# Python 运算符

## 什么是运算符？

本章节主要说明Python的运算符。举个简单的例子**4 +5 = 9**。 例子中，**4** 和 **5** 被称为**操作数**，"**+**" 称为运算符。

Python语言支持以下类型的运算符:

* [算术运算符](http://www.runoob.com/python/python-operators.html#ysf1)
* [比较（关系）运算符](http://www.runoob.com/python/python-operators.html#ysf2)
* [赋值运算符](http://www.runoob.com/python/python-operators.html#ysf3)
* [逻辑运算符](http://www.runoob.com/python/python-operators.html#ysf4)
* [位运算符](http://www.runoob.com/python/python-operators.html#ysf5)
* [成员运算符](http://www.runoob.com/python/python-operators.html#ysf6)
* [身份运算符](http://www.runoob.com/python/python-operators.html#ysf7)
* [运算符优先级](http://www.runoob.com/python/python-operators.html#ysf8)

接下来让我们一个个来学习Python的运算符。

## Python算术运算符

以下假设变量a为10，变量b为20：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| + | 加 - 两个对象相加 | a + b 输出结果 30 |
| - | 减 - 得到负数或是一个数减去另一个数 | a - b 输出结果 -10 |
| \* | 乘 - 两个数相乘或是返回一个被重复若干次的字符串 | a \* b 输出结果 200 |
| / | 除 - x除以y | b / a 输出结果 2 |
| % | 取模 - 返回除法的余数 | b % a 输出结果 0 |
| \*\* | 幂 - 返回x的y次幂 | a\*\*b 为10的20次方， 输出结果 100000000000000000000 |
| // | 取整除 - 返回商的整数部分 | 9//2 输出结果 4 , 9.0//2.0 输出结果 4.0 |

以下实例演示了Python所有算术运算符的操作：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

a = 21

b = 10

c = 0

c = a + b

print "1 - c 的值为：", c

c = a - b

print "2 - c 的值为：", c

c = a \* b

print "3 - c 的值为：", c

c = a / b

print "4 - c 的值为：", c

c = a % b

print "5 - c 的值为：", c

# 修改变量 a 、b 、c

a = 2

b = 3

c = a\*\*b

print "6 - c 的值为：", c

a = 10

b = 5

c = a//b

print "7 - c 的值为：", c

[**尝试一下 »**](http://www.runoob.com/try/showpy.php?filename=test_operator&language=py)

以上实例输出结果：

1 - c 的值为： 31

2 - c 的值为： 11

3 - c 的值为： 210

4 - c 的值为： 2

5 - c 的值为： 1

6 - c 的值为： 8

7 - c 的值为： 2

## Python比较运算符

以下假设变量a为10，变量b为20：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| == | 等于 - 比较对象是否相等 | (a == b) 返回 False。 |
| != | 不等于 - 比较两个对象是否不相等 | (a != b) 返回 true. |
| <> | 不等于 - 比较两个对象是否不相等 | (a <> b) 返回 true。这个运算符类似 != 。 |
| > | 大于 - 返回x是否大于y | (a > b) 返回 False。 |
| < | 小于 - 返回x是否小于y。所有比较运算符返回1表示真，返回0表示假。这分别与特殊的变量True和False等价。注意，这些变量名的大写。 | (a < b) 返回 true。 |
| >= | 大于等于 - 返回x是否大于等于y。 | (a >= b) 返回 False。 |
| <= | 小于等于 - 返回x是否小于等于y。 | (a <= b) 返回 true。 |

以下实例演示了Python所有比较运算符的操作：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

a = 21

b = 10

c = 0

if ( a == b ):

print "1 - a 等于 b"

else:

print "1 - a 不等于 b"

if ( a != b ):

print "2 - a 不等于 b"

else:

print "2 - a 等于 b"

if ( a <> b ):

print "3 - a 不等于 b"

else:

print "3 - a 等于 b"

if ( a < b ):

print "4 - a 小于 b"

else:

print "4 - a 大于等于 b"

if ( a > b ):

print "5 - a 大于 b"

else:

print "5 - a 小于等于 b"

# 修改变量 a 和 b 的值

a = 5;

b = 20;

if ( a <= b ):

print "6 - a 小于等于 b"

else:

print "6 - a 大于 b"

if ( b >= a ):

print "7 - b 大于等于 b"

else:

print "7 - b 小于 b"

以上实例输出结果：

1 - a 不等于 b

2 - a 不等于 b

3 - a 不等于 b

4 - a 大于等于 b

5 - a 大于 b

6 - a 小于等于 b

7 - b 大于等于 b

## Python赋值运算符

以下假设变量a为10，变量b为20：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| = | 简单的赋值运算符 | c = a + b 将 a + b 的运算结果赋值为 c |
| += | 加法赋值运算符 | c += a 等效于 c = c + a |
| -= | 减法赋值运算符 | c -= a 等效于 c = c - a |
| \*= | 乘法赋值运算符 | c \*= a 等效于 c = c \* a |
| /= | 除法赋值运算符 | c /= a 等效于 c = c / a |
| %= | 取模赋值运算符 | c %= a 等效于 c = c % a |
| \*\*= | 幂赋值运算符 | c \*\*= a 等效于 c = c \*\* a |
| //= | 取整除赋值运算符 | c //= a 等效于 c = c // a |

以下实例演示了Python所有赋值运算符的操作：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

a = 21

b = 10

c = 0

c = a + b

print "1 - c 的值为：", c

c += a

print "2 - c 的值为：", c

c \*= a

print "3 - c 的值为：", c

c /= a

print "4 - c 的值为：", c

c = 2

c %= a

print "5 - c 的值为：", c

c \*\*= a

print "6 - c 的值为：", c

c //= a

print "7 - c 的值为：", c

以上实例输出结果：

1 - c 的值为： 31

2 - c 的值为： 52

3 - c 的值为： 1092

4 - c 的值为： 52

5 - c 的值为： 2

6 - c 的值为： 2097152

7 - c 的值为： 99864

## Python位运算符

按位运算符是把数字看作二进制来进行计算的。Python中的按位运算法则如下：

下表中变量 a 为 60，b 为 13，二进制格式如下：

a = 0011 1100

b = 0000 1101

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

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| & | 按位与运算符：参与运算的两个值,如果两个相应位都为1,则该位的结果为1,否则为0 | (a & b) 输出结果 12 ，二进制解释： 0000 1100 |
| | | 按位或运算符：只要对应的二个二进位有一个为1时，结果位就为1。 | (a | b) 输出结果 61 ，二进制解释： 0011 1101 |
| ^ | 按位异或运算符：当两对应的二进位相异时，结果为1 | (a ^ b) 输出结果 49 ，二进制解释： 0011 0001 |
| ~ | 按位取反运算符：对数据的每个二进制位取反,即把1变为0,把0变为1 | (~a ) 输出结果 -61 ，二进制解释： 1100 0011， 在一个有符号二进制数的补码形式。 |
| << | 左移动运算符：运算数的各二进位全部左移若干位，由"<<"右边的数指定移动的位数，高位丢弃，低位补0。 | a << 2 输出结果 240 ，二进制解释： 1111 0000 |
| >> | 右移动运算符：把">>"左边的运算数的各二进位全部右移若干位，">>"右边的数指定移动的位数 | a >> 2 输出结果 15 ，二进制解释： 0000 1111 |

以下实例演示了Python所有位运算符的操作：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

a = 60 # 60 = 0011 1100

b = 13 # 13 = 0000 1101

c = 0

c = a & b; # 12 = 0000 1100

print "1 - c 的值为：", c

c = a | b; # 61 = 0011 1101

print "2 - c 的值为：", c

c = a ^ b; # 49 = 0011 0001

print "3 - c 的值为：", c

c = ~a; # -61 = 1100 0011

print "4 - c 的值为：", c

c = a << 2; # 240 = 1111 0000

print "5 - c 的值为：", c

c = a >> 2; # 15 = 0000 1111

print "6 - c 的值为：", c

以上实例输出结果：

1 - c 的值为： 12

2 - c 的值为： 61

3 - c 的值为： 49

4 - c 的值为： -61

5 - c 的值为： 240

6 - c 的值为： 15

## Python逻辑运算符

Python语言支持逻辑运算符，以下假设变量 a 为 10, b为 20:

|  |  |  |  |
| --- | --- | --- | --- |
| **运算符** | **逻辑表达式** | **描述** | **实例** |
| and | x and y | 布尔"与" - 如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。 | (a and b) 返回 20。 |
| or | x or y | 布尔"或" - 如果 x 是非 0，它返回 x 的值，否则它返回 y 的计算值。 | (a or b) 返回 10。 |
| not | not x | 布尔"非" - 如果 x 为 True，返回 False 。如果 x 为 False，它返回 True。 | not(a and b) 返回 False |

以上实例输出结果：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

a = 10

b = 20

if ( a and b ):

print "1 - 变量 a 和 b 都为 true"

else:

print "1 - 变量 a 和 b 有一个不为 true"

if ( a or b ):

print "2 - 变量 a 和 b 都为 true，或其中一个变量为 true"

else:

print "2 - 变量 a 和 b 都不为 true"

# 修改变量 a 的值

a = 0

if ( a and b ):

print "3 - 变量 a 和 b 都为 true"

else:

print "3 - 变量 a 和 b 有一个不为 true"

if ( a or b ):

print "4 - 变量 a 和 b 都为 true，或其中一个变量为 true"

else:

print "4 - 变量 a 和 b 都不为 true"

if not( a and b ):

print "5 - 变量 a 和 b 都为 false，或其中一个变量为 false"

else:

print "5 - 变量 a 和 b 都为 true"

以上实例输出结果：

1 - 变量 a 和 b 都为 true

2 - 变量 a 和 b 都为 true，或其中一个变量为 true

3 - 变量 a 和 b 有一个不为 true

4 - 变量 a 和 b 都为 true，或其中一个变量为 true

5 - 变量 a 和 b 都为 false，或其中一个变量为 false

## Python成员运算符

除了以上的一些运算符之外，Python还支持成员运算符，测试实例中包含了一系列的成员，包括字符串，列表或元组。

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| in | 如果在指定的序列中找到值返回 True，否则返回 False。 | x 在 y 序列中 , 如果 x 在 y 序列中返回 True。 |
| not in | 如果在指定的序列中没有找到值返回 True，否则返回 False。 | x 不在 y 序列中 , 如果 x 不在 y 序列中返回 True。 |

以下实例演示了Python所有成员运算符的操作：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

a = 10

b = 20

list = [1, 2, 3, 4, 5 ];

if ( a in list ):

print "1 - 变量 a 在给定的列表中 list 中"

else:

print "1 - 变量 a 不在给定的列表中 list 中"

if ( b not in list ):

print "2 - 变量 b 不在给定的列表中 list 中"

else:

print "2 - 变量 b 在给定的列表中 list 中"

# 修改变量 a 的值

a = 2

if ( a in list ):

print "3 - 变量 a 在给定的列表中 list 中"

else:

print "3 - 变量 a 不在给定的列表中 list 中"

以上实例输出结果：

1 - 变量 a 不在给定的列表中 list 中

2 - 变量 b 不在给定的列表中 list 中

3 - 变量 a 在给定的列表中 list 中

## Python身份运算符

身份运算符用于比较两个对象的存储单元

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **实例** |
| is | is是判断两个标识符是不是引用自一个对象 | x is y, 如果 id(x) 等于 id(y) , **is** 返回结果 1 |
| is not | is not是判断两个标识符是不是引用自不同对象 | x is not y, 如果 id(x) 不等于 id(y). **is not** 返回结果 1 |

以下实例演示了Python所有身份运算符的操作：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

a = 20

b = 20

if ( a is b ):

print "1 - a 和 b 有相同的标识"

else:

print "1 - a 和 b 没有相同的标识"

if ( id(a) == id(b) ):

print "2 - a 和 b 有相同的标识"

else:

print "2 - a 和 b 没有相同的标识"

# 修改变量 b 的值

b = 30

if ( a is b ):

print "3 - a 和 b 有相同的标识"

else:

print "3 - a 和 b 没有相同的标识"

if ( a is not b ):

print "4 - a 和 b 没有相同的标识"

else:

print "4 - a 和 b 有相同的标识"

以上实例输出结果：

1 - a 和 b 有相同的标识

2 - a 和 b 有相同的标识

3 - a 和 b 没有相同的标识

4 - a 和 b 没有相同的标识

## Python运算符优先级

以下表格列出了从最高到最低优先级的所有运算符：

|  |  |
| --- | --- |
| **运算符** | **描述** |
| \*\* | 指数 (最高优先级) |
| ~ + - | 按位翻转, 一元加号和减号 (最后两个的方法名为 +@ 和 -@) |
| \* / % // | 乘，除，取模和取整除 |
| + - | 加法减法 |
| >> << | 右移，左移运算符 |
| & | 位 'AND' |
| ^ | | 位运算符 |
| <= < > >= | 比较运算符 |
| <> == != | 等于运算符 |
| = %= /= //= -= += \*= \*\*= | 赋值运算符 |
| is is not | 身份运算符 |
| in not in | 成员运算符 |
| not or and | 逻辑运算符 |

以下实例演示了Python所有运算符优先级的操作：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

a = 20

b = 10

c = 15

d = 5

e = 0

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

print "(a + b) \* c / d 运算结果为：", e

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

print "((a + b) \* c) / d 运算结果为：", e

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

print "(a + b) \* (c / d) 运算结果为：", e

e = a + (b \* c) / d; # 20 + (150/5)

print "a + (b \* c) / d 运算结果为：", e

以上实例输出结果：

(a + b) \* c / d 运算结果为： 90

((a + b) \* c) / d 运算结果为： 90

(a + b) \* (c / d) 运算结果为： 90

a + (b \* c) / d 运算结果为： 50