:
    bloc1          # exécuté si condition1 est vraie
elif condition2:
    bloc2          # exécuté si condition1 est fausse ET condition2 est vraie
else:
    bloc3          # exécuté si toutes les conditions sont fausses
```

- Le `elif` et le `else` sont facultatifs.
- On peut enchaîner autant de `elif` que nécessaire.
- L'indentation (quatre espaces) délimite les blocs.

### Opérateurs de comparaison

`==` Égalité (ne pas confondre avec `=` qui est l'affectation).
`!=` Différent de.
`<`, `<=`, `>`, `>=` Comparaisons d'ordre.

### Opérateurs logiques

`and` Les deux conditions doivent être vraies.
`or` Au moins une condition doit être vraie.
`not` Inverse la valeur de vérité.
Priorité : `not` > `and` > `or`. En cas de doute, utiliser des parenthèses.

### Pièges fréquents

- Écrire `if x = 5` au lieu de `if x == 5` (affectation au lieu de comparaison).
- Oublier les deux-points après la condition.
- Oublier l'indentation du bloc.

## ② Exemples

### Programme 1 · Classification d'un nombre

```
def signe(n):
    if n > 0:
        return "positif"
    elif n < 0:
        return "négatif"
    else:
        return "nul"
```

### Programme 2 · Mention au baccalauréat

```
def mention(moyenne):
    if moyenne >= 16:
        return "Très bien"
    elif moyenne >= 14:
        return "Bien"
    elif moyenne >= 12:
        return "Assez bien"
    elif moyenne >= 10:
        return "Admis"
    else:
        return "Non admis"
```

### Programme 3 · Utilisation de and et or

```
age = 15
a_carte = True

if age < 12:
    print("Gratuit")
elif age < 18 or a_carte:
    print("Tarif réduit")
else:
    print("Plein tarif")
```

## ③ Exercices

**Exercice 1** *Catégorie d'âge* Écrire une fonction `categorie(age)` qui renvoie :

- `"enfant"` si l'âge est strictement inférieur à 12 ;
- `"adolescent"` entre 12 et 17 inclus ;
- `"adulte"` à partir de 18.

**Exercice 2** *Opérateurs logiques* Sans exécuter le code, déterminer la valeur affichée pour chaque ligne :

```
a, b = 5, 0
print(a > 0 and b > 0)       # Ligne 1
print(a > 0 or b > 0)        # Ligne 2
print(not (a > 0))            # Ligne 3
print(a > 0 and not b == 0)  # Ligne 4
```

**Exercice 3** *Tarif de bus* Le tarif d'un trajet en bus dépend de l'âge :

- gratuit pour les moins de 4 ans ;
- tarif réduit (0,80 €) pour les 4–17 ans ;
- plein tarif (1,60 €) sinon.

Écrire une fonction `tarif_bus(age)`.

# Solutions des exercices

### Corrigé de l'exercice 1.

```
def categorie(age):
    if age < 12:
        return "enfant"
    elif age <= 17:
        return "adolescent"
    else:
        return "adulte"
```

**Remarque :** pas besoin de tester `age >= 12` dans le `elif` car si on y arrive, c'est que la première condition est fausse (donc `age >= 12`).

### Corrigé de l'exercice 2.

1. `False` : `a > 0` est `True` mais `b > 0` est `False`, et `True and False` donne `False`.
2. `True` : `a > 0` est `True`, et `True or False` donne `True`.
3. `False` : `a > 0` est `True`, donc `not True` donne `False`.
4. `False` : `not b == 0` vaut `not True` soit `False`, et `True and False` donne `False`.

### Corrigé de l'exercice 3.

```
def tarif_bus(age):
    if age < 4:
        return 0
    elif age <= 17:
        return 0.80
    else:
        return 1.60
```

**Vérification :**

- `tarif_bus(3)` renvoie `0`;
- `tarif_bus(15)` renvoie `0.80`;
- `tarif_bus(30)` renvoie `1.60`.