运算符与表达式

1. 运算符

我们将简单浏览一下运算符和它们的用法：

技巧

你可以交互地使用解释器来计算例子中给出的表达式。例如，为了测试表达式 `2 + 3`，使用交互式的带提示符的 Python 解释器：

```
>>> 2 + 3
5
>>> 3 * 5
15
>>>
```

运算符	名称	说明	例子
+	加	两个对象相加	3 + 5 得到 8。'a' + 'b'得到'ab'。
-	减	得到负数或是一个数减去另一个数	-5.2 得到一个负数。50 - 24 得到 26。
*	乘	两个数相乘或是返回一个被重复若干次的字符串	*2 3 得到 6。'la'* 3 得到'lalala'。
**	幂	返回 x 的 y 次幂	*3 * 4 得到 81（即 3* 3 *3* 3）
/	除	x 除以 y	4/3 得到 1（整数的除法得到整数结果）。4.0/3 或 4/3.0 得到 1.3333333333333333
//	取整除	返回商的整数部分	4 // 3.0 得到 1.0
%	取模	返回除法的余数	8%3 得到 2。-25.5%2.25 得到 1.5
<<	左移	把一个数的比特向左移一定数目（每个数在内存中都表示为比特或二进制数字，即 0 和 1）	2 << 2 得到 8。——2 按比特表示为 10

运算符	名称	说明	例子
>>	右移	把一个数的比特向右移一定数目	11 >> 1 得到 5。——11 按比特表示为 1011，向右移动 1 比特后得到 101，即十进制的 5。
&	按位与	数的按位与	5 & 3 得到 1。
|	按位或	数的按位或	5 | 3 得到 7。
^	按位异或	数的按位异或	5 ^ 3 得到 6
~	按位翻转	x 的按位翻转是-(x+1)	~5 得到 6。
<	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。	5 < 3 返回 0（即 False）而 3 < 5 返回 1（即 True）。比较可以被任意连接：3 < 5 < 7 返回 True。
>	大于	返回 x 是否大于 y	5 > 3 返回 True。如果两个操作数都是数字，它们首先被转换为一个共同的类型。否则，它总是返回 False。
<=	小于等于	返回 x 是否小于等于 y	x = 3; y = 6; x <= y 返回 True。
>=	大于等于	返回 x 是否大于等于 y	x = 4; y = 3; x >= y 返回 True。
==	等于	比较对象是否相等	x = 2; y = 2; x == y 返回 True。x = 'str'; y = 'stR'; x == y 返回 False。x = 'str'; y = 'str'; x == y 返回 True。
!=	不等于	比较两个对象是否不相等	x = 2; y = 3; x != y 返回 True。
not	布尔“非”	如果 x 为 True，返回 False。如果 x 为 False，它返回 True。	x = True; not y 返回 False。

运算符	名称	说明	例子
and	布尔“与”	如果 x 为 False，x and y 返回 False，否则它返回 y 的计算值。	x = False; y = True; x and y，由于 x 是 False，返回 False。在这里，Python 不会计算 y，因为它知道这个表达式的值肯定是 False（因为 x 是 False）。这个现象称为短路计算。
or	布尔“或”	如果 x 是 True，它返回 True，否则它返回 y 的计算值。	x = True; y = False; x or y 返回 True。短路计算在这里也适用。

2. 运算符优先级

如果你有一个如 `2 + 3 * 4` 那样的表达式，是先做加法呢，还是先做乘法？我们的中学数学告诉我们应当先做乘法——这意味着乘法运算符的优先级高于加法运算符。
下面这个表给出 Python 的运算符优先级，从最低的优先级（最松散地结合）到最高的优先级（最紧密地结合）。这意味着在一个表达式中，Python 会首先计算表中较下面的运算符，然后在计算列在表上部的运算符。

下面这张表（与 Python 参考手册中的那个表一模一样）已经顾及了完整的需要。事实上，我建议你使用圆括号来分组运算符和操作数，以便能够明确地指出运算的先后顺序，使程序尽可能地易读。例如，`2 + (3 * 4)`显然比 `2 + 3 * 4` 清晰。与此同时，圆括号也应该正确使用，而不应该用得过滥（比如 `2 + (3 + 4)`）。

运算符	描述
lambda	Lambda 表达式
or	布尔“或”
and	布尔“与”
not x	布尔“非”
in，not in	成员测试
is，is not	同一性测试

运算符	描述
<，<=，>，>=，!=，==	比较
|	按位或
^	按位异或
&	按位与
<<，>>	移位
+，-	加法与减法
*，/，%	乘法、除法与取余
+x，-x	正负号
~x	按位翻转
**	指数
x.attribute	属性参考
x[index]	下标
x[index:index]	寻址段
f(arguments...)	函数调用
(experession,...)	绑定或元组显示
[expression,...]	列表显示
{key:datum,...}	字典显示
'expression,...'	字符串转换

其中我们还没有接触过的运算符将在后面的章节中介绍。

在表中列在同一行的运算符具有 相同优先级 。例如，+和-有相同的优先级。
默认地，运算符优先级表决定了哪个运算符在别的运算符之前计算。然而，如果你想要改变它们的计算顺序，你得使用圆括号。例如，你想要在一个表达式中让加法在乘法之前计算，那么你就得写成类似(2 + 3) * 4 的样子。
运算符通常由左向右结合，即具有相同优先级的运算符按照从左向右的顺序计算。例如，2 + 3 + 4 被计算成(2 + 3) + 4。一些如赋值运算符那样的运算符是由右向左结合的，即 a = b = c 被处理为 a = (b = c)。

3. 表达式

```
#!/usr/bin/python
# Filename: expression.py

length = 5
breadth = 2
area = length * breadth
print 'Area is', area
print 'Perimeter is', 2 * (length + breadth)
```

输出

```
$ python expression.py
Area is 10
Perimeter is 14
```

它如何工作

矩形的长度与宽度存储在以它们命名的变量中。我们借助表达式使用它们计算矩形的面积和边长。我们表达式 length * breadth 的结果存储在变量 area 中，然后用 print 语句打印。在另一个打印语句中，我们直接使用表达式 2 * (length + breadth)的值。
另外，注意 Python 如何打印“漂亮的”输出。尽管我们没有在'Area is'和变量 area 之间指定空格，Python 自动在那里放了一个空格，这样我们就可以得到一个清晰漂亮的输出，而程序也变得更加易读（因为我们不需要担心输出之间的空格问题）。这是 Python 如何使程序员的生活变得更加轻松的一个例子。