# Cours - Table de vérité -

## 1 Introduction

En informatique, un booléen est un type de variable à deux états ( vrai ou faux), destiné à représenter les valeurs de vérité de la logique. Il est nommé ainsi d'après George Boole (Royaume-Uni 1815-1864) fondateur de l'algèbre portant son nom. Le type de données booléen est principalement associé à des états conditionnels.

| **True , Vrai , 1** | **False , Faux , 0** |
|---|---|

On a déjà rencontré les 3 opérateurs : **and**, **or** et **not**
Exemples :

```
a=3
b=4
if a==3 and b==4 :
    pass
if  not (a<4):
    pass
if a>3  and a<5 :
    pass
if a>3 or a <5 :
    pass
if "e" in "Coucou":
    pass
```

**Exercice nº 1**

Soit le code Python ci-dessous :

```
>>> a=3.2
>>> b=4
>>> not(a==7)
>>> (a>3) or (b>5)
>>> a==7 and b==4
>>> not(a==7 and b==4)
>>> not(a==7) and not(b==4)
>>> not (int(a))
>>> not(a==7 and b==4)
>>> c="bacon"
>>> "o" in c or "e" in c
>>> len(c)==5
```

1. Quelles sont les lignes qui renvoient une valeur booléenne ?
2. Quelles sont les résultats de ces valeurs booléennes ?

**Exercice nº 2**

Pour chaque cas, dire à quelle(s) condition(s) la boucle **while** va s'arrêter :

```python
#Cas 1
while a>=4:
        ...
```

```python
#Cas 2
while a==4:
        ...
```

```python
#Cas 3
while a!=10:
        ...
```

```python
#Cas 4
while a>4 or b > 0:
        ...
```

```python
#Cas 5
while a>4 and b > 0:
        ...
```

```python
#Cas 6
while a>4 and (b > 0 or c==2):
        ...
```

## 2 Opérateur AND (ET)

Soient $a$ et $b$ deux variables booléennes.

| $a$ | $b$ | $a$ **and** $b$ |
|---|---|---|
| false | false | |
| false | true | |
| true | false | |
| true | true | |

| $a$ | $b$ | $a$ **and** $b$ |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

} table de vérité

## 3 Opérateur OR (OU)

Soient $a$ et $b$ deux variables booléennes.

| $a$ | $b$ | $a$ **or** $b$ |
|---|---|---|
| false | false | |
| false | true | |
| true | false | |
| true | true | |

| $a$ | $b$ | $a$ **or** $b$ |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

} table de vérité

## 4 Opérateur NOT (NON)

Soit $a$ une variable booléennes.

| $a$ | **not** $a$ |
|---|---|
| false | |
| true | |

| $a$ | **not** $a$ |
|---|---|
| 0 | |
| 1 | |

} table de vérité

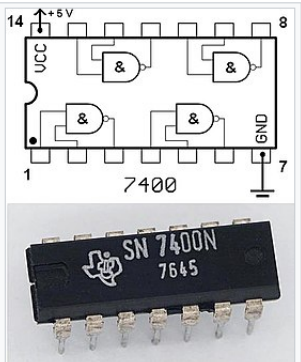
## 5 Table de vérité

**Exercice nº 3**

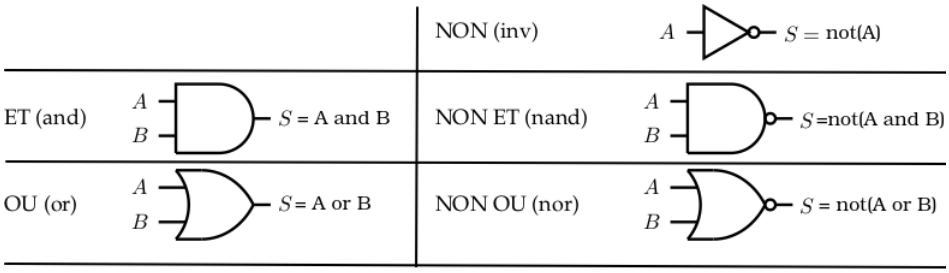
Soient $a$ et $b$ deux variables booléennes. Construire la table de vérité de :

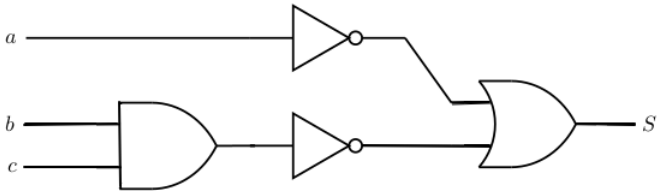
1. not(a and b)
2. not(a) or b
3. (a or b) and not(a and b)

# 6 Circuit logique




Circuit intégré 7400 contenant 4 portes NON-ET (NAND). Les deux autres broches servent à l'alimentation 0V / 5V.




### Exercice nº 4

Donner l'expression booléenne schématisée ci-dessus puis construire sa table de vérité.

# 7 Exercices

### Exercice nº 5

Soit le code Python ci-dessous :

```
>>> a=4
>>> b=10
>>> a+b
>>> a+b==14
>>> not(a==7)
>>> (a>3) or (b>5)
>>> a==7 and b==4
>>> not(a==7 and b==4)
>>> not(a==7) and not(b==4)
>>> not (int(a))
>>> c="bacon"
>>> len(c)
>>> not(len(c)==5)
>>> d=15
>>> not(d>5) or (a==4 and b==10)
>>> d>5 and a==4 and b==10
>>> not(a==4) or not(b==10)
>>> not(a==4 and b==10)
```

1. Quelles sont les lignes qui renvoient une valeur booléenne ?
2. Quelles sont les résultats de ces valeurs booléennes ?

### Exercice nº 6

Soient $a$ et $b$ deux variables booléennes. Construire la table de vérité de :

`(not(a) and b) or (not(b) and a)`

## Exercice n° 7

Soient $a$, $b$ et $c$ trois variables booléennes. Construire la table de vérité de :

1. `(not(a) and b )or c`
2. `a or (b and c)`
3. `(a or b) or not(a or c)`

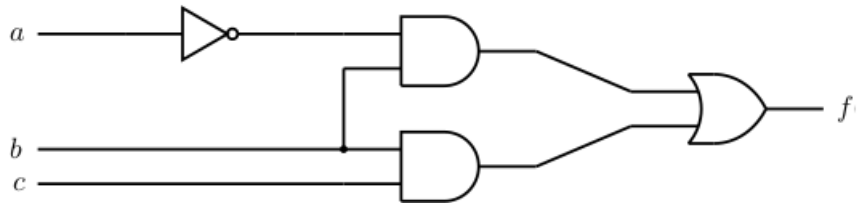
## Exercice n° 8

Soient $a$ et $b$ deux variables booléennes. En construisant les tables de vérité montrer que :

`not(a and b) = (not a) or (not b)`

## Exercice n° 9

*Déterminer la fonction Booléenne associée au circuit suivant :*




## Exercice n° 10

Compléter le tableau suivant qui permet de réaliser l'addition de 2 bits $a$ et $b$.

| $a$ | $b$ | $r$ (retenue) | $u$ (unité) |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | | |
| 1 | 0 | | |
| 1 | 1 | | |

1. Quelle fonction logique permet d'obtenir l'unité ?
2. Quelle fonction logique permet d'obtenir la retenue ?
3. Construire un circuit logique ayant en entrée $a$ et $b$ et en sortie $r$ et $u$.

## Exercice n° 11 Défis

Réaliser un circuit logique ayant en entrée $a$, $b$ et $c$ et qui réalise l'addition binaire des 3 variables. ( On mettra en sortie $r$ et $u$ comme dans l'exercice précédent)

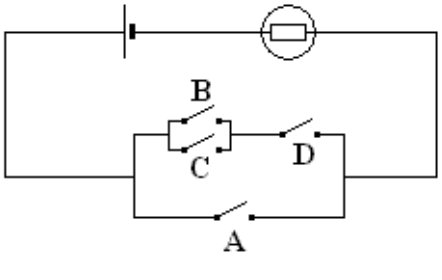
## Exercice n° 12

Construire la table de vérité de l'expression logique `S = not((A or B) and not(B))`

## Exercice n° 13

Pour ce montage électrique, l'association de fonctions logiques correspondant à l'état S = « lampe allumée » en fonction de A, B, C et D a pour expression :

`S = A OU ((B OU C) ET D)`

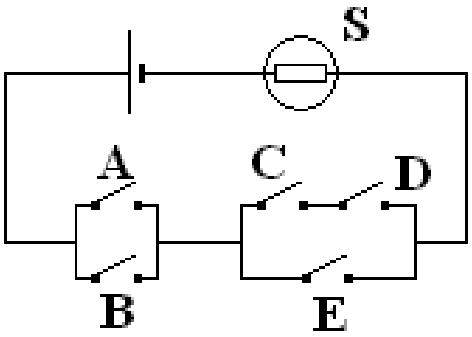


On souhaite réaliser un montage électronique comportant les entrées A, B, C et D, ainsi que des portes logiques, et une sortie S matérialisée par une lampe.

**1.** Ci-dessous, schématiser ce montage : représenter les portes logiques et réaliser les connections.


**2.** Combien y a-t-il de combinaisons possibles pour les entrées (A, B, C et D) : dresser une liste « organisée » de ces possibilités et préciser dans chaque cas sans justification si la lampe est allumée ou non.

**3.** Dans la situation suivante, quelle est l ?expression de l ?association de fonctions logiques correspond à l'état S = « lampe allumée » en fonction de A, B, C, D et E ? Réfléchir sans table de vérité en se demandant « simplement » quand la lampe est-elle allumée.




## Exercice n° 14 Q.C.M

**1.** Parmi les quatre expressions suivantes, laquelle s'évalue en `True` ?
Réponses :

a - `False and (True and False)`
b - `False or (True and False)`
c - `True and (True and False)`
d - `True or (True and False)`

**2.** Quelle expression booléenne pour la variable S correspond à la table de vérité suivante.

| A | B | S |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Réponses :
a - `A ou (non B)`
b - `(non A) ou B`
c - `(non A) ou (non B)`
d - `non (A ou B)`

**3.** Si A et B sont des variables booléennes, laquelle de ces expressions booléennes est équivalente à `(not A) or B` ?
Réponses :
a - `(A and B) or (not A and B)`
b - `(A and B) or (not A and B) or (not A and not B)`
c - `(not A and B) or (not A and not B)`
d - `A and B) or (not A and not B)`

# 8 Projet Codage logique

## 8.1 Opérateur XOR (OU exclusif)

"L'un ou l'autre mais pas les deux"

| $a$ | $b$ | $a$ **xor** $b$ |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

## 8.2 Echauffement

Méthode(s) de codage et de décodage de chaînes binaires [1]
Commencez par donner la représentation binaire, sur un octet, du cryptogramme visuel écrit au dos de la carte de crédit de votre enseignant de NSI : 142

$142_{10}$ = .....$_2$ : cette chaine binaire sera le message à transmettre

A l'aide de la pièce de monnaie, tirez à pile ou face une clé de codage (appelée "masque") composés de bits aléatoires et de même longueur que le message à transmettre.

Masque= .....$_2$

On souhaite transmettre ce cryptogramme visuel de façon sécurisé.
Voici la méthode pour le coder puis le décoder.

**Méthode de codage**

Pour coder le message à transmettre, appliquer, bit à bit, un "ou exclusif" (XOR) entre le message à transmettre et le masque aléatoire.
Message à transmettre (A) : .....
Masque aléatoire (B) : .....
Message codé = A XOR B : .....

**Méthode de décodage**

Maintenant que le message est codé, appliquer, bit à bit, un "ou exclusif" (XOR) entre le message coder et le masque aléatoire (qui a été utilisé pour le codage).
Message codé (A) : .....
Masque aléatoire (B) : .....
A XOR B : .....

Que remarque-t-on ?

1. chaînes de caractères constituées uniquement de "0" et de "1"

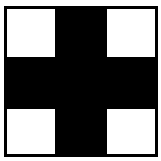
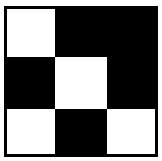
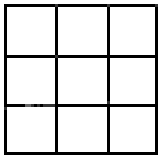
## 8.3 Codage et décodage d'une image

Les dessins ci-dessous sont réalisés avec une grille $3 \times 3$.

| **NOIR 1 True** | **BLANC 0 False** |
|---|---|

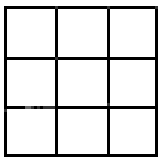
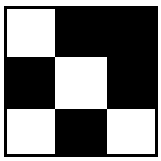
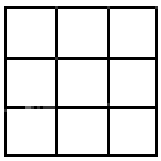
Appliquer le `xor`, pixel par pixel pour obtenir le message codé.

**Codage**

| Message à coder (A) | Masque aléatoire (B) | Message codé (A xor B) |
|---|---|---|
|  |  |  |

**Décodage**

Le message à décoder est celui obtenu ci-dessus.

| Message à décoder (A) | Masque aléatoire (B) | Message décodé (A xor B) |
|---|---|---|
|  |  |  |

## 8.4 Implémentation en Python

1. Télécharger puis décompresser l'archive
2. Créer un fichier `xorVotreNom.py` qui sera compléter au fur à à mesure des questions à partir du fichier `codage_logique.py` fourni.
3. Tester la fonction `lecture()`.

```
>>> lecture('texte.txt')
```

4. Tester la procédure `ecriture()`.

```
>>> ecriture("Coder c'est facile... enfin presque!","nouveau_document.txt")
#Observer qu'un nouveau document texte a été créé.
```

5. Compléter la fonction `masque_aleatoire()`. L'importation de la fonction `randint()` du module `random` sera nécessaire :

```
# 1ere ligne du fichier xorVotreNom.py
from random import randint
print(randint(0,1)) # A tester

# Dans la console Python
>>> masque_aleatoire(8)
'10100111'
```

6. Compléter la fonction `xor_entre()` pour qu'elle renvoie une chaîne binaire résultant d'un "ou exclusif" bit à bit entre les chaînes binaires passées en paramètres.
En python, un "ou exclusif" se code par l'opérateur `^`.

```
>>> "0"^"1"
TypeError: unsupported operand type(s) for ^: 'str' and 'str'
>>> 0^1
1
```

*Remarque : la chaîne binaire retournée est une chaîne de caractères*

```
>>> xor_entre("0011","0101")
'0110'
```

7. Compléter la fonction `convert_10_to_2()` pour qu'elle renvoie une chaîne binaire qui est la représentation binaire sur un octet (8 bits) du nombre en base 10 passé en paramètre.

```
>>> convert_10_to_2(13)
'00001101'
```

8. Dans votre fichier `xorVotreNom.py`, copier (sans faire de modification) et tester les fonctions :
   — `convert_2_to_10`
   — `convert_code_to_caractere`
   — `convert_caractere_to_code`

```
>>> convert_2_to_10('00010110')
22
>>> convert_code_to_caractere(66)
'B'
>>> convert_caractere_to_code('B')
66
```

9. Compléter les fonctions `convert_code_to_text()` et `convert_text_to_code()` en réinvestissant les notions de boucle, d'accumulateur, ainsi qu'en utilisant certaines des fonctions précédemment implémentées.
10. Vous avez tous les outils pour décoder le contenu de `messageA.txt` avec le masque `messageA_masque.txt`
    **Validez votre travail auprès de votre professeur !**
11. A l'aide d'un bloc-notes, écrire dans un fichier `message.txt` une longue tirade personnelle (200 mots minimum).
12. Tapez les instructions nécessaires au codage de ce fichier. Vous générerez ainsi 2 fichiers :
    — `messageCode.txt` (fichier texte contenant votre message codé)
    — `masque.txt` (fichier texte contenant le masque utilisé pour le codage)
13. Envoyer par mail ces deux fichiers sous la forme d'une archive `.zip` :
    **Attention : Ne surtout pas m'envoyer votre fichier `message.txt` !**
14. Cette méthode de chiffrement des données est largement utilisée pour envoyer des informations. Elle possède cependant un point faible, lequel ?