Les conditions

1. Les conditions simples

Une condition permet d'effectuer un branchement dans un programme. C'est-à-dire que suivant le résultat d'un test, nous allons décider d'exécuter une partie A du code ou une partie B. Si on compare un programme à une suite d'opérations disposées sur une route, le test correspond à un embranchement où l'on doit choisir de tourner à droite ou à gauche. En Python, le résultat d'un test prend deux valeurs : `False` pour désigner un résultat faux et `True` pour désigner un résultat correct. Voici un exemple de la syntaxe :

```
if age >= 18 :
     print("majeur")
else :
     print("mineur")
```

Contrairement aux autres langages, il n'est pas nécessaire de commencer le test par une parenthèse ouvrante et de le terminer par une parenthèse fermante. Le mot-clé `if` ("si" en français) désigne le début d'une condition. Après le `if`, nous trouvons le test exprimé sous forme d'une condition : `age >= 18`. La fin du test est marquée par l'utilisation du symbole : et d'un retour à la ligne. Plus loin, nous trouvons le mot-clé `else` suivi du symbole `:` et d'un retour à la ligne indiquant le début de l'autre embranchement.

L'indentation est un point cher à Python. L'indentation correspond à la disposition particulière du code faisant apparaître des décalages depuis la marge gauche. L'indentation imposée par Python s'avère être un précieux allié, car elle force tous les développeurs à structurer leurs codes et à le faire de la même manière. En Python, l'indentation permet de désigner une sous-partie du code. Ainsi, dans l'exemple précédent, les deux affichages `print` sont décalés de plusieurs espaces par rapport à la position du `if` et du `else`. On peut aussi utiliser une tabulation. Dans une même sous-partie, tous les débuts de ligne doivent se situer au même niveau, ce qui sous-entend que l'on peut se permettre des indentations différentes d'une partie à l'autre. Voici un exemple :

```
if age >= 18 :            IF + condition
    print("Tu es")        |        | branchement 1
    print("majeur")       |        v
else :                    ELSE
       print("Tu es")     |        | branchement 2
       print("mineur")    v        v
print("Terminé")
```

2. Les conditions avec tests multiples

Il est possible dans tout langage de programmation de cumuler plusieurs tests dans une même condition. Par exemple, une voiture peut démarrer si elle a de l'essence ET si la clef de contact est enclenchée. On peut aussi trouver une autre configuration de test : un pneu doit être changé s'il a roulé plus de 50 000 km OU s'il reste moins de 3 mm d'épaisseur de gomme. Les deux conditions logiques ET/OU s'utilisent naturellement comme on le fait dans le langage de tous les jours. Pour clarifier les choses, nous allons présenter le résultat obtenu par les opérateurs logiques ET/OU en fonction des résultats des deux conditions qui les entourent.

Condition 1	Condition 2	Cond1 ET Cond2
VRAI	VRAI	VRAI

VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	FAUX

Dans le cas de l'opérateur ET logique, le résultat retourné est vrai si les deux conditions sont vraies uniquement. Il suffit qu'une seule de ces conditions soit fausse pour que le résultat du ET logique soit faux.

Condition 1	Condition 2	Cond1 ET Cond2
VRAI	VRAI	VRAI
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

Dans le cas de l'opérateur OU logique, le résultat retourné est vrai si au moins une des deux conditions est vraie. Pour que le résultat retourné soit faux, il faut que les deux conditions soient fausses.

Le résultat donné par les deux opérateurs logiques ET/OU ne dépend pas de la position des conditions. Ainsi le résultat de : condition1 ET/OU condition2 reste le même que si on l'avait écrit à l'envers : condition2 ET/OU condition1.

Avec trois opérateurs ET logiques, le principe reste le même. Si les trois conditions sont vraies, alors le résultat global est vrai. Si une seule condition est fausse, le résultat global est faux :

Condition 1	Condition 2	Condition 3	C1 ET C2 ET C3
VRAI	VRAI	VRAI	VRAI
VRAI	FAUX	VRAI	FAUX
FAUX	VRAI	VRAI	FAUX
FAUX	FAUX	VRAI	FAUX
VRAI	VRAI	FAUX	FAUX
VRAI	FAUX	FAUX	FAUX
FAUX	VRAI	FAUX	FAUX
FAUX	FAUX	FAUX	FAUX

Avec trois opérateurs OU logiques, le principe reste le même. Si les trois conditions sont fausses, alors le résultat global est faux. Si au moins une condition est vraie, alors le résultat global est vrai :

Condition 1	Condition 2	Condition 3	C1 OU C2 OU C3
VRAI	VRAI	VRAI	VRAI
VRAI	FAUX	VRAI	VRAI
FAUX	VRAI	VRAI	VRAI
FAUX	FAUX	VRAI	VRAI
VRAI	VRAI	FAUX	VRAI
VRAI	FAUX	FAUX	VRAI

FAUX	VRAI	FAUX	VRAI
FAUX	FAUX	FAUX	FAUX

Dans le cas où vous devez mélanger des opérateurs OU logique avec des opérateurs ET logique dans la même expression, il faut faire attention. En effet, il y a des règles de priorité qui s'appliquent et elles sont généralement mal connues. Dans ce cas précis, il faut ajouter des parenthèses pour lever l'ambiguïté et donner clairement l'ordre dans lequel les tests doivent être effectués. Nous vous présentons un exemple :

(C1 OU C2) ET (C3 OU C4) comparé à C1 OU (C2 ET C3) OU C4

Nous supposons que les conditions C1 et C2 sont vraies et que les conditions C3 et C4 sont fausses. Nous allons évaluer chacune des deux écritures :

```
Pour :                    (C1  OU  C2)   ET    (C3  OU  C4)
Nous avons          →     (V   OU   V)   ET    (F   OU   F)
                    →          V              ET              F
```

Ce qui donne comme résultat final : FAUX.

```
Pour :                    C1    OU   (C2   ET   C3)   OU    C4
Nous avons          →     V     OU   (V    ET    F)   OU    F
                    →     V     OU          F              OU   F
```

Ce qui donne comme résultat final : VRAI.

Lorsque des opérateurs ET logique et OU logique se mélangent dans une expression, retenez qu'il faut placer des parenthèses pour clarifier les tests sous peine d'erreur.

En informatique, nous parlons d'opérateurs ET/OU, comme nous parlons d'opérateurs + - * /. Pourtant, les premiers retournent les valeurs vrai/faux, alors que les seconds retournent le résultat d'un calcul. Cela dit, la mécanique reste la même : qu'ils soient arithmétiques ou logiques, les opérateurs prennent des valeurs en entrée et retournent un résultat au final. Dans cette optique, l'opérateur ET et la multiplication sont deux notions similaires. On parle d'ailleurs d'opérateurs binaires car ces opérateurs prennent deux valeurs en entrée.

3. L'inversion

Il est possible d'utiliser l'inversion (ou négation) lors de l'écriture des conditions. Cet opérateur logique retourne vrai lorsque la condition sur laquelle il est appliqué est fausse, et inversement. Par exemple, nous pouvons écrire :

Si inversion (PartieEnCours) alors affiche "Partie Terminée"

L'inversion est un des rares opérateurs unaires que l'on trouve dans les langages de programmation. La plupart des opérateurs (ET/OU + - * /) s'appliquent sur deux valeurs, l'inversion ne s'applique que sur une valeur, d'où l'adjectif unaire.

4. La syntaxe des tests

Dans le langage Python, nous trouvons les syntaxes suivantes :

- Vrai : `True`
- Faux : `False`
- ET logique : `and`
- OU logique : `or`
- INV logique : `not`
- Égalité : `==`
- Différent : `!= ou <>`
- Supérieur strict : `>`
- Supérieur ou égal : `>=`
- Inférieur strict : `<`
- Inférieur ou égal : `<=`

Traduisez les conditions suivantes en Python :

- La collision a lieu si l'abscisse x est supérieure ou égale à 100.
- La réduction s'applique si l'enfant est mineur ou si la personne a 60 ans et plus.
- Les deux frères ont-ils le même âge ?
- Si tu ne réussis pas ton BAC...
- Si la distance est inférieure (stricte) à deux fois le rayon...

Voici les réponses :

```
x >= 100
age < 18 ou age >= 60
age_frere1 == age_frere_2
not(BAC_reussi)
distance < 2 * rayon
```