# Go - Opérateurs

Un opérateur est un symbole qui indique au compilateur d'effectuer des manipulations mathématiques ou logiques spécifiques. Le langage Go est riche en opérateurs intégrés et fournit les types d'opérateurs suivants -

* Opérateurs arithmétiques
* Opérateurs relationnels
* Opérateurs logiques
* Opérateurs au niveau du bit
* Opérateurs d'affectation
* Opérateurs divers

Ce didacticiel explique les opérateurs arithmétiques, relationnels, logiques, au niveau du bit, d'affectation et d'autres un par un.

## **Opérateurs arithmétiques**

Le tableau suivant montre tous les opérateurs arithmétiques pris en charge par le langage Go. Supposons une variable**A** détient 10 et variable **B** détient 20 alors -

[**Afficher des exemples**](https://tutoriels.edu.lat/pub/go/go-arithmetic-operators/go-operateurs-arithmetiques)

|  |  |  |
| --- | --- | --- |
| Opérateur | La description | Exemple |
| + | Ajoute deux opérandes | A + B donne 30 |
| - | Soustrait le deuxième opérande du premier | A - B donne -10 |
| \* | Multiplie les deux opérandes | A \* B donne 200 |
| / | Divise le numérateur par le dénominateur. | B / A donne 2 |
| % | Opérateur de module; donne le reste après une division entière. | B% A donne 0 |
| ++ | Opérateur d'incrémentation. Il augmente la valeur entière de un. | A ++ donne 11 |
| - | Opérateur de décrémentation. Il diminue la valeur entière de un. | A-- donne 9 |

## **Opérateurs relationnels**

Le tableau suivant répertorie tous les opérateurs relationnels pris en charge par le langage Go. Supposons une variable**A** détient 10 et variable **B** détient 20, alors -

[**Afficher des exemples**](https://tutoriels.edu.lat/pub/go/go-relational-operators/go-operateurs-relationnels)

|  |  |  |
| --- | --- | --- |
| Opérateur | La description | Exemple |
| == | Il vérifie si les valeurs de deux opérandes sont égales ou non; si oui, la condition devient vraie. | (A == B) n'est pas vrai. |
| ! = | Il vérifie si les valeurs de deux opérandes sont égales ou non; si les valeurs ne sont pas égales, la condition devient vraie. | (A! = B) est vrai. |
| > | Il vérifie si la valeur de l'opérande gauche est supérieure à la valeur de l'opérande droit; si oui, la condition devient vraie. | (A> B) n'est pas vrai. |
| < | Il vérifie si la valeur de l'opérande gauche est inférieure à la valeur de l'opérande droit; si oui, la condition devient vraie. | (A <B) est vrai. |
| > = | Il vérifie si la valeur de l'opérande gauche est supérieure ou égale à la valeur de l'opérande droit; si oui, la condition devient vraie. | (A> = B) n'est pas vrai. |
| <= | Il vérifie si la valeur de l'opérande gauche est inférieure ou égale à la valeur de l'opérande droit; si oui, la condition devient vraie. | (A <= B) est vrai. |

## **Opérateurs logiques**

Le tableau suivant répertorie tous les opérateurs logiques pris en charge par le langage Go. Supposons une variable**A** contient 1 et variable **B** détient 0, alors -

[**Afficher des exemples**](https://tutoriels.edu.lat/pub/go/go-logical-operators/go-operateurs-logiques)

|  |  |  |
| --- | --- | --- |
| Opérateur | La description | Exemple |
| && | Opérateur logique ET appelé. Si les deux opérandes sont différents de zéro, la condition devient vraie. | (A && B) est faux. |
| || | Opérateur OU logique appelé. Si l'un des deux opérandes est différent de zéro, la condition devient vraie. | (A || B) est vrai. |
| ! | Opérateur PAS logique appelé. Utilisez pour inverser l'état logique de son opérande. Si une condition est vraie, l'opérateur logique NOT rendra faux. | ! (A && B) est vrai. |

Le tableau suivant présente tous les opérateurs logiques pris en charge par le langage Go. Supposons une variable**A** est vrai et variable **B** est faux, alors -

|  |  |  |
| --- | --- | --- |
| Opérateur | La description | Exemple |
| && | Opérateur logique ET appelé. Si les deux opérandes sont faux, la condition devient fausse. | (A && B) est faux. |
| || | Opérateur OU logique appelé. Si l'un des deux opérandes est vrai, la condition devient vraie. | (A || B) est vrai. |
| ! | Opérateur PAS logique appelé. Utilisez pour inverser l'état logique de son opérande. Si une condition est vraie, alors l'opérateur logique NOT la rendra fausse. | ! (A && B) est vrai. |

## **Opérateurs au niveau du bit**

Les opérateurs au niveau du bit travaillent sur les bits et effectuent des opérations bit par bit. Les tables de vérité pour &, | et ^ sont les suivantes:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| p | q | p & q | p | q | p ^ q |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 |

Supposons que A = 60; et B = 13. Au format binaire, ils seront comme suit -

A = 0011 1100

B = 0000 1101

-----------------

A&B = 0000 1100

A | B = 0011 1101

A ^ B = 0011 0001

~ A = 1100 0011

Les opérateurs Bitwise pris en charge par le langage C sont répertoriés dans le tableau suivant. Supposons que la variable A a 60 et la variable B 13, alors -

[**Afficher des exemples**](https://tutoriels.edu.lat/pub/go/go-bitwise-operators/go-operateurs-au-niveau-du-bit)

|  |  |  |
| --- | --- | --- |
| Opérateur | La description | Exemple |
| & | L'opérateur binaire AND copie un bit dans le résultat s'il existe dans les deux opérandes. | (A & B) donnera 12, soit 0000 1100 |
| | | L'opérateur OR binaire copie un bit s'il existe dans l'un ou l'autre des opérandes. | (A | B) donnera 61, soit 0011 1101 |
| ^ | L'opérateur XOR binaire copie le bit s'il est défini dans un opérande mais pas dans les deux. | (A ^ B) donnera 49, soit 0011 0001 |
| << | Opérateur de décalage binaire gauche. La valeur des opérandes de gauche est déplacée vers la gauche du nombre de bits spécifié par l'opérande de droite. | Un << 2 donnera 240, soit 1111 0000 |
| >> | Opérateur de décalage binaire vers la droite. La valeur des opérandes de gauche est déplacée vers la droite du nombre de bits spécifié par l'opérande de droite. | Un >> 2 donnera 15, soit 0000 1111 |

(x << n == x \* 2 ^ n) (x >> n == x \* 2 ^ (- n))

D'après la spécification à http://golang.org/doc/go\_spec.html , il semble qu'au moins avec des entiers, il s'agit d'un décalage binaire. par exemple, le binaire 00001000 >> 1 serait 00000100 et 00001000 << 1 serait 00010000.

## **Opérateurs d'affectation**

Le tableau suivant répertorie tous les opérateurs d'affectation pris en charge par la langue Go -

[**Afficher des exemples**](https://tutoriels.edu.lat/pub/go/go-assignment-operators/go-operateurs-d-affectation)

|  |  |  |
| --- | --- | --- |
| Opérateur | La description | Exemple |
| = | Opérateur d'affectation simple, attribue des valeurs des opérandes du côté droit à l'opérande du côté gauche | C = A + B attribuera la valeur de A + B à C |
| + = | Ajouter l'opérateur d'affectation ET, il ajoute l'opérande droit à l'opérande gauche et attribue le résultat à l'opérande gauche | C + = A équivaut à C = C + A |
| - = | Soustraire l'opérateur d'affectation ET, il soustrait l'opérande droit de l'opérande gauche et attribuer le résultat à l'opérande gauche | C - = A équivaut à C = C - A |
| \* = | Multiplier l'opérateur d'affectation ET, il multiplie l'opérande droit par l'opérande gauche et affecte le résultat à l'opérande gauche | C \* = A équivaut à C = C \* A |
| / = | Opérateur d'affectation de division ET, il divise l'opérande gauche avec l'opérande droit et affecte le résultat à l'opérande gauche | C / = A équivaut à C = C / A |
| % = | Opérateur d'assignation de module ET, il prend le module en utilisant deux opérandes et affecte le résultat à l'opérande de gauche | C% = A équivaut à C = C% A |
| << = | Opérateur de décalage ET d'affectation à gauche | C << = 2 est identique à C = C << 2 |
| >> = | Opérateur de décalage ET d'affectation à droite | C >> = 2 est identique à C = C >> 2 |
| & = | Opérateur d'affectation AND au niveau du bit | C & = 2 est identique à C = C & 2 |
| ^ = | Opérateur OU exclusif et d'affectation au niveau du bit | C ^ = 2 est identique à C = C ^ 2 |
| | = | OR inclusif au niveau du bit et opérateur d'affectation | C | = 2 est identique à C = C | 2 |

## **Opérateurs divers**

Il existe quelques autres opérateurs importants pris en charge par Go Language, notamment **sizeof** et **?:.**

[**Afficher des exemples**](https://tutoriels.edu.lat/pub/go/go-misc-operator/go-operateurs-divers)

|  |  |  |
| --- | --- | --- |
| Opérateur | La description | Exemple |
| & | Renvoie l'adresse d'une variable. | &une; fournit l'adresse réelle de la variable. |
| \* | Pointeur vers une variable. | \*une; fournit un pointeur vers une variable. |

## Priorité des opérateurs dans Go

La priorité des opérateurs détermine le regroupement des termes dans une expression. Cela affecte la manière dont une expression est évaluée. Certains opérateurs ont une priorité plus élevée que d'autres; par exemple, l'opérateur de multiplication a une priorité plus élevée que l'opérateur d'addition.

Par exemple x = 7 + 3 \* 2; ici, x est attribué à 13, et non à 20 car l'opérateur \* a une priorité plus élevée que +, il est donc d'abord multiplié par 3 \* 2, puis ajouté en 7.

Ici, les opérateurs avec la priorité la plus élevée apparaissent en haut du tableau, ceux avec la priorité la plus basse apparaissent en bas. Dans une expression, les opérateurs de priorité supérieure seront évalués en premier.

[**Afficher des exemples**](https://tutoriels.edu.lat/pub/go/go-operators-precedence/go-priorite-des-operateurs)

|  |  |  |
| --- | --- | --- |
| Catégorie | Opérateur | Associativité |
| Postfix | () [] ->. ++ - - | De gauche à droite |
| Unaire | + -! ~ ++ - - (type) \* & sizeof | De droite à gauche |
| Multiplicatif | \* /% | De gauche à droite |
| Additif | + - | De gauche à droite |
| Décalage | << >> | De gauche à droite |
| Relationnel | <<=>> = | De gauche à droite |
| Égalité | ==! = | De gauche à droite |
| ET au niveau du bit | & | De gauche à droite |
| XOR au niveau du bit | ^ | De gauche à droite |
| OU au niveau du bit | | | De gauche à droite |
| ET logique | && | De gauche à droite |
| OU logique | || | De gauche à droite |
| Affectation | = + = - = \* = / =% = >> = << = & = ^ = | = | De droite à gauche |
| Virgule | , | De gauche à droite |