Les opérateurs booléens

Table des matières

I. Algorithme et algèbre de Boole

Contexte

La logique qui nous occupera dans les pages qui suivent est l'étude des fonctions logiques fondamentales qui traitent des variables binaires en utilisant à la fois des méthodes systématiques et mathématiques.

Les opérations fondamentales de la logique sont ET, OU et NON.

Pour représenter ces trois opérations essentielles lors de l'écriture d'équations logiques, nous utiliserons des signes spécifiques. George Boole, un mathématicien britannique, a eu l'idée originale d'utiliser des notations algébriques pour fournir les bases du raisonnement logique informatique. Ainsi, pour exprimer les relations entre les variables logiques, aussi appelées variables booléennes, nous aurons besoin à la fois de la logique booléenne et de l'algèbre booléenne.

Après George Boole, cette logique a trouvé ses premières applications dans les circuits électriques, comme la commutation téléphonique.

Rappel

Un algorithme est la description d'une suite d'étapes qui permet d'obtenir des résultats à partir d'éléments donnés en entrée. Une recette de cuisine est un exemple d'algorithme permettant d'obtenir un bon plat à l'aide d'ingrédients.

Dans le monde numérique d'aujourd'hui, les algorithmes permettent de combiner diverses informations pour produire une grande variété de résultats, comme par exemple la simulation de l'évolution de la propagation de la grippe en hiver, la comparaison des images numériques de visages ou d'empreintes digitales, ou encore le pilotage de façon autonome des sondes spatiales ou des automobiles. L'algorithme permet donc l'expression claire de concepts de résolution de problèmes indépendants d'un langage de programmation. Son utilisateur doit simplement suivre toutes les instructions afin d'obtenir le résultat que l'algorithme est censé produire.

Un algorithme doit respecter certaines règles afin d'être compréhensible et lisible par plusieurs personnes. Il est composé d'une suite d'instructions à exécuter afin d'atteindre un objectif.

Le corps de l'algorithme

La convention algorithmique se compose des éléments suivants :

- **ID :** l'identité de l'algorithme
- **Fonction :** le rôle de l'algorithme
- **Statut :** les informations communiquées à l'algorithme
- **Conclusion :** le résultat du traitement
- **Théorie :** le principe directeur de l'algorithme

Les éléments sont définis par les mots-clés « *commencer* » et « *fin* ». Ils se terminent par une phrase décrivant les variables utilisées.

Les données sont un ensemble d'informations qui ont été modifiées par un logiciel. Les données d'un algorithme sont stockées sous forme de mémoire centrale dans les variables.

Exemple **Introduction à l'algèbre de Boole**

Dans l'univers noir et blanc des idéaux, il existe une vérité absolue. En l'occurrence, tout est vrai ou faux. Dans ce contexte philosophique, tenez compte des exemples suivants :

- « *Un plus un, donne deux.* » Vrai ou faux ? C'est *probablement* vrai !
- « *1 + 1 = 2* » ET « *2 + 2 = 4* » Vrai ou faux ? C'est vrai également.

Autant qu'il est possible :

« *1 + 1 = 3* » OU « *Sydney se trouve en Australie* » Vrai ou faux ? C'est vrai ! Même si « *1 + 1 = 3* » n'est pas vrai, le Résultat Optimal (RO) dans l'instruction l'indique : si n'importe quelle partie de l'instruction est vraie, la totalité de l'instruction est vraie.

Prenons à présent un exemple plus troublant :

« *2 + 2 = 4* » OU « *1 + 1 = 3* » ET « *1 - 3 = -1* » Vrai ou faux ?

La véracité ou le caractère faux d'une déclaration dépend de l'ordre dans lequel vous l'évaluez.

Si vous indiquez d'abord « *2 + 2 = 4 OU 1 + 1 = 3* », la déclaration est véridique. Pour fixer l'ordre de l'évaluation et éliminer toute ambiguïté, il faut définir certaines règles, comme dans l'algèbre régulière.

Avant de décider comment classer les caractères, nous faisons ce que la plupart des mathématiciens veulent faire en remplaçant les phrases par des symboles. Ces symboles représentent ce que l'on appelle des **variables propositionnelles.**

Soit P la variable « *2 plus 2 est égal à 4* », Q sera la variable « *1 + 1 est égal à 3* » et R sera « *1 - 3 est égal à - 1* ».

L'exemple ci-dessus peut également être écrit plus succinctement comme suit :

P OU *Q* ET *R*

Pour aller plus loin, les mathématiciens ont également substitué les opérations avec des signes OU par « $\vee$ » et ET par « $\wedge$ », la déclaration devient alors :

$P \vee Q \wedge R$

Nous avons aussi besoin d'établir des priorités. On évalue ET d'abord, ensuite OU.

La déclaration P ∨ Q ∧ *R* est donc vraie, on peut écrire :

$P \vee Q \wedge R$=VRAI

Méthode

Opérations fondamentales et notations

Dans cette section, on notera les valeurs `\textrm{VRAI}=1`VRAI=1et `\textrm{FAUX}=0`FAUX=0 comme les bases du logarithme népérien.

Le terme `\textrm` est utilisé pour que le texte soit en police romaine dans une expression mathématique.

Les opérations de base de l'algèbre booléenne sont au nombre de trois, et les notations utilisées dépendent du domaine d'étude (logique, électronique, algébrique, etc.). Nous utilisons une notation ressemblant aux langages de programmation courants comme C ou JavaScript.

Opération	Écriture d'algorithme utilisée	Écriture affichée
ET	&	\times . ou \wedge
OU	‖	ou \vee
NON	!	P', \~P, \neg P ou \overline{P}

Propriété : la logique ET prévaut sur le OU

Remarque

Les variables booléennes en majuscules seront indiquées dans ce cours : P Q R. Il s'agit d'un élément de proposition.

En langage informatique, les syntaxes sont parfois différentes.

Opération	Python	JavaScript
Valeurs booléennes		
VRAI	True	True
FAUX	False	False
Opérateurs booléens		
NON	not a	!a
ET	a and b	a && b
OU	a or b	a || b

Définition **non(x) est Vrai si et seulement si x est Faux.**

Table : non(x)

x	non(x)
F	V
V	F

Ce tableau signifie que si la variable X est fausse, alors non(X) sera Vrai et si X est Vrai, alors non(X) sera Faux.

```
1 print(not True)
2 print(not False)
```

Les résultats de ce listing informatique seront les suivants :

- False
- True

Définition **La fonction et (And)**

x et y est Vrai si et uniquement si x est Vrai et y est Vrai.

Vous trouverez des exemples sur la table de vérité ci-dessous.

Table de vérité : et(x;y)

x	y	et(x;y)
F	F	F
F	V	F
V	F	F
V	V	V

Interprétation du tableau :

- Premièrement, tapez (Faux et Faux), à l'affichage, on aura, Faux.
- Deuxièmement, tapez (Faux et Vrai), à l'affichage, on aura, Faux.
- Troisièmement, tapez (Vrai et Faux), à l'affichage, on aura, Faux.
- Quatrièmement, tapez (Vrai et Vrai), à l'affichage, on aura, Vrai.

Définition **La fonction ou (or)**

x ou y est Vrai si et seulement si au moins x est Vrai ou y est Vrai.

Vous trouverez des exemples sur la table ci-dessous.

Table : ou(x;y)

x	y	ou(x;y)
F	F	F
F	V	V
V	F	V
V	V	V

Interprétation du tableau :

- Premièrement, éditez (Faux or Faux), à l'affichage, nous aurons Faux.
- Deuxièmement, éditez (Faux or Vrai), à l'affichage, nous aurons Vrai.
- Troisièmement, éditez (Vrai or Faux), à l'affichage, nous aurons Vrai.
- Quatrièmement, éditez (Vrai or Vrai), à l'affichage, nous aurons Vrai.

L'algèbre de Boole

L'algèbre de Boole, ou calcul booléen, concerne des opérations et fonctions sur les variables logiques.

- Il faut au moins deux variables, c'est-à-dire prendre deux valeurs afin de pouvoir effectuer des calculs. Ce mode de calcul a été inventé par George Boole (1815-1864), un mathématicien britannique.
- On appelle B l'ensemble constitué de deux valeurs de vérité {FAUX, VRAI}.
- En Python, les deux valeurs de type booléen sont **False** et **True**.

Les opérateurs booléens ont la capacité d'exprimer des opérations logiques simples.

De même que les opérateurs arithmétiques « / » (diviser) et « * » (multiplier) sont prioritaires sur les opérateurs « + » (additionner) et « - » (soustraire), il existe un ordre de priorité pour les opérateurs booléens.

Voici l'ordre de priorité :

- « *not* » est évalué en premier,
- « *et* » est évalué en second,
- « *ou* » est évalué en dernier.

Il est possible de modifier cet ordre en ajoutant des parenthèses ().

Méthode

Exercice : Quiz

[solution n°1 p.19]

Question 1

L'utilisateur d'un algorithme n'a pas besoin de respecter toutes les étapes pour afficher le résultat final.

- Vrai
- Faux

Question 2

Les éléments d'un algorithme sont déterminés par les mots d'ordre « *début* » et « *fin* ».

- Vrai
- Faux

Question 3

L'exactitude ou la fausseté d'une affirmation dépend de l'ordre dans lequel vous l'évaluez.

- Vrai
- Faux

Question 4

Les opérations de base de l'algèbre booléenne sont au nombre de deux.

- Vrai
- Faux

Question 5

x et y est VRAI si et seulement si x est VRAI et y est FAUX.

- Vrai
- Faux

III. Opérateurs logiques et conditions

Opérateurs logiques booléens

Georges Boole a défini trois opérateurs logiques. Ce sont des méthodes de combinaison des variables booléennes entre elles. Négation NON, conjonction ET (multiplication) et disjonction OU (addition) sont les trois concepts en question.

- **Négation NON**

  Cet opérateur s'applique à une seule variable booléenne et génère la valeur négative de la variable. La négation d'une variable booléenne est l'opposé ou le négatif de cette variable.

  « *NON A » est vrai si et seulement si A est faux* .

  « *NON A* » est aussi noté Ā (se lit « *A barre* »).

  Par exemple, si A = 0 alors $\bar{A} = 1$.

- **ET Conjonction**

  Le signe logique de cet opérateur est « *ET* ». Si une ou les deux variables sont à l'état logique 0, le résultat de cette opération pour deux variables booléennes est 0. Le résultat 1 est obtenu lorsque les deux variables d'entrée sont à l'état logique 1.

  « *A ET B* » est vrai si et seulement si A est vrai **et** B l'est aussi.

  « *A ET B* » est aussi noté « *A x B* ».

  Exemple : si A = 0 et B = 1, alors A · B = 0 (les valeurs A et B doivent être vraies pour A x B = 1).

- **Disjonction OU**

  Le signe logique de cet exploitant est « *OU* ». Cet opérateur définit l'addition dans l'algèbre booléenne.

  Si les deux variables ajoutées ont l'état logique 0, le résultat de l'addition booléenne a la valeur 0. Si l'une des deux variables est au moins à l'état logique 1 ou les deux le sont simultanément, elle prend la valeur 1.

  « *A OU B* » est vrai si et seulement si A est vrai **ou** B est vrai.

  « *A OU B* » est aussi noté « *A + B* ».

  Exemple : si A = 0 et B = 1, alors A + B = 1.

Les opérateurs booléens

Dans la logique booléenne, les valeurs possibles sont Vrai ou Faux.

Si a et b sont deux booléens, alors non (a ou b) est équivalent à (non a) et (non b) et non (a et b) est équivalent à (non a) ou (non b).

- **La commande conditionnelle**

  « *Si* », « *alors* » et « *sinon* » permettent de lier le résultat d'un énoncé algébrique à l'exécution d'un algorithme.

  Voici la syntaxe :

  si expression booléenne alors

  suite d'instructions exécutées si l'expression est vrai

  sinon

  suite d'instructions exécutées si l'expression est fausse

  fin si

  Par exemple : si a = 1 alors le résultat est rouge sinon le résultat est vert.

- **L’instruction conditionnelle**

  La deuxième partie de l’instruction est optionnelle, on peut avoir la syntaxe suivante :

  « *Si expression booléenne alors*

  suite d’instructions exécutées si l’expression est vrai

  fin si »

  Par exemple : si a = 1 alors le résultat est rouge.

- **Les répétitions conditionnelles**

  Étant donné que la situation est évaluée avant que les instructions ne soient exécutées, il est possible qu'elles ne soient jamais exécutées en cas de non validité des conditions.

  L'ensemble d’instructions doit avoir un impact sur la condition pour qu'elle soit évaluée comme fausse.

  Assurez-vous toujours que la condition devient fausse à la fin de la période.

Exemple

On souhaite créer un algorithme qui a pour objectif de créer des rectangles de n'importe quelle taille, à condition que la taille des rectangles qu'ils créent soit supérieure à un pixel.

Les répétitions conditionnelles sont :

Nom : saisir LargeurRectangle

Rôle : vérification validité largeur saisie

Données : la largeur

Principe : tant que la largeur est < 1, on demande de ressaisir la largeur

Résultat :

```
début
écrire ("indiquez la largeur du rectangle :") largeur <- lire()
  tant que
       largeur < 1 faire
       écrire ("erreur : indiquez une valeur strictement positive")
       écrire ("indiquez la largeur du rectangle :")
  largeur <- lire()
       ftant fin Lexique - largeur : entier, largeur courante saisie
```

Opération sur les booléens

La vérification si un objet est un booléen ou non avec la fonction type se fait comme suit :

```
1 variable = False
2 print(variable)
3      False
4 print(type(variable))
5      <class ‘bool’>
```

Opérateurs mathématiques, booléens et comparables

Le titre de cette section peut sembler complexe, mais il ne l'est pas autant.

Les opérateurs de calcul

Il existe des symboles mathématiques élémentaires : addition (+), soustraction (-), multiplication (*), division (/) et puissance (**).

Voici un petit exemple concernant les opérateurs de maths que vous pouvez utiliser sur Python :

```
$valeur = 1 + 1
$écart = 20 - 5
$résultat = 15 * 2
$liaison = 30 / 30
$faculté = 2 ^ 2
```

Ces opérateurs mathématiques fournissent des valeurs qui sont affectées à diverses variables. Vous pouvez voir ces valeurs à l'aide de l'inspecteur.

Si vous souhaitez simplement effectuer un calcul simple, vous pouvez le faire de la manière indiquée ci-dessous :

```
écris 2010-12
```

Voici maintenant un exemple avec des parenthèses :

```
écris ( ( 20 - 5 ) * 2 / 30 ) + 1
```

Les premiers calculs seront effectués pour les expressions contenant des parenthèses. Dans cet exemple, « *20 - 5* » sera calculé, multiplié par 2 et divisé par 30 avant d'ajouter 1 (ce qui donne 2). D'autres situations peuvent également nécessiter l'utilisation de parenthèses.

De plus, KTurtle, qui est un environnement de programmation éducatif utilisant le dessin par images vectorielles, possède des fonctionnalités mathématiques plus sophistiquées sous la forme de commandes. Regardez les commandes ci-dessous, mais gardez à l'esprit qu'elles concernent des opérations avancées : arrondi[1], hasard[2], racine[3], pi[4], sin[5], cos[6], tan[7], arcsin[8], arccos[9], arctan[10].

Opérateurs booléens (vrai / faux)

Contrairement aux opérateurs mathématiques, qui fonctionnent avec des nombres, les opérateurs booléens fonctionnent avec des valeurs booléennes (vrai et faux). Il n'y a que trois opérateurs booléens : **and**, **or**, et **not**. L'exemple TurtleScript suivant montre comment les utiliser :

```
a = 6
b = 7
c = 42
print(1, a == 6)
print(2, a == 7)
print(3, a == 6 and b == 7)
print(4, a == 7 and b == 7)
print(5, not a == 7 and b == 7)
print(6, a == 7 or b == 7)
print(7, a == 7 or b == 6)
print(8, not (a == 7 and b == 6))
print(9, not a == 7 and b == 6)
```

Ce qui affiche comme résultat :

```
1 True
2 False
3 True
4 False
5 True
```

1 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#round
2 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#random
3 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#sqrt
4 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#pi
5 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#sin
6 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#cos
7 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#tan
8 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#arcsin
9 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#arccos
10 https://docs.kde.org/stable5/fr/kturtle/kturtle/commands.html#arctan

```
6 True
7 False
8 True
9 False
```

Bien que nous fournissions ces résultats sous forme de brefs commentaires à la fin de chaque ligne, l'utilisation de l'inspecteur vous permet de voir les valeurs.

- **and** ne vaut vrai que si les deux côtés sont vrais.
- **or** est vrai dans le cas où l'un des deux côtés est vrai.
- **not** devient vrai en faux, et vice versa.

Exemple

Considérons l'exemple suivant avec **and** :

L'opérateur **and** permet, dans le cas où toutes les valeurs sont True, de retourner la dernière valeur traitée. Dans un autre cas, il permet de retourner la première valeur si la valeur est False.

```
nom = input("Quel est votre nom? ")
mdp = input("Quel est le mot de passe? ")
if nom == "Kaoutar" and mdp == "One":
  print("Bienvenue Kaoutar")
elif nom == "Hiba" and password == "Two":
  print("Welcome Hiba")
else:
  print("Utilisateur non reconnu")
```

Résultat d'exécution :

```
Quel est votre nom? Kaoutar
Quel est le mot de passe? One
Bienvenue Kaoutar
```

La sortie de trois opérateurs de comparaison est combinée dans ce morceau de Python en utilisant les opérateurs **and**. Cela signifie que pour que le texte « ***Bienvenue Kaoutar*** » s'affiche, les trois opérateurs doivent fournir des informations précises.

Un autre cas avec **or :**

```
True and False or True
True
```

Il faut noter que l'opérateur **and** est prioritaire sur l'opérateur **or.**

Et pour clôturer un exemple avec « *non* » (**not**), cela transforme « *vrai* » en « *faux* » et « *faux* » en « *vrai* ».

Exemple

Algorithme :

```
$n = 1
si non ($n == 3) {
taper "bonjour"
} sinon {
taper "pas bonjour ;-)"
}
```

Opérateurs de comparaison

Examinons simplement cette comparaison :

```
#vérifier si x=2 et y=4 sont égaux
  print(2==4)
#vérifier si x=2 et y=4 ne sont pas égaux
  print(bool(2!=4))
#vérifier si x=2 est strictement supérieur à y=4
  print(2>4)
#vérifier si x=2 est strictement inférieur à y=4
  print(2<4)
#vérifier si x=2 est supérieur ou égale à y=4
  print(bool(2>=4))
#vérifier si x=2 est inférieur ou égale à y=4
  print(2<=4)
```

Résultat d'exécution :

```
False
True
False
True
False
True
```

Le tableau ci-dessous représente les différents modèles de questions.

X == Y	Identique	La réponse est « *vrai* » si **X** est identique à **Y**
X!= Y	Non identique	La réponse est « *vrai* » si **X** non identique à **Y**
X > Y	Transcendant	La réponse est « *vrai* » si **X** est transcendant à **Y**
X < Y	Strictement inférieur	La réponse est « *vrai* » si **X** est plus petit dénominateur que **Y**
X >= Y	Supérieur ou égal	La réponse est « *vrai* » si **X** est transcendant ou proche de **Y**
X <= Y	Inférieur ou égal	La réponse est « *vrai* » si **X** est subalterne ou identique à **Y**

Ici, on en compare le nombre 2 au nombre 4 avec les opérateurs « ***égal à*** », « ***différent de*** », « ***supérieur à*** », « ***inférieur à*** », « ***supérieur ou égal à*** **»,** « ***inférieur ou égal à*** **».** Le résultat de cette comparaison est stocké dans la variable réponse en tant que réelle valeur booléenne.

Avec les opérateurs de comparaison, tous les nombres et variables (qui incluent des nombres) peuvent être comparés les uns aux autres.

Voici tous les opérateurs de comparaison potentiels : And, AndAlso, Or, Xor et Not.

Les opérateurs And, AndAlso, Or et Xor sont tous binaires, puisqu'ils acceptent deux opérations chacun c'est à dire qu'ils prennent en compte deux conditions, tandis que l'opérateur Not est unaire car il n'accepte qu'une seule condition.

Opérateurs de logique binaire

Sur deux expressions booléennes, l'opérateur **Xor** effectue une exclusion logique. Si une seule des deux expressions possibles correspond à **Truevaleur**, **Xor** renvoie **True**. Si l'une des deux phrases est évaluée à **True** puis évaluée à **False**, **Xor** renvoie **False**.

Exemple

Voici un exemple des opérateurs And, Or et Xor.

Déclarer les variables a, b, c, d, e, f, g comme booléen.

a = 23 > 14 And 11 > 8

b = 14 > 23 And 11 > 8

Les instructions précédentes définissent a sur vrai et b sur faux.

c = 23 > 14 Or 8 > 11

d = 23 > 67 Or 8 > 11

Les instructions précédentes définissent c sur vrai et d sur faux.

e = 23 > 67 Xor 11 > 8

f = 23 > 14 Xor 11 > 8

g = 14 > 23 Xor 8 > 11

Les instructions précédentes définissent e sur vrai, f sur faux et g sur faux.

Opérations logiques Court-circuit

Les opérateurs AndAlso et And sont assez similaires puisqu'ils effectuent tous deux une conjonction logique sur deux instructions booléennes. La principale distinction entre les deux est qu'elle présente en outre un comportement de circuit logique. La seconde expression d'une expression AndAlso n'est pas appréciée dans le cas où la première expression de l'expression AndAlso renvoie « *Faux* » car elle n'est pas modifiable au résultat final.

En outre, l'opérateur OrElse procède à une déconnexion logique de court-circuit sur deux expressions booléennes. Si la première expression d'une expression OrElse retourne « *True* », cette dernière n'est pas évaluée car elle n'est pas modifiable au résultat et OrElse renvoie « *True* ».

Méthode

Calcul au niveau du bit

Deux valeurs entières sont évaluées en utilisant des opérations binaires en format binaire (base 2). Ils comparent les bits avec les emplacements correspondants, puis assignent des valeurs basées sur la comparaison.

Exemple

L'exemple suivant représente l'opérateur « *and* ». On tape le code suivant qui signifie que la variable x sera un nombre entier.

```
Dim x As Integer
x = 3 And 5
```

Elle est définie dans l'exemple ci-dessus. Cela se produit pour les motifs suivants :

Ces valeurs sont considérées comme binaires :

3 sous forme binaire est 011

5 sous forme binaire est égal à 101

3 AND 5 = 1

Car

3 vaut 011 en binaire

5 vaut 101

On a donc

011

+ 101

= 001

Exemple

Si nous convertissons 8 en binaire = 1000

et 6 en binaire = 110

nous avons donc

1000

+ 110

= 0000

Alors 8 AND 6 = 0

L'opérateur « *AND* » admet la proposition qui est la conjonction.

Exemple

Le « *OR* » est toujours inclusif, c'est-à-dire que dès qu'il y aura un 1, alors le résultat correspondra à un 1.

Exemple : 5 OR 3 = 7

5 vaut 101 en binaire

3 vaut 011

On a donc

101

+ 011

= 111

Autre exemple :

9 OR 4 = 11

9 vaut 1001

4 vaut 100

On a donc

1001

+ 100

= 1101

Exercice : Quiz

[solution n°2 p.19]

Question 1

Il existe trois méthodes de combinaison des variables booléennes.

- O Vrai
- O Faux

Question 2

Dans l'utilisation de l'opérateur ET, si A = 0 et B = 1, alors A * B = 1.

- Vrai
- Faux

Question 3

Les opérateurs booléens fonctionnent avec des chiffres tout comme les opérateurs mathématiques.

- Vrai
- Faux

Question 4

Dans l'algorithme ci-dessous, l'affichage final renverra « *Faux Vrai Vrai Faux Faux* » :

```
A = non vrai
B = non 3**4 < 4**3
C = non 10 % 3 <= 10 % 2
D = non 3**2 + 4**2 != 5**2
E = non non faux
Taper(A,B,C,D,E)
```

- Vrai
- Faux

Question 5

Dans l'algorithme ci-dessous, l'affichage final renverra : Vrai Vrai Faux vrai Faux.

```
A = 2**3 == 108%100 or 'Bonjour' == 'Salut'
B = vrai ou Faux
C = 100**0.5 >= 50 ou Faux
D = vrai ou vrai
E = 1**100 == 100**1 or 3 * 2 * 1 != 3 + 2 + 1
F = -(1**2) < 2**0 et 10 % 10 <= 20 - 10 * 2
Tapez(A,B,C,D,E)
```

- Vrai
- Faux

Question 6

Quelle sera la réponse si l'on a 9 and 3 = ?

- 0001
- 1010
- 1000

V. Essentiel

Réaliser un algorithme consiste à exprimer en pseudo-code les règles de résolution d'un problème afin de le soumettre à un ordinateur (via un programme). Les variables utilisées pour stocker des données dans des programmes sont les cas de mémoire.

L'algèbre booléenne, également connue sous le nom de calcul booléen, est une branche des mathématiques qui se concentre sur une interprétation algébrique de la logique en termes de variables, d'opérateurs et de fonctions sur des variables logiques. Georges Boole a défini trois opérateurs logiques. Ce sont des méthodes de combinaison des variables booléennes entre elles. Négation NON, conjonction ET (multiplication) et disjonction OU (addition) sont les trois concepts en question.

VI. Auto-évaluation

A. Exercice

Vous postulez dans une entreprise d'informatique. Lors de l'entretien d'embauche, on vous fait passer quelques tests afin d'évaluer vos connaissances.

Question 1 [solution n°3 p.21]

En sachant qu'en binaire 6 = 110 et 5 = 101, calculer 6 AND 5.

Question 2 [solution n°4 p.21]

Donner la table de vérité de la fonction ou (OR).

B. Test

Exercice 1 : Quiz [solution n°5 p.21]

Question 1

Deux expressions booléennes font l'objet d'une disjonction ou d'une inclusion logique lorsque l'opérateur **ET** est utilisé.

- ○ Vrai
- ○ Faux

Question 2

Trois valeurs intégrales sont évaluées à l'aide d'opérations binaires sous forme binaire.

- ○ Vrai
- ○ Faux

Question 3

Les opérateurs de comparaison fournissent des expressions booléennes en comparant l'opérateur le plus à droite à l'opérateur le plus à gauche et en évaluant le résultat comme « *Vrai* » ou « *Faux* ».

- ○ Vrai
- ○ Faux

Question 4

Quelle sera la réponse si on a 8 OR 7 = ?

- 1110
- 0000
- 1111

Question 5

Les seuls types sur lesquels des opérations sur les bits peuvent être effectuées sont les types numériques.

- Vrai
- Faux

Solutions des exercices

Exercice p. 7 Solution n°1

Question 1

L'utilisateur d'un algorithme n'a pas besoin de respecter toutes les étapes pour afficher le résultat final.

- ○ Vrai
- ◉ Faux

L'utilisateur d'un algorithme doit simplement suivre toutes les instructions afin d'obtenir le résultat que l'algorithme est censé produire.

Question 2

Les éléments d'un algorithme sont déterminés par les mots d'ordre « *début* » et « *fin* ».

- ◉ Vrai
- ○ Faux

Les éléments d'un algorithme sont caractérisés par les mots clés « *début* » et « *fin* » et marqués au final par des variables utilisées.

Question 3

L'exactitude ou la fausseté d'une affirmation dépend de l'ordre dans lequel vous l'évaluez.

- ◉ Vrai
- ○ Faux

La véracité ou la fausseté d'un énoncé dépend de l'ordre dans lequel vous l'évaluez.

Question 4

Les opérations de base de l'algèbre booléenne sont au nombre de deux.

- ○ Vrai
- ◉ Faux

Les opérations de base de l'algèbre booléenne sont au nombre de trois : ce sont les opérations « *not* », « *et* », « *ou* ».

Question 5

x et y est VRAI si et seulement si x est VRAI et y est FAUX.

- ○ Vrai
- ◉ Faux

x et y est VRAI si et seulement si x est VRAI et y est VRAI.

Exercice p. 14 Solution n°2

Question 1

Il existe trois méthodes de combinaison des variables booléennes.

- ◉ Vrai
- ○ Faux

Q Les méthodes de combinaison des variables booléennes sont au nombre de trois : NON, ET, OU.

Question 2

Dans l'utilisation de l'opérateur ET, si A = 0 et B = 1, alors A * B = 1.

- ○ Vrai
- ◉ Faux

Q Si A = 0 et B = 1, alors A * B = 0.

Question 3

Les opérateurs booléens fonctionnent avec des chiffres tout comme les opérateurs mathématiques.

- ○ Vrai
- ◉ Faux

Q Contrairement aux opérateurs mathématiques, qui fonctionnent avec des nombres, les opérateurs booléens fonctionnent avec des valeurs booléennes (vrai et faux).

Question 4

Dans l'algorithme ci-dessous, l'affichage final renverra « *Faux Vrai Vrai Faux Faux* » :

```
A = non vrai
B = non 3**4 < 4**3
C = non 10 % 3 <= 10 % 2
D = non 3**2 + 4**2 != 5**2
E = non non faux
Taper(A,B,C,D,E)
```

- ○ Vrai
- ◉ Faux

Q L'algorithme renverra : Faux Vrai Vrai Vrai Faux.

Question 5

Dans l'algorithme ci-dessous, l'affichage final renverra : Vrai Vrai Faux vrai Faux.

```
A = 2**3 == 108%100 or 'Bonjour' == 'Salut'
B = vrai ou Faux
C = 100**0.5 >= 50 ou Faux
D = vrai ou vrai
E = 1**100 == 100**1 or 3 * 2 * 1 != 3 + 2 + 1
F = -(1**2) < 2**0 et 10 % 10 <= 20 - 10 * 2
Tapez(A,B,C,D,E)
```

- (●) Vrai
- ( ) Faux

Il renverra : Vrai Vrai Faux Vrai Faux.

Question 6

Quelle sera la réponse si l'on a 9 and 3 = ?

- (●) 0001
- ( ) 1010
- ( ) 1000

0001 est la bonne réponse donc 9 AND 3 = 1.

p. 16 **Solution n°3**

6 = 110 et 5 = 101

donc

110

AND 101

= 100

donc 6 and 5 = 100 binaire ou 4 en décimal

p. 16 **Solution n°4**

La table de vérité est la suivante : (F correspond à Faux et V à vrai)

x	y	ou(x;y)
F	F	F
F	V	V
V	F	V
V	V	V

Exercice p. 16 **Solution n°5**

Question 1

Deux expressions booléennes font l'objet d'une disjonction ou d'une inclusion logique lorsque l'opérateur **ET** est utilisé.

- ○ Vrai
- ◉ Faux

Deux expressions booléennes font l'objet d'une disjonction ou d'une inclusion logique lorsque l'opérateur **OU** est utilisé.

Question 2

Trois valeurs intégrales sont évaluées à l'aide d'opérations binaires sous forme binaire.

- ○ Vrai
- ◉ Faux

Deux valeurs intégrales sont évaluées à l'aide d'opérations binaires sous forme binaire.

Question 3

Les opérateurs de comparaison fournissent des expressions booléennes en comparant l'opérateur le plus à droite à l'opérateur le plus à gauche et en évaluant le résultat comme « *Vrai* » ou « *Faux* ».

- ○ Vrai
- ◉ Faux

Les opérateurs de comparaison tels que =, >> et >== produisent des expressions booléennes en comparant l'expression de l'opérateur le plus à gauche à l'expression de l'opérateur le plus à droite et en évaluant le résultat comme Vrai ou Faux.

Question 4

Quelle sera la réponse si on a 8 OR 7 = ?

- ○ 1110
- ○ 0000
- ◉ 1111

1111 est la bonne réponse donc 8 OR 7 = 15.

Question 5

Les seuls types sur lesquels des opérations sur les bits peuvent être effectuées sont les types numériques.

- ○ Vrai
- ◉ Faux

Les seuls types sur lesquels des opérations sur les bits peuvent être effectuées sont les types intégraux.