1. Définition

Un **booléen** en logique et en programmation informatique est un type de **variable à deux états**.
Les variables de ce type sont ainsi soit à l'état ***vrai*** soit à l'état ***faux*** (en anglais *true* ou *false*).

En langage machine, on utilise le bit pour représenter des booléens : ainsi un ***0*** représentera la valeur ***faux*** et un ***1*** représentera la valeur ***vrai***.

En langage de programmation Python, le type d'une telle variable est ***bool***, les deux valeurs possibles sont True ou False.

Exemple : Le test (A=B) peut avoir deux valeurs : Vrai (ou 1) si A est réellement égal à B et Faux (ou 0) sinon. On dit que c'est la variable booléenne associée au test A=B.

2. Tableau des opérateurs relationnels ou de comparaison

Opérateur	Expression	Signification
==	x == y	Égal
!=	x != y	Non égal
>	x > y	Plus grand que
<	x < y	Plus petit que
>=	x >= y	Plus grand ou égal à
<=	x <= y	Plus petit ou égal à
is	x is y	est le même objet
is not	x is not y	n'est pas le même objet

Exemple : Dans le Shell de Python, taper les lignes suivantes et compléter celles en pointillés :

```
>>> x=7
>>> y=17
>>> x==y
.....
>>> x!=y
.....
>>> x>y
.....
>>> x>=y
.....
>>> x<y
.....
>>> x<=y
.....
>>> x is y
.....
>>> x is not y
.....
```

Exercice : Expliquer ces lignes tapées dans le shell de Python :

```
>>>a='encyclopédie1'
>>>b='encyclopédie2'
>>>a==b
False
>>>len(a)
13
>>>a[:12]==b[:12]
True
```

3. Les opérateurs logiques

L'algèbre de Boole utilise plusieurs opérateurs que l'on nomme opérateurs booléens, opérateurs logiques, ou encore fonctions logiques ou portes logiques (terme plus propre à l'électronique).
Les principaux opérateurs logiques sont le NON, le OU, le ET et le OU exclusif.

*** Le NON : négation, contraire**

En Python, cette instruction s'appelle ***not***

Exemple : Dans le Shell de Python, taper les lignes suivantes et compléter celles en pointillés :

```
>>> 2<1
.....................
>>> not 2<1
.....................
```

On peut énumérer toutes les possibilités d'un opérateur logique dans un tableau appelé **table de vérité.**

Table de vérité du NON : On note *p* une variable de type booléen

Table de vérité en Python

p	*not p*
True	
False	

Table de vérité en langage binaire

p	$\overline{p}$
1	
0	

*** Le OU logique :** En Python, cette instruction s'appelle ***or***

Exemple : Dans le Shell de Python, taper les lignes suivantes et compléter celles en pointillés :

```
>>> 8<9
.................
>>> 2<1
.................
>>> (8 < 9) or (2 < 1)
.................
```

Table de vérité du OU : On note *p* et *q* deux variables de type booléen
Retrouvez à l'aide de Python la table de vérité du OU

Table de vérité en Python

p	*q*	*p or q*
True	*True*	
True	*False*	
False	*True*	
False	*False*	

Table de vérité en langage binaire : le OU est représenté par +

p	*q*	*p + q*

*** Le ET logique :** En Python, cette instruction s'appelle ***and***

Exemple : Dans le Shell de Python, taper les lignes suivantes et compléter celles en pointillés :

```
>>> 8<9
.....
>>> 2<1
.....
>>> (8 < 9) and (2 < 1)
.....
>>> x=36
>>> (x > 13) and (x < 27)
.....
>>> x=20
>>> (x > 13) and (x < 27)
.....
```

Table de vérité du ET : On note *p* et *q* deux variables de type booléen
Retrouvez à l'aide de Python la table de vérité du ET

Table de vérité en Python

p	*q*	*p and q*
True	*True*	
True	*False*	
False	*True*	
False	*False*	

Table de vérité en langage binaire : le ET est représenté par *

p	*q*	*p* q*

*** Le OU exclusif (se note *xor*):** En Python, cette instruction s'appelle à l'aide de la touche ^

Exemple : Dans le Shell de Python, taper les lignes suivantes et compléter celles en pointillés :

```
>>> 8<9
.....
>>> 2<1
.....
>>> (8 < 9) ^ (2 < 1)
.....
>>> x=36
>>> (x > 13) ^ (x < 27)
.....
>>> x=20
>>> (x > 13) ^ (x < 27)
.....
```

Table de vérité du XOR : On note *p* et *q* deux variables de type booléen
Retrouvez à l'aide de Python la table de vérité du XOR

Table de vérité en Python

p	*q*	*p ^ q*
True	*True*	
True	*False*	
False	*True*	
False	*False*	

Table de vérité en langage binaire : le XOR est représenté par $\oplus$

p	*q*	$p \oplus q$

Ce qu'il faut en retenir :

Soient p et q deux variables de type booléen
NON p : permet d'obtenir le contraire de p
p OU q : si p est vrai ou q est vrai alors (p ou q) est vrai, faux sinon.
p ET q : si p est vrai et q est vrai alors (p et q) est vrai, faux sinon.
P XOR q : soit p est vrai, soit q est vrai alors (p XOR q) est vrai. Si p et q sont vrais ou si p et q sont faux alors (p XOR q) est faux.

Un peu de culture...

George Boole, mathématicien anglais, a utilisé pour la première fois en 1850 une algèbre à 2 éléments pour l'étude de la logique mathématique. Il a défini une algèbre permettant de modéliser les raisonnements sur les propositions vraies ou fausses. Etudiée après Boole par de nombreux mathématiciens, l'Algèbre de Boole a trouvé par la suite de nombreux champs d'application : réseaux de commutation, théorie des probabilités, recherche opérationnelle (étude des alternatives).
Les premières applications dans le domaine des calculateurs apparaissent avec les relais pneumatiques (ouverts ou fermés). Aujourd'hui, les ordinateurs sont composés de transistors électroniques fonctionnant sur 2 modes : bloqué ou passant. Ils utilisent une arithmétique binaire. L'algèbre de Boole constitue un des principaux fondements théoriques pour leur conception et leur utilisation. Les circuits sont des implémentations matérielles de fonctions booléennes.