# 运算符

运算符是可以操纵操作数值的符号。

a = 10

b = 20

c = a + b

=,+都是运算符啊10, 20, a, b称为操作数。

Python语言支持以下类型的运算符 -

1.算术运算符

2.比较(关系)运算符

3.赋值运算符

4.逻辑运算符

5.按位运算符

6.成员运算符

7.身份运算符

### 算术运算符

假设变量a的值是10，变量b的值是21，则 -

| **运算符** | **描述** | **示例** |
| --- | --- | --- |
| + | 加法运算，将运算符两边的操作数增加。 | a + b = 31 |
| - | 减法运算，将运算符左边的操作数减去右边的操作数。 | a – b = -11 |
| \* | 乘法运算，将运算符两边的操作数相乘 | a \* b = 210 |
| / | 除法运算，用右操作数除左操作数 | b / a = 2.1 |
| % | 模运算，用右操作数除数左操作数并返回余数 | b % a = 1 |
| \*\* | 对运算符进行指数(幂)计算 | a \*\* b，表示10的21次幂 |
| // | 地板除 - 操作数的除法，其结果是删除小数点后的商数。 但如果其中一个操作数为负数，则结果将被保留，即从零(向负无穷大)舍去 | 9//2 = 4 ， 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0 |

### 比较（关系）运算符

比较(关系)运算符比较它们两边的值，并确定它们之间的关系。它们也称为关系运算符。假设变量a的值10，变量b的值是20，则 -

| **运算符** | **描述** | **示例** |
| --- | --- | --- |
| == | 如果两个操作数的值相等，则条件为真。 | (a == b)求值结果为 false |
| != | 如果两个操作数的值不相等，则条件为真。 | (a != b)求值结果为 true |
| > | 如果左操作数的值大于右操作数的值，则条件成为真。 | (a > b)求值结果为 false |
| < | 如果左操作数的值小于右操作数的值，则条件成为真。 | (a < b)求值结果为 true |
| >= | 如果左操作数的值大于或等于右操作数的值，则条件成为真。 | (a >= b)求值结果为 false |
| <= | 如果左操作数的值小于或等于右操作数的值，则条件成为真。 | (a <= b)求值结果为 true |

实例：

a = 10

b = 20

bRet = a > b

print(bRet)

### 赋值运算符

假设变量a的值10，变量b的值是20，则 -

| **运算符** | **描述** | **示例** |
| --- | --- | --- |
| = | 将右侧操作数的值分配给左侧操作数 | c = a + b表示将a + b的值分配给c |
| += | 将右操作数相加到左操作数，并将结果分配给左操作数 | c += a等价于c = c + a |
| -= | 从左操作数中减去右操作数，并将结果分配给左操作数 | c -= a 等价于 c = c - a |
| \*= | 将右操作数与左操作数相乘，并将结果分配给左操作数 | c \*= a 等价于 c = c \* a |
| /= | 将左操作数除以右操作数，并将结果分配给左操作数 | c /= a 等价于 c = c / a |
| %= | 将左操作数除以右操作数的模数，并将结果分配给左操作数 | c %= a 等价于 c = c % a |
| \*\*= | 执行指数(幂)计算，并将值分配给左操作数 | c \*\*= a 等价于 c = c \*\* a |
| //= | 运算符执行地板除运算，并将 |  |

通常来讲，a += b比a=a+b的效率要高，其他带=号的赋值运算也是一样。

实例：

a = 10

b = 20

a += b

print(a)

### 逻辑运算符

Python语言支持以下逻辑运算符。假设变量a的值为True，变量b的值为False，那么

| **运算符** | **描述** | **示例** |
| --- | --- | --- |
| and | 如果两个操作数都为真，则条件成立。 | (a and b)的结果为False |
| or | 如果两个操作数中的任何一个非零，则条件成为真。 | (a or b)的结果为True |
| not | 用于反转操作数的逻辑状态。 | not(a and b) 的结果为True。 |

实例：

v1 = 10

v2 = 20

v3 = 15

if v1 < v2 and v2 < v3:

print(‘11111’)

if v1 < v2 or v2 < v3:

print(‘22222’)

### 按位运算符

按位运算符执行逐位运算。 假设变量a = 60; 和变量b = 13; 现在以二进制格式，它们将如下 -

a = 0011 1100

b = 0000 1101

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

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

Python的内置函数bin()可用于获取整数的二进制表示形式。

以下是Python语言支持位运算操作符 -

| **运算符** | **描述** | **示例** |
| --- | --- | --- |
| & | 如果它存在于两个操作数中，则操作符复制位到结果中 | (a & b) 结果表示为 0000 1100 |
| IMG_256 | 如果它存在于任一操作数，则复制位。 | (aIMG_257b) = 61 结果表示为 0011 1101 |
| ^ | 二进制异或。如果它是一个操作数集合，但不是同时是两个操作数则将复制位。 | (a ^ b) = 49 (结果表示为 0011 0001) |
| ~ | 二进制补码，它是一元的，具有“翻转”的效果。 | (~a ) = -61有符号的二进制数，表示为1100 0011的补码形式。 |
| << | 二进制左移，左操作数的值由右操作数指定的位数左移。 | a << 2 = 240 (结果表示为 1111 0000) |
| >> | 二进制右移，左操作数的值由右操作数指定的位数右移。 | a >> 2 = 15(结果表示为0000 1111) |

### 成员运算符

Python成员运算符测试给定值是否为序列中的成员，例如字符串，列表或元组。 有两个成员运算符，如下所述 -

| **运算符** | **描述** |
| --- | --- |
| in | 如果在指定的序列中找到一个变量的值，则返回true，否则返回false。 |
| not in | 如果在指定序列中找不到变量的值，则返回true，否则返回false。 |

实例：

list1 = [1, 2, 3, 8, 10, -2]

bRet = -2 in list1

print(bRet)

### 身份运算符

身份运算符比较两个对象的内存位置。常用的有两个身份运算符，如下所述 -

| **运算符** | **描述** |
| --- | --- |
| is | 如果运算符任一侧的变量指向相同的对象，则返回True，否则返回False。 |
| is not |  |

例如:

>>> a = 20

>>> b = 20

>>> id(a)

1466658416

>>> id(b)

1466658416

>>> a is b

True

### 运算符优先级

下表列出了从最高优先级到最低优先级的所有运算符，如下所示 -

| **序号** | **运算符** | **描述** |
| --- | --- | --- |
| 1 | \*\* | 指数(次幂)运算 |
| 2 | ~ + - | 补码，一元加减(最后两个的方法名称是+@和-@) |
| 3 | \* / % // | 乘法，除法，模数和地板除 |
| 4 | + - |  |
| 5 | >> << | 向右和向左位移 |
| 6 | & | 按位与 |
| 7 | ^IMG_256 | 按位异或和常规的“OR” |
| 8 | <= < > >= | 比较运算符 |
| 9 | <> == != | 等于运算符 |
| 10 | = %= /= //= -= += \*=\*\*= | 赋值运算符 |
| 11 | is is not | 身份运算符 |
| 12 | in not in | 成员运算符 |
| 13 | not or and | 逻辑运算符 |

实例：operators\_precedence.py

a = 20

b = 10

c = 15

d = 5

print ("a:%d b:%d c:%d d:%d" % (a,b,c,d ))

e = (a + b) \* c / d #( 30 \* 15 ) / 5

print ("Value of (a + b) \* c / d is ", e)

说明：如果优先级一样，则从左往右算。