Variables booléens et structures conditionnelles

Une variable booléen est une variable qui peut prendre deux états :
- **Vrai** (True en anglais et dans python)
- **Faux** (False en anglais et dans python)

Les variables booléennes servent à plusieurs choses, mais ce qui nous intéresse sont les structures conditionnelles.

Les **structures conditionnelles** permettent l'écriture de **blocs de code alternatifs**.

En pseudo code :

```
SI condition ALORS
        bloc de code exécuté si la condition est VRAI
SINON
        bloc de code exécuté si la condition est FAUSSE
FIN SI
bloc de code hors structure conditionnelle donc exécuté
```

Exemple en python :

```
condition = True
if condition :
    print("Conditionnel 1 : Bloc de code si la condition est VRAI")
else:
    print("Conditionnel 1 : Bloc de code si la condition est FAUSSE")

print("Code fin si 1")

condition = False
if condition :
    print("Conditionnel 2 : Bloc de code si la condition est VRAI")
else:
    print("Conditionnel 2 : Bloc de code si la condition est FAUSSE")

print("Code fin si 2")
```

```
>>> Conditionnel 1 : Bloc de code si la condition est VRAI
>>> Code fin si 1
>>> Conditionnel 2 : Bloc de code si la condition est FAUSSE
>>> Code fin si 2
```

Pour la première structure conditionnelle, le bloc de code du **if** est exécuté.
Pour la deuxième structure conditionnelle, le bloc de code du **else** est exécuté.

Expression Booléenne :

Dans l'exemple précédant, la variable *condition* est une variable booléenne.
Dans la plupart des cas, notre condition consistera en une expression que l'on devra nécessairement évaluer et qui retournera une valeur booléenne.

Conditions numériques :

>	Plus grand que	>=	Plus grand ou égal à	==	Égal à
<	Plus petit que	<=	Plus petit ou égal à	!=	Différent de

Erreur classique : Confondre l'affectation (=) avec la comparaison (==).

Exemples de conditions numériques :

Condition :	2 > 5	5 <= 5	5 < 5	2 == 2	2 != 2
Sortie :	False	True	False	True	False

```
note1 = 9
note2 = 8
if ( note1 + note2 ) / 2 >= 10 :        # dans ce cas 8.5 >= 10
   print("Vous avez la moyenne")
else:
   print("Vous n'avez pas la moyenne")
```

```
>>> Vous n'avez pas la moyenne
```

Autres conditions :

On verra cette année que l'on peut avoir des expressions booléennes qui comparent des chaînes de caractères, des objets, des fonctions...

Plus de 2 embranchements : if, elif, else.

Il est possible de proposer 3 branches ou plus. Après une première branche conditionnelle **if**, chaque branche suivante peut être ajouté avec sa propre condition grâce au mot clé **elif** (contraction de **else if**), l'ensemble pouvant être à nouveau complété d'une ultime branche **else** sélectionnée lorsque toutes les autres ont été écartées.

```
note  = 15
if note < 10 :
        print("échec")
elif note <= 12 :
        print("mention AB")
elif note <= 14 :
        print("mention B")
elif note <= 16 :
        print("mention TB")
else:
        print("mention Félicitations")
```

```
>>> mention TB
```

Opérateurs Booléens

Souvent, nous aurons besoin de combiner des expressions booléenne. Pour ça, il existe 3 opérateurs booléens :

and	Conjonction (ET logique)	*c1* ***and*** *c2* est VRAI uniquement si les deux conditions sont VRAI.
or	Disjonction (OU logique)	*c1* ***or*** *c2* est VRAI si au moins une des deux conditions est VRAI.
not	Négation (NON logique)	***not*** *c* est VRAI si c est FAUX et ***not*** *c* est FAUX si c est VRAI.

Exemple :
print(3>2 **and** *4>5)* # False car la deuxième condition est fausse

Tables de vérité

Une table de vérité donne tous les résultats possibles d'un opérateur booléen en fonction de la ou les entrées.
Pour le Si c1 est faux et c2 est faux, alors c1 ET c2 est faux.
En python, False and False donne False.

ET LOGIQUE (and)

c1	c2	c1 ET c2
F	F	F
F	V	F
V	F	F
V	V	V

OU LOGIQUE (or)

c1	c2	c1 OU c2
F	F	F
F	V	V
V	F	V
V	V	V

NON LOGIQUE (not)

c	non c
F	V
V	F

Priorités des opérateurs booléens :

Comme en mathématiques, les opérateurs booléens ont des priorité opératoire.
- Le **NON** est prioritaire - suivi du **ET** - et enfin du **OU**.

Exemple : *(a* ***or not*** *b* ***and*** *c)* doit être compris comme *(a* ***or*** *((****not*** *b)* ***and*** *c)).*

Paresse des opérateurs :
Dans le cas des conjonctions (ET), si une première condition est fausse, alors les autres conditions ne sont pas vérifiées puisque le résultats sera faux de toute façon.