Fiche de révision 1 | Instructions conditionnelles

Introduction

En Python, les instructions conditionnelles permettent d'obtenir des programmes avec des comportements différents selon les valeurs des variables ou selon certains tests. L'instruction principale est `if` qui permet de tester si une condition est vérifiée ou non.

0.1 Variables booléennes

Booléens

Un **booléen** est une valeur logique qui ne peut prendre que deux états :

- `True` (vrai)
- `False` (faux)

Les booléens servent à représenter le résultat d'un test.

Exemple 1

```
pluie = True
froid = False
print(pluie)   # True
print(froid)   # False
```

Remarque 1. `True` et `False` sont des mot-clés réservés par Python et commencent bien par une majuscule.

0.2 Opérateurs de comparaison

Comparaisons

Une **comparaison** est une expression qui renvoie un booléen (`True` ou `False`). Pour effectuer une comparaison, on utilise un des différents opérateurs présentés ensuite. Les opérateurs usuels sont résumés dans le tableau ci-après.

Opérateur	Signification
==	égalité (gauche = droite ?)
!=	différent
<	inférieur strict
<=	inférieur ou égal
>	supérieur strict
>=	supérieur ou égal

Exemple 2

```
a = 5
b = 7
print(a < b)     # True
print(a == b)    # False
print(a != b)    # True
```

Attention

Ne pas confondre = (affectation) avec == (test d'égalité).

0.3 Opérateurs logiques

Logique booléenne

On peut combiner plusieurs conditions avec les opérateurs logiques :

- `and` : vrai si les deux conditions sont vraies.
- `or` : vrai si au moins une condition est vraie.
- `not` : inverse la valeur logique.

Exemple 3

```
age = 16
print(age >= 15 and age < 18)   # True
print(age < 15 or age > 18)     # False
print(not (age >= 18))          # True
```

Attention

`not` est prioritaire sur `and`, et `and` est prioritaire sur `or`. Toujours utiliser des parenthèses pour lever les ambiguïtés.

0.4 if, elif, else

Structure conditionnelle

Une instruction conditionnelle permet d'exécuter des instructions seulement si une condition est vraie.

Syntaxe générale

```
if condition1: # Si la condition1 est vraie
    bloc1 # Alors on exécute le bloc1
elif condition2: # Sinon, si la condition2 est vraie
    bloc2 # Alors on exécute le bloc2
else: #Sinon, dans le cas général
    bloc3
```

Exemple 4

```
note = 12

if note >= 16:
    print("Très bien")
elif note >= 10:
    print("Suffisant")
else:
    print("Insuffisant")
```

Attention

Chaque bloc d'instructions doit être correctement **indenté** (touche tabulation).

Remarque 2. Certaines valeurs sont automatiquement considérées comme `False` : `0`, `0.0`, `""` (chaîne vide), `[]` (liste vide), `None`.

Exemple 5

```
mot = ""
if mot:
    print("Mot non vide")
else:
    print("Mot vide")    # s'exécute
```