SoitValeur(<Booléen b>, <0|1>)

Affecte au booléen *b* la valeur *false* ou *true*.
Vous pouvez aussi utiliser **SoitValeur(<Booléen b>, <false|true>)**

Exemple: Si *b* est un booléen, **`SoitValeur(b,1)`** lui affectera la valeur *true*.

SoitValeur(<Objet A>, <Objet B>)

Si *A* est un objet libre, sa valeur prend la valeur actuelle de *B* (i.e. *A* ne changera pas de valeur si *B* est ensuite modifié).

Si *A* est un point appartenant à un chemin ou une région, et si B est lui aussi un point appartenant au même chemin ou à la même région, l'assignation est correctement réalisée.

Sinon ce qui est appliqué est : SoitValeur(A, PointPlusProche(<Chemin C>, <Point B>)) ou SoitValeur(A,PointPlusProcheRégion(<Région R>, <Point B>)).

Exemple: Si *f* est une fonction, **`SoitValeur(f, ElémentAuHasard({cos(x), 3x+2, ln(x)}))`** assigne aléatoirement mais de manière définitive à f d'être l'une des fonctions proposées dans la liste.

SoitValeur(<Liste L>, <Nombre n >, <Objet O>)

Affecte au *n ème* élément de la liste **libre** *L* la valeur actuelle de l'objet *O* .
Le nombre *n* est au plus égal à Longueur(L)+1.
Cette syntaxe permet donc d'ajouter un élément en fin de la liste de L.

Idée :
Vous avez une liste de points, liste1={A,B,C}, donc un objet dépendant, vous voulez, par script, lui ajouter un point D.
La commande SoitValeur sera sans effet sur liste1, il vous faut copier cette dernière en objet libre par la Commande CopierObjetLibre
Ainsi, exemple de script :

```
liste2=CopierObjetLibre(liste1)
SoitValeur(liste2,4,D).
```

SoitValeur(< Objet dépendant>, ?)

Ceci est une syntaxe spéciale qui rend *non défini* un objet dépendant, ce qui évite de le redéfinir en utilisant `=`

SoitValeur(<Liste déroulante>, <Nombre n >)

Définit *n* comme position sélectionnée dans la liste déroulante.

Commandes Si et SoitValeur dans les Scripts

Dans de nombreux langages de programmation, **Si** signifie *si la condition est vraie fait ceci ; sinon fait cela*.

Dans GeoGebra les arguments de **Si** ne sont pas des commandes, mais des valeurs, dont une devient la valeur du résultat.

Par conséquent, si vous désirez par exemple que *b* prenne la valeur 2 si la condition *a > 2* est réalisée, la méthode correcte pour ce faire est dans le script GeoGebra « Par Actualisation » du nombre *a* `SoitValeur(b,Si(a>2,2,b))`. Toute autre manière d'imbriquer SoitValeur et Si est incorrecte.

SoitValeur dans les Scripts

Dans un script, **privilégiez toujours la commande SoitValeur à l'affectation par le signe =**, cette dernière est moins rapide car elle provoque le recalcul de toute la construction.

pour *i* libre, *j* libre ou non,

les commandes :	sont équivalentes à :
`i = 4`	`SoitValeur(i, 4)`
`i = i + 1`	`SoitValeur(i, i + 1)`
`i = i + j`	`SoitValeur(i, i + j)`

i reste libre, seule sa valeur est modifiée.

`i = j` provoque une redéfinition.
Mais `i = CopierObjetLibre(j)` devrait fonctionner comme SoitValeur.
Une redéfinition est lente, surtout si il y a beaucoup d'objets qui dépendent de *i*.
Cela entraînera de graves problèmes dans les scripts d'actualisation.