Booléens, instruction conditionnelle

Skander Zannad et Judicaël Courant

2013-09-16

Lire chapitre 5.

1 Booléens

1.1 Types de données déjà vus

- Entiers (12345678901234567890);
- Flottants (9.109e−31);
- Chaînes de caractères ('hello_world');
- Listes ([1, 2, 3, 'soleil ']).

Il en existe d'autres (on peut créer les siens).
Un autre type fondamental : les **booléens**

1.2 Définition

Booléen : type de données ne comprenant que deux valeurs, «vrai» et «faux», représentant les valeurs de vérité.

En Python : True et False

1.3 Opérations

```
>>> not True
False
>>> False and True
False
>>> False or True
True
```

b1	b2	not b1	b1 and b2	b1 or b2
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

1.4 Tests

```
>>> 3.5 > 2
True
>>> 3 <= 2
False
>>> 'bonjour' == 'hello'
False
>>> 3 != 2
True
```

1.5 À quoi servent les booléens ?

- À rendre un résultat de type vrai/faux (exemple : fonction pour tester si un nombre est premier);
- À effectuer une action de façon conditionnelle.

2 Instruction conditionnelle

2.1 Version minimale

Permet d'effectuer une action si un certain booléen est vrai.
Exemple : calcul de l'impôt sur le revenu (simplifié).
5.5% du revenu au delà du seuil de 5963 euros.

```
seuil = 5963
impot = 0
condition = revenu >= seuil
if condition:
    impot = 0.055 * (revenu - seuil)
```

2.2 «Sinon»...

En fait, le calcul est plus compliqué : pour un revenu r exprimé en euros, l'impôt dû est $a \times r - b$ euros suivant le tableau suivant :

r	a	b
$r \leq 5963$	0	0
$5963 < r \leq 11896$	5,5%	327,965
$11896 < r \leq 26420$	14%	1339,125
$26420 < r \leq 70830$	30%	5566,325
$70830 < r \leq 150000$	41%	13357,625
$150000 < r$	45%	21682,675

```
if r <= 5963:
    a = 0
    b = 0
elif r <= 11896:
    a = 0.055
    b = 327.965
elif r <= 26420:
    a = 0.14
    b = 1339.125
elif r <= 70830:
    a = 0.30
    b = 5566.325
elif r <= 150000:
    a = 0.41
    b = 13356.625
else:
    a = 0.45
    b = 21682.675
impot = a * r - b
```

(s'il n'y a que deux cas, pas de `elif`)

2.3 Exemple : test de primalité

On veut tester si n est premier :

```
b = True
for d in range(2,n):
    # b => n pas divisible par 2, 3, ..., d-1.
    if n % d == 0:
        b = False
# b <=> n est premier
```

Remarque : les derniers tours de boucle sont inutiles dès que la variable `b` a été mise à `False`.

2.4 Sortie prématurée de boucle

Instruction `break` :

```
b = True
for d in range(2,n):
```

```
# b => n pas divisible par 2, 3, ..., d-1.
if n % d == 0:
    b = False
    break
# b <=> n est premier
```