Travaux pratiques d'introduction 4

Les structures conditionnelles

Informatique tronc commun MPSI et PCSI

I Présentation

I.1 Les booléens

Les comparaisons et tests d'égalité sont des expressions ordinaires, qui comme les expressions arithmétiques produisent un résultat. Ce résultat est l'une des deux valeurs dites booléennes, l'une représentant le vrai (notée `True`) l'autre le faux (notée `False`).
Les opérateurs de comparaison sont :

- le test d'égalité qui s'écrit `==` et non `=`,
- la différence qui s'écrit `!=`,
- la comparaison qui peut s'écrire : `<`, `<=`, `>` ou `>=`.

De la même manière que l'ensemble des entiers vient avec des opérations arithmétiques, l'ensemble des booléens est associé à trois opérations booléennes permettant d'en combiner les valeurs :

- **`and`** : il faut que les deux opérateurs de comparaison soient évalués à `True`
- **`or`** : il faut qu'au moins un des deux opérateurs de comparaison soit évalué à `True`
- **`not`** : inverse le résultat.

Ces opérations booléennes permettent donc également de combiner plusieurs tests de comparaison et d'égalité dans une unique condition.
Remarque : lors de l'évaluation de `a or b` on sait que le résultat est vrai dès que la résultat de la proposition `a` est vérifiée. Python ne calculera pas le résultat de `b` dans ce cas. De même pour `a and b`, si `a` est faux Python renvoie `False` sans chercher à vérifier `b`. On parle d'évaluation paresseuse.

I.2 Structures conditionnelles

La structure conditionnelle la plus simple est composée par :

- l'instruction **`if`** suivi d'une expression booléenne qui se termine par `:`,
- un bloc d'instructions indenté, ces instructions ne seront exécutées que si l'expression booléenne est évaluée avec la valeur `True`

```
def bac(note) :
    if note >= 10 :
        return 'admis'
```

Si le test permet de discriminer entre deux blocs d'instructions alors la structure conditionnelle s'écrit de la manière suivante :

- l'instruction **`if`** suivi d'une expression booléenne puis de `:`,
- un bloc d'instructions indenté, ces instructions ne seront exécutées que si l'expression booléenne est évaluée avec la valeur `True`,

- l'instruction **`else`** au même niveau que **`if`** suivi de `:`,
- un nouveau bloc d'instructions indenté qui sera effectué si l'expression booléenne est évalué avec la valeur `False`.

```
def bac2(note) :
      if note >= 10 :
            return 'admis'
      else :
            return 'non admis'
```

Si il y a plus de deux cas, il faudra séparer ces différents cas.
La structure conditionnelle s'écrit alors :

- l'instruction **`if`** suivi d'une expression booléenne puis de `:`,
- un bloc d'instructions indenté, ces instructions ne seront exécutées que si l'expression booléenne est évaluée avec la valeur `True`,
- un certain nombre d'instructions **`elif`** au même niveau que **`if`** suivies d'une expression booléenne puis de `:`,
- pour chaque instruction un bloc indenté qui sera effectué si l'expression booléenne vaut `True` et les précédentes valent `False`,
- l'instruction **`else`** au même niveau que **`if`** suivi de `:`,
- un nouveau bloc d'instructions indenté qui sera effectué si toutes les expressions booléennes précédentes sont évaluées avec la valeur `False`

```
def mentionbac(note) :
      if note < 10 :
            return 'non admis'
      elif note < 12 :
            return 'admis sans mention'
      elif note < 14 :
            return 'admis mention assez bien'
      elif note < 16 :
            return 'admis mention bien'
      else :
            return 'admis mention très bien'
```