NSi	Variables booléennes	TP6

1. Introduction :

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.

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.

Saisir, testez et complétez les tables de vérité de chacune des fonctions suivantes

2. Fonction OUI :

Définissons une fonction **OUI** qui a comme paramètre une variable **a** et qui renvoie **a**

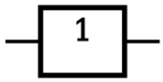
Logigramme : 

Table de verité

a	Valeur retournée
0	
1	

Fonction
```
def OUI (a):
    return a
```

3. Fonction NON :

Définissons une fonction **NON** qui a comme paramètre une variable **a** et qui renvoie le complément de **a**

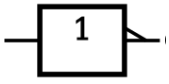
Logigramme : 

Table de verité

a	Valeur retournée
0	
1	

Fonction :
```
def NON (a):
    return not a
```

4. Définissons une fonction **ET** qui a comme paramètres deux variables **a**,**b** et qui renvoie **a and b**

Logigramme : 

Table de verité

a	b	Valeur retournée
0	0	
0	1	
1	0	
1	1	

Fonction : def ET (a,b):
return a and b

5. Définissons une fonction **OU** qui a comme paramètres deux variables **a**,**b** et qui renvoie **a or b**

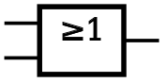
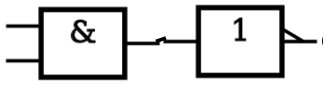
Logigramme : 

Table de verité

a	b	Valeur retournée
0	0	
0	1	
1	0	
1	1	

Fonction def OR (a,b):
return a and b

6. Définissons une fonction **NON-ET** qui a comme paramètres deux variables **a**,**b** et qui renvoie **not (a and b)**

Logigramme : 

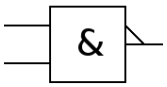
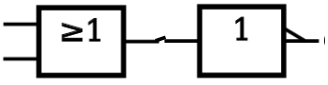


Table de verité

a	b	Valeur retournée
0	0	
0	1	
1	0	
1	1	

Fonction def NAND (a,b):
return not (a and b)

7. Définissons une fonction **NON-OU** qui a comme paramètres deux variables **a**,**b** et qui renvoie **not (a or b)**

Logigramme : 

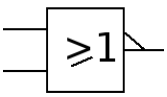
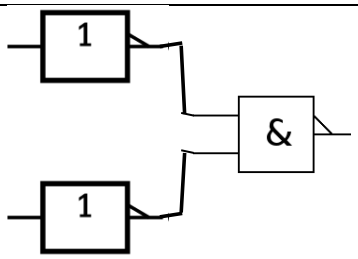


Table de verité

a	b	Valeur retournée
0	0	
0	1	
1	0	
1	1	

Fonction def NOR (a,b):
return not (a or b)

8. Définissons une fonction **MYSTERE :**
Cette fonction a comme paramètres deux variables **a**,**b** et qui renvoie : **not ((not a) and (not b))**

Logigramme :		Table de verité	
Fonction	MYSTERE (a,b): return not ((not a) and (not b))		

a	b	Valeur retournée
0	0	
0	1	
1	0	
1	1	

Après avoir comparé la table de vérité de la fonction **MYSTERE**, déduisez-en à quelle fonction correspond-elle ?