Fiche technique **25** **Les valeurs booléennes**

http://url.univ-irem.fr/ft27

QR Code

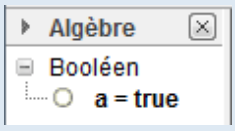
GeoGebra permet de manipuler les valeurs booléennes `true` et `false`. Il est ainsi possible de créer des variables à valeurs booléennes et de créer des tests qui renverront `true` ou `false` selon les conditions précisées. Les valeurs booléennes permettent, en particulier, de cacher ou de montrer des objets en fonction des actions de l'utilisateur (voir la fiche technique **Montrer ou cacher un objet**, page 571).

1 Créer une variable booléenne

On peut affecter la valeur `true` ou `false` à une variable.

Méthode

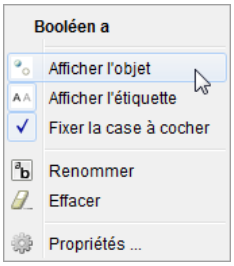
- Positionner le curseur de la souris dans le champ de saisie.
- Inscrire : `a=true`.

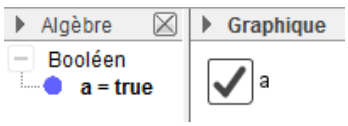


- Valider en appuyant sur la touche ↵ pour créer la variable booléenne a prenant la valeur logique `true`.

Remarque :

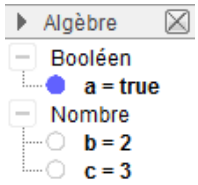
Par défaut, une variable booléenne n'est pas représentée dans la vue ***Graphique***. On peut cependant la rendre visible dans la vue ***Graphique*** (en effectuant, par exemple, un clic avec le bouton droit de la souris sur la variable dans la vue ***Algèbre*** et en cochant, dans le menu contextuel, Afficher l'objet).





La variable booléenne apparaît alors sous la forme d'une case à cocher dont la légende est le nom de la variable. L'état de la case à cocher et la valeur de la variable booléenne sont liés (la variable vaut `true` si la case est cochée, ou `false` sinon).

Remarque :



Les variables booléennes peuvent être utilisées au sein de calculs. Les valeurs logiques `true` et `false` sont alors remplacées respectivement par 1 ou par 0. Par exemple, si `a` désigne une variable booléenne, et si `b=2`, alors `c=a+b` vaut 3 quand `a=true` et vaut 2 quand `a=false`.
Cette manière de traiter les booléens permet aussi de compléter certains champs (vitesse d'animation d'un curseur, couleurs dynamiques, rapport des échelles sur les axes, ...) à l'aide d'un calcul élaboré avec une variable booléenne. Ce qui permet d'obtenir certains effets intéressants dans les figures, en particulier lorsque la variable est dynamiquement modifiable à l'aide d'une case à cocher.

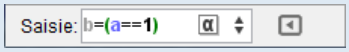
2 Tests et opérateurs logiques

Affecter directement la valeur `true` ou `false` à une variable présente un intérêt assez limité. En revanche, effectuer un test qui renvoie la valeur `true` ou `false` se révèle bien plus intéressant.

Méthode

`a` est une variable numérique déjà définie.

- Positionner le curseur de la souris dans le champ de saisie.
- Inscrire : `b=(a==1)`.



- Valider en appuyant sur la touche ↵ pour créer la variable booléenne `b` prenant la valeur logique `true` lorsque `a` vaut 1, et `false` sinon.

Pour effectuer des comparaisons, utiliser les symboles suivants, accessibles depuis le panneau des caractères spéciaux (bouton α), ou, saisissables au clavier.

Opération	Symbole	Clavier
Égal	$\stackrel{?}{=}$	==
Différent	$\neq$	!=
Inférieur	$<$	<
Inférieur ou égal	$\leq$	<=
Supérieur	$>$	>
Supérieur ou égal	$\geq$	>=

Pour des tests complexes, les opérateurs logiques Et, Ou, Non sont également utilisables.

Opération	Symbole	Clavier
Et	$\wedge$	&&
Ou	$\vee$	||
Non	$\neg$	!

Il existe aussi deux opérateurs booléens qui permettent de comparer les positions relatives de deux droites (il n'existe pas de raccourci clavier pour ces symboles) :

$\parallel$ renvoie `true` si les droites sont parallèles ou `false` sinon.

$\perp$ renvoie `true` si les droites sont perpendiculaires ou `false` sinon.

Quelques exemples :

Exemple(s)

- a>2 renvoie true si $a > 2$, ou false sinon.
- A==B renvoie true si les points A et B sont confondus, ou false sinon.
- (x(A)>=0)&&(y(A)>=0) renvoie true si le point A est situé dans le quadrant positif, ou false sinon.
- (d ∥ d')||(d ⊥ d') renvoie true si les droites d et d' sont perpendiculaires ou parallèles, false sinon.
- ¬(n>90°) renvoie true si la mesure de l'angle n est inférieure ou égale à 90°, ou false sinon.

3 Les commandes logiques

GeoGebra possède quelques commandes qui renvoient une valeur booléenne.

La commande **EstDansRégion**[<point>,<région>] renvoie true si le point <point> appartient à la région <région>, ou false sinon.

Exemple(s)

- **EstDansRégion**[A,poly1] renvoie true si le point A est à l'intérieur du polygone poly1, ou false sinon.
- **EstDansRégion**[A, **Cercle**[B,2]] renvoie true si le point A appartient au disque de centre B et de rayon 2 unités, , ou false sinon.
- **EstDansRégion**[A,2x+y>1] renvoie true si le point A est dans le demi-plan solution de l'inéquation $2x + y > 1$, ou false sinon.

La commande **EstDéfini**[<objet>] renvoie true si l'objet <objet> est défini, ou false sinon.

Exemple(s)

- **EstDéfini**[4/0] renvoie true (car pour GeoGebra, $\frac{4}{0} = \infty$).
- Si $a = -1$, **EstDéfini**[sqrt(a)] renvoie false.
- Si E est le point d'intersection de deux segments, **EstDéfini**[E] renvoie true si les segments sont sécants, ou false sinon.

La commande **EstEntier**[<nombre>] renvoie true si le nombre <nombre> est entier, ou false sinon.

Exemple(s)

- **EstEntier**[12] renvoie true.
- **EstEntier**[2/5] renvoie false.
- **EstEntier**[x(A)] renvoie true si l'abscisse du point A est entière, ou false sinon.

La commande **EstPremier**[<nombre>] renvoie true si le nombre <nombre> est premier, ou false sinon.

Exemple(s)

- `EstPremier[1]` renvoie `false`.
- `EstPremier[2]` renvoie `true`.
- `EstPremier[y(A)]` renvoie `true` si l'ordonnée du point A est un nombre premier, ou `false` sinon.

4 Tests géométriques

GeoGebra possède un certain nombre de commandes (qui renvoient `true` ou `false`) permettant d'effectuer des tests visant à déterminer une éventuelle relation entre différents objets.

La commande `SontAlignés[<point>,<point>,<point>]` renvoie `true` si les trois points fournis en arguments sont alignés, ou `false` sinon.

Exemple(s)

- `SontAlignés[(0,0),(0,1),(0,2)]` renvoie `true`.
- `SontAlignés[A,B,C]` renvoie `false` si $A \notin (BC)$.

La commande `SontCocycliques[<point>,<point>,<point>,<point>]` renvoie `true` si les quatre points fournis en arguments appartiennent à un même cercle, ou `false` sinon.

Exemple(s)

- `SontCocycliques[(1,0),(0,1),(-1,0),(0,-1)]` renvoie `true`.
- `SontCocycliques[A,B,C,D]` renvoie `false` si A, B, C et D ne sont pas sur un même cercle.

La commande `SontConcourantes[<droite>,<droite>,<droite>]` renvoie `true` si les trois droites fournies en arguments passent par un même point, ou `false` sinon (si les droites sont parallèles, la commande renvoie `true`).

Exemple(s)

- `SontConcourantes[Droite[(0,0),(0,1)],Droite[(0,0),(1,0)],Droite[(0,0),(0.5,0.5)]]` renvoie `true` puisque les trois droites passent par l'origine du repère.
- `SontConcourantes[d,e,f]` renvoie `true` si $d \parallel e$ et $e \parallel f$.
- `SontConcourantes[Médiatrice[A,B],Médiatrice[A,C],Médiatrice[B,C]]` renvoie `true`, y compris si les points A, B et C sont alignés.

La commande `SontEgaux[<objet>,<objet>]` renvoie `true` si les valeurs des deux objets fournis en arguments sont égales ou `false` sinon (dans GeoGebra, la valeur d'un objet est visible dans la vue ***Algèbre*** à condition d'avoir sélectionné Valeur dans le menu Options ▶ Descriptions).

Exemple(s)

- `SontEgaux[Segment[(0,0),(0,1)],Segment[(0,0),(1,0)]]` renvoie `true` puisque les deux segments ont la même longueur.
- `SontEgaux[poly1,poly2]` renvoie `true` si `poly1` et `poly2` désignent des polygones de même aire.
- `SontEgaux[y=2x+1,2y-4x=2]` renvoie `true`.

Remarque : Plutôt que d'utiliser la commande `SontEgaux`, il est également possible d'utiliser l'opérateur booléen $\stackrel{?}{=}$. Par exemple, `poly1==poly2` renvoie `true` si les deux polygones `poly1` et `poly2` ont la même aire.

La commande **SontParallèles**[<droite>,<droite>] renvoie true si les deux droites fournies en arguments sont parallèles, ou false sinon.

Exemple(s)

- **SontParallèles**[**Droite**[(0,0),(0,1)],**Droite**[(1,0),(1,1)]] renvoie true.
- **SontParallèles**[d,e] renvoie false si d et e sont des droites sécantes.

La commande **SontPerpendiculaires**[<droite>,<droite>] renvoie true si les deux droites fournies en arguments sont perpendiculaires, ou false sinon.

Exemple(s)

- **SontPerpendiculaires**[**Droite**[A,B],**Médiatrice**[A,B]] renvoie true.
- **SontPerpendiculaires**[d,e] renvoie false si d et e ne sont pas perpendiculaires.

Pour déterminer la véracité des tests précédemment décrits, GeoGebra effectue un certain nombre de contrôles numériques sur les objets passés en arguments. Le logiciel se contente donc de vérifier si les objets semblent être dans la « bonne » configuration ou non, ce qui peut parfois aboutir à des résultats inexacts. Pour que les propriétés géométriques des objets soient prises en compte par GeoGebra, il faut utiliser la commande **Prouver** qui, elle, repose sur une vérification formelle de la figure construite.

La commande **Prouver** adopte la syntaxe suivante : **Prouver**[<expression booléenne>], où <expression booléenne> est l'une des commandes vue plus haut dans ce paragraphe. Cette commande retourne true ou false ou bien non défini lorsque le moteur de vérification formelle ne parvient pas à se prononcer.

Il faut garder à l'esprit que la commande **Prouver** utilise une méthode formelle pour déterminer la véracité d'une expression. Ansi, si les points A, B et C sont trois points libres de coordonnées respectives $(0;0)$, $(1;0)$ et $(0;1)$:

- **SontPerpendiculaires**[**Droite**[A,B],**Droite**[A,C]] renvoie la valeur true car, à l'écran, les droites sont perpendiculaires ;

tandis que

- **Prouver**[**SontPerpendiculaires**[**Droite**[A,B],**Droite**[A,C]]] renvoie la valeur false car les droites (AB) et (AC) ainsi définies ne sont pas toujours perpendiculaires (puisque les points A, B et C sont libres).

Exemple(s)

- Si ABC est un triangle rectangle en A et si DBC est un triangle rectangle en D, alors, **Prouver**[**SontCocycliques**[A,B,C,D]] renvoie true.
- Si A, B et C sont trois points libres, si M est le milieu de $[AB]$ et si N est le milieu de $[AC]$, alors, **Prouver**[**SontParallèles**[**Droite**[M,N],**Droite**[B,C]]] renvoie true.
- **Prouver**[**SontAlignés**[(0,0),(0,1),(0,2)]] renvoie false.

5 Les commandes conditionnelles

GeoGebra possède plusieurs commandes qui prennent en paramètre une valeur booléenne.

La plus intéressante d'entre elles, la commande **Si** possède deux syntaxes :

- **Si**[<booléen>,<objet>] retourne l'objet <objet> si le booléen <booléen> est égal à true.

- `Si[<booléen>,<objet 1>,<objet 2>]` retourne l'objet `<objet 1>` si le booléen `<booléen>` est égal à `true`, ou l'objet `<objet 2>` sinon. Les objets `<objet 1>` et `<objet 2>` doivent être de même nature.

Dans les deux cas, le paramètre booléen peut être écrit sous forme d'une variable booléenne, d'une condition, ou d'une commande logique.

Exemple(s)

- `Si[a,"Bonjour","Au revoir"]` renvoie le texte « Bonjour » si la variable booléenne `a` vaut `true`, ou le texte « Au revoir » sinon.
- `Si[a==5,Médiatrice[A,B]]` affiche la médiatrice du segment $[AB]$ lorsque la variable numérique a vaut 5.
- `Si[EstDéfini[A],Cercle[A,3]]` affiche le cercle de centre A et de rayon 3 unités si le point A est défini.
- `f=Si[x<1,x^2,-x+2]` retourne la fonction f telle que $f(x)=\begin{cases} x^2 & \text{si } x<1 \\ -x+2 & \text{si } x\geqslant 1 \end{cases}$.
- `f=Si[x<0,x^2+1,Si[x<2,-x^2+1,-2x+1]]` retourne la fonction f telle que

$$f(x)=\begin{cases} x^2+1 & \text{si } x<0 \\ -x^2+1 & \text{si } 0\leqslant x<2 \\ -2x+1 & \text{si } x\geqslant 2 \end{cases}$$

- `g=Si[f>0,f]` retourne la fonction g telle que $g(x)=f(x)$ pour tout x où $f(x)>0$.
- `g=Si[Dérivée[f]>0,f]` retourne la fonction g égale à la fonction f sur les intervalles où f est croissante.

Remarque :

En cas d'utilisation dans un script GeoGebra, il est préférable d'imbriquer la commande `Si` dans une autre commande, car celle-ci, contrairement à un langage de programmation « classique », retourne un objet au lieu d'effectuer un branchement conditionnel.
Par exemple, pour affecter la valeur 5 à la variable `a` lorsque la variable `b` vaut 2, en script GeoGebra, on écrira :

```
SoitValeur[a,Si[b==2,5,a]]
```

plutôt que

```
Si[b==2,SoitValeur[a,5]]
```

La commande `GarderSi` accepte deux syntaxes :

- `GarderSi[<condition>,<liste>]` retourne une liste contenant uniquement les éléments de la liste `<liste>` qui vérifient la condition fournie en premier paramètre. Dans la condition `<condition>`, il convient d'utiliser la variable `x` pour faire référence aux éléments de la liste. On utilisera cette syntaxe pour les listes à valeurs numériques.
- `GarderSi[<condition>,<variable>,<liste>]` offre une syntaxe plus souple puisque le paramètre `<variable>` permet de préciser le nom de la variable faisant référence aux éléments de la liste.

Exemple(s)

- `GarderSi[x<5,{15,3,2,7,10}]` retourne la liste $\{3,2\}$.
- `GarderSi[EstEntier[z/2],z,{3,4,6,7,10,5,0,1}]` retourne la liste $\{4,6,10,0\}$.
- `GarderSi[Degré[f]>2,f,{x^5,x,x^3,x^2}]` retourne la liste $\{x^5,x^3\}$.
- `GarderSi[x(U)<0,U,{A,B,C}]` retourne la liste des points qui, parmi les points A, B et C, ont une abscisse négative.
- `GarderSi[AM<2,M,{A,B,C}]` retourne la liste des points qui, parmi les points A, B et C, sont

situés à une distance strictement inférieure à 2 unités du point A.

La commande **NbSi** accepte deux syntaxes :

- **NbSi**[<condition>,<liste>] retourne le nombre d'éléments de la liste <liste> qui vérifient la condition fournie en premier paramètre. Dans la condition <condition>, il convient d'utiliser la variable x pour faire référence aux éléments de la liste. On utilisera cette syntaxe pour les listes à valeurs numériques.
- **NbSi**[<condition>,<variable>,<liste>] offre une syntaxe plus souple puisque le paramètre <variable> permet de préciser le nom de la variable faisant référence aux éléments de la liste.

Exemple(s)

✎ **NbSi**[(x>=2)&&(x<=7),{10,4,7,9,11,14}] retourne le nombre 2.

✎ **NbSi**[x(U)<0,U,{A,B,C}] retourne le nombre de points qui, parmi les points A, B et C, ont une abscisse négative.

✎ **NbSi**[x<3,A1:A10] retourne le nombre de cellules, dans la plage A1 :A10, dont la valeur est strictement inférieure à 3.