T1.2 Valeurs booléennes

{{ initexo(0) }}

!!! history “Histoire de l’informatique”

```
[George Boole](https://fr.wikipedia.org/wiki/George_Boole){:target="_blank"} (1815-1864) est

Cette algèbre binaire n'accepte que deux valeurs, 0 et 1, et a donc d'importantes et nombreu
```

1.2.1 Un peu de logique

En informatique, comme en mathématiques, on s’intéresse à la valeur de vérité de phrases ou d’expressions qui peuvent être soit vraies, soit fausses. Mais rien d’autre, c’est le principe du tiers-exclu{: target="_blank"}.

Par exemple, que diriez-vous de ces phrases?

- A: Vous êtes en classe de première.
- B: Baudelaire a écrit «Les fleurs du mal».
- C: La Terre est plate.
- D: $3 \times 4 = 12$.
- E: La lettre `e` est dans le mot `abracadabra`.
- F: Georges Perec a écrit un roman de près de 300 pages sans aucune lettre `e`.
- G: $2^{10} < 10^3$
- H: La couleur orange est la plus belle des couleurs.

1.2.2 Algèbre de Boole

!!! abstract “Valeurs et opérations fondamentales” L’algèbre de Boole consiste à étudier des opérations sur un ensemble uniquement constitué de deux éléments qu’on appelle **booléens**.

```
Selon le contexte (logique, calcul, électronique), ces deux éléments sont notés:

- Faux (F) / Vrai (V)
- 0 / 1
- `False`/`True` (en Python, comme dans de nombreux langages)

Les opérations fondamentales ne sont plus l'addition et la multiplication mais:

- la **négation**, notée ¬, ou plus simplement «NON» (`not` en Python);
- la **conjonction**, notée &, ou plus simplement «ET» (`and` en Python);
- la **disjonction**, notée |, ou plus simplement «OU» (`or` en Python).
```

??? note “Tables de vérité” === “Négation, ¬, «NON», `not`” |x|¬x| |:-:|:-:| |F|V| |V|F|

```
=== "Conjonction, &, «ET», `and`"
    |x|y|x & y|
    |:-:|:-:|:-:|
    |F|F|F|
    |F|V|F|
    |V|F|F|
    |V|V|V|

=== "Disjonction, |, «OU», `or`"
    |x|y|x \| y|
    |:-:|:-:|:-:|
    |F|F|F|
    |F|V|V|
    |V|F|V|
    |V|V|V|
```

1.2.3 Avec Python

!!! note “`True` & `False`” - Il existe deux valeurs booléennes en Python : `True` et `False`. - Une variable prenant l’une de ces deux valeurs est de type `bool`.

```
```python
>>> type(True)
<class 'bool'>
>>> x = False
>>> x
False
>>> type(x)
<class 'bool'>
```
```

!!! info inline end “Opérateurs de comparaison” |Opérateur|Signification| |:-:|:-:| |==| est égal à| |!=|est différent de| |<|inférieur à| |>|supérieur à| |<=|inférieur ou égal à| |>=|supérieur ou égal à| |in| appartient à| |not in| n’appartient pas à|

!!! note “Exemples” `python >>> a = 2 >>> a == 3 False >>> a == 2 True >>> a != 1 True >>> a > 2 False >>> a <= 5 True >>> a % 2 == 0 True >>> x = (0 == 1) >>> x False >>> y = (3 + 2 == 5) >>> y True >>> 'e' in 'abracadabra' False >>> 'b' in 'abracadabra' True >>> 'A' not in 'abracadabra' True >>> not True False >>> True and False False >>> True and True True >>> False or True True >>>`

1.2.4 Exercices

{{ initexo(0) }}

!!! example “{{ exercice() }}” === “Énoncé” Prédire si les variables suivantes contiennent le booléen `True` ou le booléen `False`. Contrôlez ensuite en exécutant le code et en inspectant le contenu des variables.

    ```python
    a = (2 > 1)
    b = (3 == 1+2)
    c = (1 < 0)
    d = (2 != 5/2)
    e = (2 != 5//2)
    f = ('a' == 'A')
    g = not a
    h = b and c
    i = b or c
    j = not c and (d or e)
    ```
=== "Correction"
    {{ correction(False,
    "
    "
) }}

!!! example “{{ exercice() }} : le «ou exclusif»” === “Énoncé” Une autre fonction logique importante est le **ou exclusif**, ou «disjonction exclusive».

    C'est le «ou» dans le sens de «Fromage *ou* dessert» dans un menu au restaurant. Soit l'

    Il se note en général `xor` ou `^` en logique (`^` en Python).

    Si x et y sont deux booléens, alors **x ^ y = (x & ¬y) | (¬x & y)**.

    Construire la table de vérité du `xor`.
=== "Correction"
    {{ correction(False,
    "
x	y	x ^ y
F	F	F
F	V	V
V	F	V
V	V	F
    "
) }}

!!! example “{{ exercice() }}” === “Énoncé” Construire la table de vérité de l'expression **(x | y) & z** où x, y et z sont trois booléens.

```
=== "Correction"
    {{ correction(False,
    "
    |x|y|z|x \| y|(x \| y) & z|
    |:-:|:-:|:-:|:-:|:-:|
    |F|F|F|F|F|
    |F|V|F|V|F|
    |V|F|F|V|F|
    |V|V|F|V|F|
    |F|F|V|F|F|
    |F|V|V|V|V|
    |V|F|V|V|V|
    |V|V|V|V|V|
    "
    ) }}
```

!!! example "{{ exercice() }}" === "Énoncé" À l'aide de tables de vérité, démontrer les lois de Morgan:

```
    - ¬(x | y) = ¬x & ¬y
    - ¬(x & y) = ¬x | ¬y
```

```
=== "Correction"
    {{ correction(False,
    "
    - ¬(x | y) = ¬x & ¬y

    |x|y|x \| y|¬(x \| y) | ¬x | ¬y |¬x & ¬y|
    |:-:|:-:|:-:|:-:|:-:|:-:|:-:|
    |F|F|F|V|V|V|V|
    |F|V|V|F|V|F|F|
    |V|F|V|F|F|V|F|
    |V|V|V|F|F|F|F|

    - ¬(x & y) = ¬x | ¬y

    |x|y|x & y|¬(x & y) | ¬x | ¬y |¬x \| ¬y|
    |:-:|:-:|:-:|:-:|:-:|:-:|:-:|
    |F|F|F|V|V|V|V|
    |F|V|F|V|V|F|V|
    |V|F|F|V|F|V|V|
    |V|V|V|F|F|F|F|

    "
    ) }}
```