

Expressions booléennes

Une expression booléenne (vient de **George Boole**, mathématicien britannique du XIX^e siècle) est une expression pouvant être vraie ou fausse mais pas les deux. Par exemple l'expression $1 < 2$ est évaluée à vraie (True) dans l'interpréteur Python alors que $1 > 2$ est évaluée à fausse (False).

Dans la suite, pour des raisons de simplicité on associe la valeur 1 à la valeur Vraie

```
>>> 1 < 2
True
>>> 1 > 2
False
```

(True) et la valeur 0 à la valeur Faux (false)

Les **conditions d'arrêt** ou les test sont des expressions booléennes parfois complexes

```
>>> type(1)
<class 'int'>
>>> type(True)
<class 'bool'>
>>> 1 == True
True
>>> 0 == False
True
```

car composées d'autres expressions booléennes reliées entre elles par des **opérateurs booléens** ou **connecteurs booléens** (Voir exercice). Un **opérateur booléen** permet de connecter deux expressions booléennes pour former une nouvelle expression booléenne.

Nous utiliserons principalement les opérateurs **et**, **ou** puis **non**

a	b	a et b
0	0	0
0	1	0
1	0	0
1	1	1

Voici la table de vérité de l'opérateur **et**

On retiendra que **a et b n'est vrai que lorsque a et b sont vrais** de ce fait le langage Python n'évalue pas l'expression b si a est fausse

Exemple d'application

Problème On dispose d'un ensemble de nombres **dans lequel** peut se trouver plusieurs fois le nombre 0

On veut compter le nombre de fois où $x > \frac{1}{x}$ (évidemment pour x est différent de 0)

une façon de faire est :

```

nb = 0
#pour chaque nombre x de l'ensemble faire
if x != 0:
    if x > 1/x:
        nb += 1

```

Une autre façon de faire est de laisser Python évaluer l'expression booléenne

```
x != 0 and f(x)
```

de telle sorte que si le premier terme $x \neq 0$ est évalué à faux dans ce cas $x > 1/x$ n'est pas évalué et il n'y a pas de division par zéro!

```

nb = 0
#pour chaque nombre x de l'ensemble faire
if x != 0 and x > 1/x:
    nb += 1

```

Voici la table de vérité de l'opérateur **ou**

a	b	a ou b
0	0	0
0	1	1
1	0	1
1	1	1

On retiendra que a et b n'est faux que lorsque a et b sont faux de ce fait le langage Python n'évalue pas l'expression b si a est vraie

Voici la table de vérité de l'opérateur **non**

a	non a
0	1
1	0

On verra en exercices qu'il existe 16 connecteurs logiques pour deux expressions booléennes et que tous ces connecteurs s'expriment en fonction des connecteurs **non**, **et**, **ou**

Ces connecteurs apparaissent donc comme des connecteurs fondamentaux

Exercices

Ex 1

1. Combien de connecteurs logiques à 2 places ?
2. Combien de connecteurs logiques à 3 places ?
3. Combien de connecteurs logiques à n places ?

Ex 2

1. Montrer que $\text{non}(a \text{ et } b)$ a la même table de vérité que $\text{non}(a)$ ou $\text{non}(b)$
2. En déduire que le connecteur logique **et** s'exprime en fonction des connecteurs **non** et **ou**
3. Montrer que $\text{non}(a \text{ ou } b)$ a la même table de vérité que $\text{non}(a)$ et $\text{non}(b)$
4. Montrer que le connecteur logique **ou** s'exprime en fonction des connecteurs **non** et **et**

Ex 3

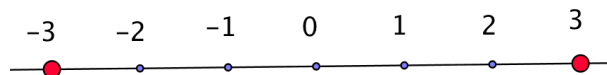
Un opérateur logique très utilisé est le ou exclusif ou xor (en anglais) défini comme le ou à la différence que $\text{xor}(1,1) = 0$

Montrer que $\text{xor}(a,b)$ a la même table de vérité que $(\text{non}(a) \text{ et } b)$ ou $(a \text{ et non}(b))$

Ex 4

Un jeton part de l'origine et se déplace aléatoirement ainsi : Si le lancer d'une pièce a donné **Pile** alors le jeton se déplace d'une unité vers la droite sinon il se déplace d'une unité vers la gauche et le déplacement s'arrête lorsque l'abscisse du jeton est 3 ou -3. Il s'agit de compter le nombre moyen de lancers de la pièce par déplacement

Compléter l'algorithme suivant



Algorithme 1 : Déplacement du jeton

Données : Un jeton à l'origine, un intervalle gradué $[-3;3]$ et un générateur de nombres aléatoires

Résultat : Le nombre de lancers de la pièce

```
1 début
2   position ← 0
3   nbLancers ← 0
4   tant que ..... faire
5       |   piece ← entierAleatoire(0,1)
6       |   .....
7   fin
8   afficher(nbLancers)
9 fin
```
