Introduction à la programmation avec Python

COURS - Comparaison, booléens, tests

J. Boucher

Lycée Pierre-Paul RIQUET, Première NSI

10 septembre 2024

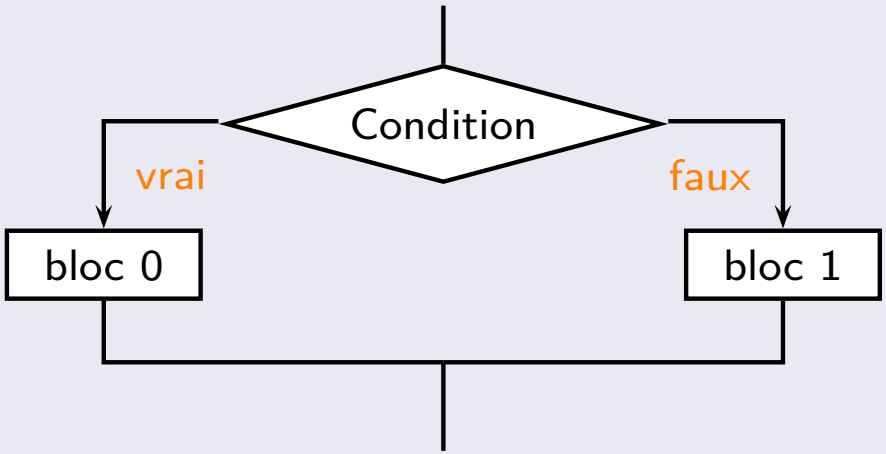
Plan

1. Conditions et branchements

Structures conditionnelles en Python

L'instruction conditionnelle `if/elif/else` est une construction qui permet d'exécuter certains blocs d'instructions selon que des conditions soient remplies ou non.

Logigramme



Python

```
1 a = 16
2 if a%2 == 0:
3     print(a, "est pair")
4 else:
5     print(a, "est impair")
```

On dit qu'elle permet d'effectuer des branchements dans le flot d'exécution du programme.

Les tests

Les conditions qui s'expriment à l'aide d'opérateurs de comparaion sont appelées tests.

- `>` : « supérieur à »
- `>=` : « supérieur ou égal à »
- `<` : « inférieur à »
- `<=` : « inférieur ou égal à »
- `==` : « est égal à »
- `!=` : « est différent de »

→ Le résultat d'un test ne peut être que *vrai* ou *faux*.

Dans l'exemple précédent, l'expression `a%2 == 0` teste l'égalité entre le reste de la division euclidienne de a avec 2 et `0`.

- si a est pair, l'égalité est vraie. On dit que l'expression est évaluée à `True`.
- si a est impair, l'égalité est fausse. On dit que l'expression est évaluée à `False`.

Pour a `16`, 16%2 vaut 0, donc `a%2 == 0` est évaluée à `True`.

2. Expressions booléennes

Les booléens

- L'ensemble des booléens est un ensemble qui ne contient que deux valeurs.
  On le note $\{vrai\ ;\ faux\}$ ou $\{V\ ;\ F\}$ ou $\{1\ ;\ 0\}$.

- En python, une expression booléenne est donc une expression qui est évaluée soit à `True`, soit à `False`.

  Exemple le test `1 > 2`, évalué à `False` par l'interprète, est bien une expression booléenne.

- Tout comme les *entiers*, les *flottants* et les *chaînes de caractères*, *les booléens* représentent un type de base de variable.

  Exemple L'affectation `b = True` correspond à l'état de l'interprète `{ b True }` .
  On dit que la variable nommé `b`, de type booléen, vaut `True`.

Opérations booléennes

- On peut définir des opérations sur les booléens dont le résultat est lui-même un booléen.

Opérateur	Nom	Description
`and`	conjonction	`b1 and b2` est évaluée à `True` lorsque les booléens `b1` et `b2` valent tous les deux `True`.
`or`	disjonction	`b1 or b2` est évaluée à `True` lorsqu'au moins un des booléens, `b1` ou `b2`, vaut `True`.
`not`	négation	`not b1` est évaluée à `True` lorsque `b1` vaut `False`.

- Ces opérations respectent des règles de priorité :

`not` > `and` > `or`

3. Conditionnelles imbriquées

Une instruction conditionnelle peut elle-même contenir une instruction conditionnelle, on parle de structures conditionnelles imbriquées.

```
1 if  score == 51 :
2 |  print("C'est gagné!")
3 else:
4 |  if  score > 51:
5 |  |  print("Dommage!") score = 25
6 |  else:
7 |  |  print("Continue, plus que", 51 - score,"points à
  |  |   marquer") print("Nouveau score :",score)
```

→ Les arbres constituent un bon outil de représentation des branchements de structures conditionnelles imbriquées.