

1 Le type

Le nom du type des booléens est `bool`, du nom de George Boole [1815-1864] : mathématicien anglais qui s'est intéressé aux propriétés algébriques des valeurs de vérité.

2 L'ensemble des valeurs

Il s'agit d'un ensemble de seulement 2 données pertinentes, dites *valeurs de vérité* : `{True, False}`.

3 Les opérations

Opérations booléennes les plus courantes sont :

Opération	Entrée	Sortie
<code>not</code>	<code>bool</code>	<code>bool</code>
<code>and</code>	<code>bool × bool</code>	<code>bool</code>
<code>or</code>	<code>bool × bool</code>	<code>bool</code>

4 La sémantique des opérations

De manière similaire aux tables de multiplications, on écrit facilement les tables de ces fonctions puisque le type est fini (et petit).

<code>not</code>	<code>True</code>	<code>False</code>
	<code>False</code>	<code>True</code>

<code>and</code>	<code>True</code>	<code>False</code>
<code>True</code>	<code>True</code>	<code>False</code>
<code>False</code>	<code>False</code>	<code>False</code>

<code>or</code>	<code>True</code>	<code>False</code>
<code>True</code>	<code>True</code>	<code>True</code>
<code>False</code>	<code>True</code>	<code>False</code>

5 Exemples

Exemples de calculs sur les booléens :

```
>>> True or False
True
>>> true or false
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'true' is not defined
>>> True and False
False
>>> not(True)
False
>>> type(True)
<type 'bool'>
>>> True and
  File "<stdin>", line 1
    True and
    ^
SyntaxError: invalid syntax
```

Comme on le voit :

- l'opération `type` écrit bien le type d'une donnée ;
- la syntaxe est stricte, un oubli de majuscule suffit à faire une erreur ;
- ces calculs ont peu d'intérêt en soi, mais deviennent indispensables lorsqu'on veut manipuler des conditions sur les types plus élaborés :

```
>>> if (5 > 2+3) or (8 == 2 ** 3) :  
...     print("OK!")  
... else :  
...     print("Faux!")  
...  
OK!
```