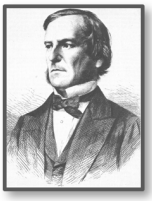
FONCTIONS BOOLEENNES



George Boole (1815-1864),mathématicien anglais, est considéré créateur de la logique moderne. Son algèbre binaire, "L'algèbre de Boole", qu'il exprime en 1854 est utilisée de nos jours en électricité, en électronique et dans la mise de programmes informatiques.

1 – FONCTIONS BOOLEENNES

En informatique, tous les mécanismes de tests sont basés sur la logique booléenne : Une expression est soit vraie (True) soit fausse (False). Les opérateurs de comparaison permettent de comparer deux valeurs et d'indiquer si la condition énoncée est vérifiée ou non (a est plus grand que b, a et b ont la même valeur, etc…). True correspond au niveau logique 1 et False au niveau logique 0.

Une fonction booléenne est une fonction qui associe à une variable booléenne,"Faux" ou "Vrai" (False ou True, 0 ou 1) à une ou plusieurs autres variables booléennes. Ces fonctions booléennes sont utilisées dans en architecture des ordinateurs, dans les programmes informartiques, dans certains algorithmes...

Une fonction booléenne peut s'exprimer par des **expressions utilisant des opérateurs booléens** (non, ou, et ...) ou bien des **tables de vérité**.

2 – FONCTIONS BOOLEENNES ELEMENTAIRES

2.1 – Fonction *non*

a	not(a)
False	
True	

a	not(a)
0	
1	

Question : Exécuter les instructions permettant de compléter les tables de vérité ci-dessus.

In [ ]:

```
not(False)
```

In [ ]:

```
not(True)
```

La fonction booléenne *non* (*not*) permet de complémenter une variable booléenne. C'est à dire que si la variable est "Vrai" la fonction *non* la passe à "Faux" et inversement si la variable est "Faux", la fonction *non* la passe à "Vrai".

2.2 – Fonction *et*

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	

Question : Exécuter les instructions permettant de compléter les tables de vérité ci-dessus.

In [ ]:

```
False and False
```

In [ ]:

```
False and True
```

In [ ]:

```
True and False
```

In [ ]:

```
True and True
```

La fonction booléenne *et* (*and*) retourne un booléen qui est "Vrai" si **le booléen a et le booléen b sont "Vrai"**.

2.3 – Fonction *ou*

a	b	a or b
False	False	
False	True	
True	False	
True	True	

a	b	a and b
0	0	
0	1	
1	0	
1	1	

Question : Exécuter les instructions permettant de compléter les tables de vérité ci-dessus.

In [ ]:

```
False or False
```

In [ ]:

```
False or True
```

In [ ]:

```
True or False
```

In [ ]:

```
True or True
```

La fonction booléenne *ou* (*or*) retourne un booléen qui est "Vrai" si **le booléen a est "Vrai" et/ou le booléen b est "Vrai"**.

2.4 – Fonction *ou exclusif*

a	b	a xor b
False	False	
False	True	
True	False	
True	True	

a	b	a xor b
0	0	
0	1	
1	0	
1	1	

Question : Exécuter les instructions permettant de compléter les tables de vérité ci-dessus.

In [ ]:

```
False ^ False
```

In [ ]:

```
False ^ True
```

In [ ]:

```
True ^ False
```

In [ ]:

```
True ^ True
```

La fonction booléenne *ou-exclusif* (*xor*) retourne un booléen qui est "Vrai" si **le booléen a est différent du booléen b**.

Question : Vérifier et justifier que les résultats des tests correspondent à ceux attendus.

In [ ]:

```
a = 1
b = 2
```

In [ ]:

```
a == 1 and b < 3
```

In [ ]:

```
a == 1 and b == 3
```

In [ ]:

```
a == 1 or b > 3
```

In [ ]:

```
a < 1 or b > 3
```