第四章：运算符与表达式

Table of Contents

[1. 序言 2](#_Toc50389112)

[1.1 章节简介 2](#_Toc50389113)

[2. 算术运算符 3](#_Toc50389114)

[2.1 加法运算符 3](#_Toc50389115)

[2.2 减法运算符 3](#_Toc50389116)

[2.3 乘法运算符 4](#_Toc50389117)

[2.4 除法运算符 4](#_Toc50389118)

[3. 赋值运算符 5](#_Toc50389119)

[4. 比较(关系)运算符 5](#_Toc50389120)

[5. 逻辑运算符 7](#_Toc50389121)

[6. 位运算符 8](#_Toc50389122)

[6.1 位与运算符& 8](#_Toc50389123)

[6.2 位或运算符| 8](#_Toc50389124)

[6.3 位异或运算符^ 8](#_Toc50389125)

[6.4 左移位运算符 << 9](#_Toc50389126)

[6.5 右移位运算符 >> 9](#_Toc50389127)

[7. 成员运算符 9](#_Toc50389128)

[8. 身份运算符 10](#_Toc50389129)

[9. 运算符的优先级 11](#_Toc50389130)

[10. 字符串的格式化 12](#_Toc50389131)

[10.1 用%实现字符串格式化 12](#_Toc50389132)

[10.2 用format()方法实现字符串格式化 12](#_Toc50389133)

# 序言

## 章节简介

这一章节帮助我们了解Python这门语言的基础语法结构，目标是在阐明Python代码的结构的同时，又不过太过纠缠于特殊的规则或者细节。 举个简单的例子a=4+3这个表达式，在Python中的意思是我们先计算等号右边的加法公式，将得到的结果(7)赋值给变量a，那么a的值即为7，这里的等号(=)是一个赋值的运算，而不是我们在数学公式里常用表示两个代数表达式相等的意思。和其他大多数的语言一样，python 中常用的操作符也有算术操作符、比较操作符、逻辑操作符，但是又有一些差别，下面详细介绍：

a. 算术运算符

b. 赋值运算符

c. 比较(关系)运算符

d. 逻辑运算符

e. 位运算符

f. 成员运算符

g. 身份运算符

h. 字符串的格式化

# 算术运算符

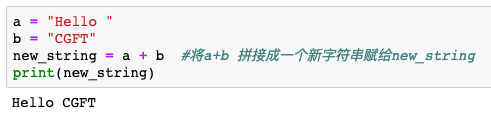
和其他大多数的语言一样，python 也有 +（加）、-（减）、\*（乘）、/（除）、%（取余）这 5 个运算符，除此之外还有两个特殊的运算符，分别是 //（整除）和 \*\*（幂运算符，或叫乘方运算符）。我们通过一些实例来了解.

## 加法运算符

加法运算符很简单，和数学中的规则一样，请看下面的代码:

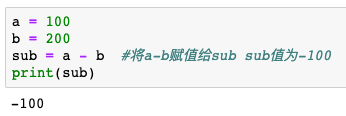


当+用于数字时，表示将两个数字相加。当+用于字符串时，它还有拼接字符串的作用，如下所示：



## 减法运算符

减法运算和数学中的规则相同：

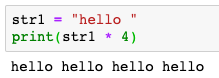


## 乘法运算符

乘法运算和数学中的规则相同：



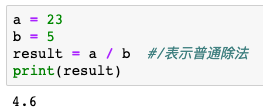
\*除了可以用作乘法运算，还可以用来重复字符串，也即将 n 个同样的字符串连接起来:



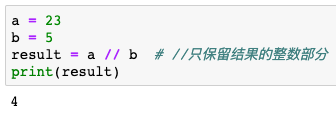
## 除法运算符

跟除法相关的运算符有三个/、//和%，下面分别介绍。

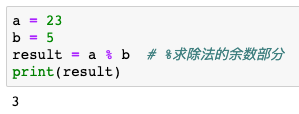
/表示普通除法，使用它计算出来的结果和数学中的计算结果相同。



//表示整除，只保留结果的整数部分，舍弃小数部分；注意是直接丢掉小数部分，而不是四舍五入。



%是求余运算符，计算两数相除的余数：



两个乘号\*\*是幂运算符：



# 赋值运算符

赋值运算符是一个等号(=)，我们在前面的学习已经多次见到，表示的意义是将赋值运算符右侧的复杂表达式进行运算处理，将结果赋值给左边的变量。

# 比较(关系)运算符

用于对变量或表达式的结果进行大小、真假等比较。比较的结果为真，则返回True，为假，则返回False。通常用在条件语句中作为判断的依据。

==运算符将两个对象进行比较，判断它们是否相等：



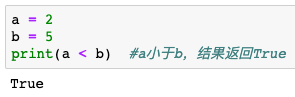
!=判断两个对象是否不相等：



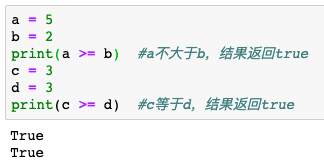
>判断等式左边的对象是否大于右边：



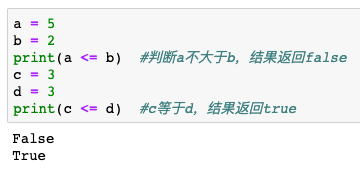
>判断等式左边的对象是否小于右边：



>=判断等式左边的对象是否大于或等于右边：



<=判断等式左边的对象是否小于或等于右边：



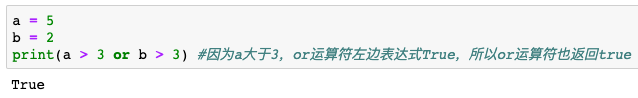
# 逻辑运算符

对布尔值的对象进行逻辑上的与或非运算。

and表示的是逻辑与的运算，当and两边的表达式都为True时，该运算符结果返回True：



or表示的是逻辑或的运算，当or两边的表达式都为False时，该运算符返回False；否则返回True：



not表示逻辑非运算，条件为真，结果为假；条件为假，结果为真：

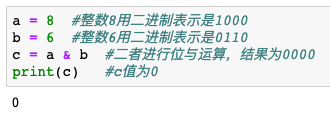


# 位运算符

我们的变量在计算机中实际是以二进制的方式进行保存的，比如整数7，用二进制来表示是111(2^2+2^1+2^0)，而位运算符就是把操作数按二进制表示以后进行计算，我们一次举例：

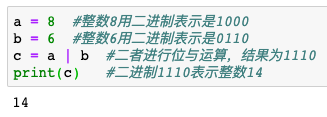
## 位与运算符&

位与运算(&)：两个操作数按二进制数表示，对应位都为1，结果位才为1。



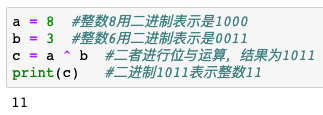
## 位或运算符|

位与运算(|)：两个操作数按二进制数表示，对应位有一个为1，结果位即为1。



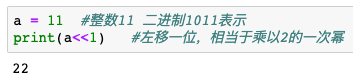
## 位异或运算符^

位异或运算(^)：两个操作数按二进制数表示，对应位同为1或同为0，结果为0，否则为1。



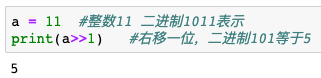
## 左移位运算符 <<

左移位运算符(<<)：把二进制操作数，向左移动相应位数，左边最高位溢出时被丢弃，右边空位用0补齐（左移位相当于乘以2的n次幂）



## 右移位运算符 >>

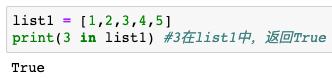
右移位运算符(>>)：把二进制操作数，向右移动相应位数，右边溢出位被丢弃，左边最高位如果是0补0，是1补1（右移位相当于除以2的n次幂）



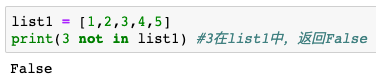
# 成员运算符

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in, 如果在指定的序列中找到值返回 True，否则返回 False:



运算符not in, 如果在指定的序列中找到值返回False，否则返回True:

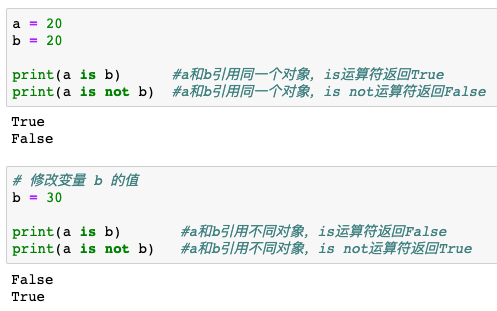


# 身份运算符

身份运算符用于比较两个对象的存储单元。

**is**运算符：is 是判断两个标识符是不是引用自一个对象(关于什么是对象及对象的引用我们在第三章里已经介绍过)，如果是返回True否则返回False。x is y, 类似 id(x) == id(y) , 如果引用的是同一个对象则返回 True，否则返回 False。//注：id() 函数用于获取变量引用的对象的内存地址。

**is not**运算符：is not 是判断两个标识符是不是引用自不同对象。x is not y ， 类似 id(a) != id(b)。如果引用的不是同一个对象则返回结果 True，否则返回 False。



# 运算符的优先级

以下表格列出了从最高到最低优先级的所有运算符，在程序实际运行时，Python会按照运算符的优先级的不同从高到底，先计算最高优先级的运算符，再进行低优先级的运算符的运算。

|  |  |
| --- | --- |
| \*\* | 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 and or | 逻辑运算符 |

这么复杂的优先级难以记住怎么办？其实不必担心，当记不住运算符之间的比较优先级时，同时也为了增强代码的可读性，我们还可以通过使用()来明确告诉Python，先计算()里的表达式，再进行后续的计算。



# 字符串的格式化

最后一个常见的问题是如何输出格式化的字符串。我们经常会输出类似“*亲爱的xxx你好！你xx月的话费是xx，余额是xx*”之类的字符串，而xxx的内容都是根据变量变化的，所以，需要一种简便的格式化字符串的方式。

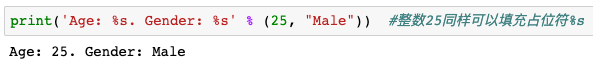
## 用%实现字符串格式化

在Python中，采用的格式化方式和C语言是一致的，用%实现，举例如下：



是否看出了格式化的规则？ %运算符就是用来格式化字符串的。在字符串内部，%s表示用字符串替换，%d表示用整数替换，有几个%?占位符，后面就跟几个变量或者值，顺序要对应好。如果只有一个%?，括号可以省略。

常见的占位符有%d(整数替换)、%f(浮点数替换)、%s(字符串替换)。如果不太确定用哪种类型的占位符，可以使用%s，%s会把任何数据类型转换为字符串：



## 用format()方法实现字符串格式化

还有另一种格式化的方式，我们使用字符串的format()方法，它会用传入的参数依次替换字符串内的占位符{0}、{1}……，具体实例如下：

