Opérateurs binaires (bitwise operators)

Un opérateur binaire travaille non sur la valeur d'une variable, mais sur la représentation binaire (une suite de bits) de cette variable, le tout en complément à 2. L'entier signé -5 est représenté par le complément à 2 de la valeur 5-1=4, soit sur 8 bits, le complément à 2 de 00000100 est 11111011.

Python suppose maintenant pour ces opérateurs que les valeurs sont représentées par une suite de bits infinie, ce qui permet de rendre portable ces opérateurs.
Dans ce cas, -5 serait représenté en complément à 2 par 1...1111111111111011.

Liste des opérateurs

x « y : renvoie x avec ses bits décalés y fois vers la gauche
(on rajoute des zéros à droite)
revient à multiplier x par 2^y

```
>> x=1
>> x <<= 1 ; x
2
>> x <<= 1 ; x
4
```

x » y : renvoie x avec ses bits décalés y fois vers la droite
(on rajoute des zéros à gauche)
renvoit $x//2^y$ (et non $x/2^y$) ! division entière !

```
>> x=250
>> x >>= 1 ; x
125
>> x >>= 1 ; x
62
```

x & y : Renvoie un ET logique
Chaque bit de la sortie est à 1 si les bits correspondants dans x et y sont à 1,
sinon 0

```
>> x=1; y=-1
>> x & y
1
>> x=1; y=3
>> x & y
1
```

1 est représenté par 0...000001
-5 est représenté par 1...111011
donc x& y représente 0...000001, soit 1

1 est représenté par 0...000001
3 est représenté par 0...000011
donc x& y représente 0...000001, soit 1

`x | y` : Renvoie un `OU` logique
Chaque bit de la sortie est à 0 si les bits correspondants dans x et y sont à 0, sinon 1

`~x` : Renvoie un `NON` logique
Complément à 2 de x, correspond à -x-1

`x ^ y` : Renvoie un `OU EXCLUSIF` logique, bit à bit, depuis les bits de poids faible. Si les 2 valeurs n'ont pas le même nombre de bit, on complète avec les zéros nécessaires devant la représentation de la plus petite valeur.

`10100^1010` → `11110`