

6

Basic Calculations in Python

Python 常见运算

从加减乘除开始学运算符



有时人们不想听到真相，因为他们不想打碎自己的幻象。

Sometimes people don't want to hear the truth because they don't want their illusions destroyed.

—— 弗里德里希·尼采 (Friedrich Nietzsche) | 德国哲学家 | 1844 ~ 1900



```
▶ XXXXX
▶ XXXXX
▶ XXXXX
▶ XXXXX
▶ XXXXX
▶
```



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：jiang.visualize.ml@gmail.com

6.1 几类运算符

Python 中的运算符可以分为以下几类：

- ▶ 算术运算符：用于数学运算，例如加法 (+)、减法 (-)、乘法 (*)、除法 (/)、取余数 (%)、乘幂 (**) 等。
- ▶ 比较运算符：用于比较两个值之间的关系，例如等于 (==)、不等于 (!=)、大于 (>)、小于 (<)、大于等于 (>=)、小于等于 (<=) 等。
- ▶ 逻辑运算符：用于处理布尔型数据，例如与 (and)、或 (or)、非 (not) 等。
- ▶ 赋值运算符：用于给变量赋值，例如等号 (=)、自加运算 (+=)、自减运算 (-=)、自乘运算 (*=)、自除运算 (/=)。
- ▶ 成员运算符：用于检查一个值是否为另一个值的成员，例如 in、not in 等。
- ▶ 身份运算符：用于检查两个变量是否引用同一个对象，例如 is、is not 等。

以上是 Python 中常见的运算符，可以根据不同的场景选择合适的运算符进行操作。

Arithmetic operators			Logical operators		
+		%	==	!=	and
x	/	**	>	<=	or
-		//	<	>=	not

Bitwise operators	Membership operators	Identity operators
&	in	is
~	not in	is not
^		

Assignment operators						
+=	-=	*=	/=	%=	**=	//=

图 1. 常用运算符

6.2 算术运算符

Python 算术运算符用于数学运算，包括加法、减法、乘法、除法、取模和幂运算等。下面分别介绍这些算术运算符及其使用方法。

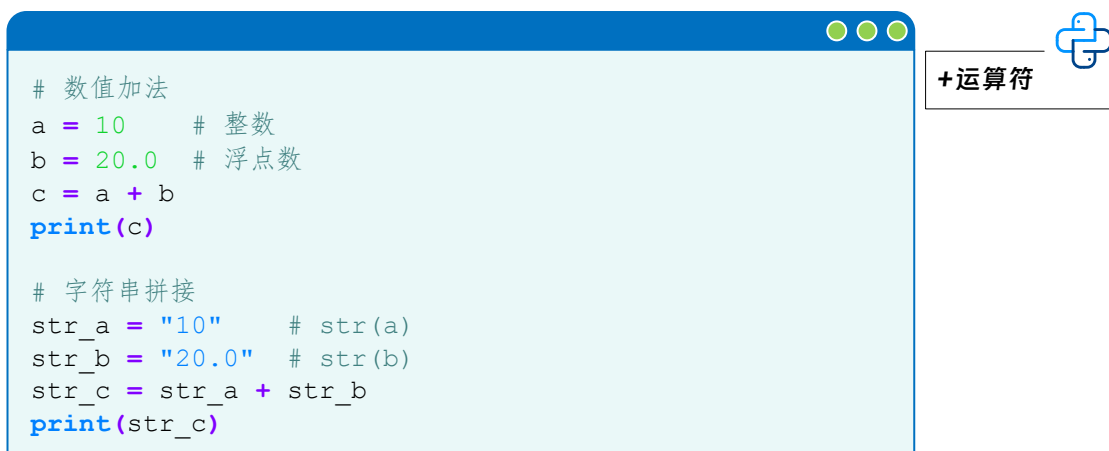
加减法

加法运算符 (+) 用于将两个数值相加或将两个字符串拼接起来。

请大家在 JupyterLab 中自行练习图2。

注意，当进行加法运算时，如果操作数的类型不一致，Python 会自动进行类型转换。如果一个数是整数，而另一个是浮点数，则整数会被转换为浮点数，然后进行加法运算。运算结果为浮点数。加法时，如果一个数是整数，而另一个是复数，则整数会被转换为复数，然后进行加法运算。结果为复数。如果一个操作数是浮点数，而另一个是复数，则浮点数会被转换为复数，然后进行加法运算。运算结果为复数。

减法运算符 - 用于将两个数值相减，不支持字符串运算，错误信息为 `TypeError: unsupported operand type(s) for -: 'str' and 'str'`。



```
# 数值加法
a = 10      # 整数
b = 20.0    # 浮点数
c = a + b
print(c)

# 字符串拼接
str_a = "10"    # str(a)
str_b = "20.0"  # str(b)
str_c = str_a + str_b
print(str_c)
```

+ 运算符

图 2. 加法

乘除法

乘法运算符 (*) 用于将两个数值相乘或将一个字符串重复多次。

注意，NumPy 数组完成矩阵乘法 (matrix multiplication) 时用的运算符为 @。

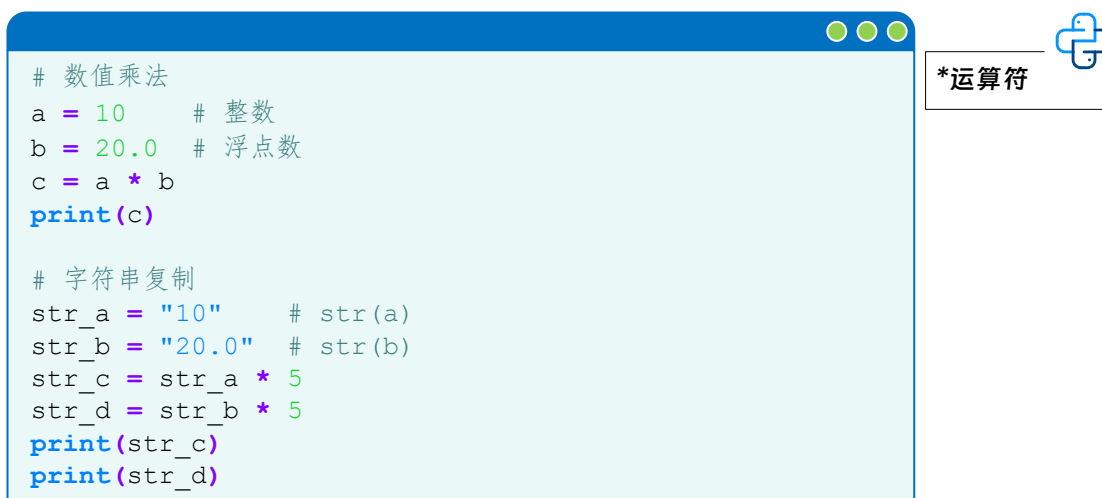


图 3. 乘法

除法运算符 / 用于将两个数值相除，结果为浮点数。

在 Python 中，正斜杠 / (forward slash) 和反斜杠 \ (backward slash) 具有不同的用途和含义。在路径表示中，正斜杠 / 用作目录分隔符，用于表示文件系统路径。在除法运算中，正斜杠用作除法操作符。

在 Windows 文件路径表示中，反斜杠用作目录分隔符。在字符串中，反斜杠 \ 用作转义字符，用于表示特殊字符或字符序列，比如：

- ▶ \n 换行符，将光标位置移到下一行开头。
- ▶ \r 回车符，将光标位置移到本行开头。
- ▶ \t 水平制表符，也即 Tab 键，一般相当于四个空格。
- ▶ \\ 反斜杠；在使用反斜杠作为转义字符时，为了表示反斜杠本身，需要使用两个连续的反斜杠\\。
- ▶ \' 单引号
- ▶ \" 双引号
- ▶ \ 在字符串行尾的续行符，即一行未完，转到下一行继续写。

取模运算符 % 用于获取两个数值相除的余数，比如 $10 \% 3$ 的结果为 1。幂运算符 ** 用于将一个数值的幂次方，比如 $2^{**}3$ 的结果为 8。



什么是转义字符？

转义字符是一种在字符串中使用的特殊字符序列，以反斜杠 \ 开头。在 Python 中，转义字符用于表示一些特殊字符、控制字符或无法直接输入的字符。通过使用转义字符，我们可以在字符串中插入换行符、制表符、引号等特殊字符。

括号

在 Python 中，运算符有不同的优先级。有时我们需要改变运算符的优先级顺序，可以使用圆括号 (parentheses) 来改变它们的顺序。圆括号可以用于明确指定某些运算的执行顺序，确保它们在其他运算之前或之后进行。

请大家自行比较下两例：

```
result = 2 + 3 * 4
result = (2 + 3) * 4
```

根据运算符的优先级规则，乘法运算 `*` 具有更高的优先级，因此先执行乘法，然后再进行加法。所以结果是 14。如果我们想先执行加法运算，然后再进行乘法运算，可以使用圆括号来改变优先级。

6.3 比较运算符

Python 比较运算符用于比较两个值，结果为 True 或 False。

相等、不等

相等运算符 `==` 比较两个值是否相等，返回 True 或 False。不等运算符 `!=` 比较两个值是否不相等，返回 True 或 False。

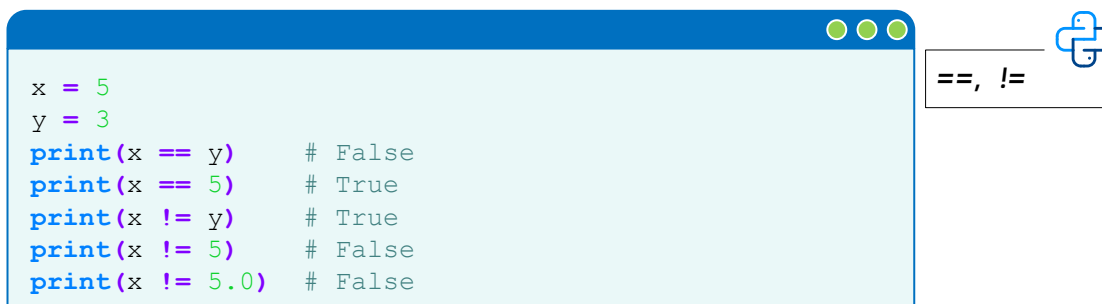


图 4. 相等、不等

大于、大于等于

大于运算符 `>` 比较左边的值是否大于右边的值，返回 True 或 False。大于等于运算符 `>=` 比较左边的值是否大于等于右边的值，返回 True 或 False。

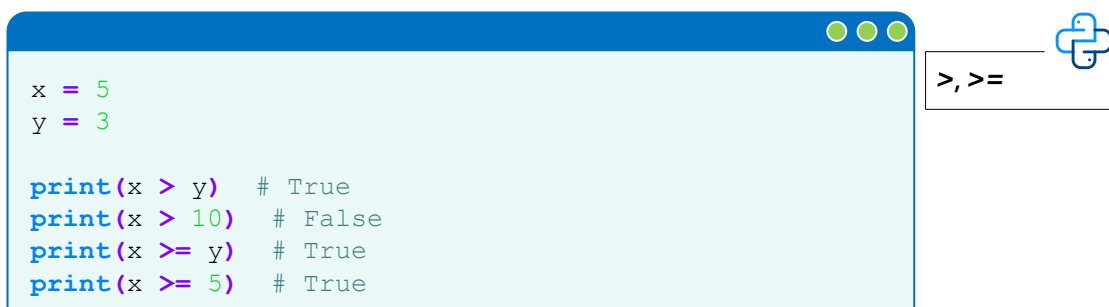


图 5. 大于、大于等于

小于、小于等于

小于运算符 < 比较左边的值是否小于右边的值，返回 True 或 False。小于等于运算符 <= 比较左边的值是否小于等于右边的值，返回 True 或 False。

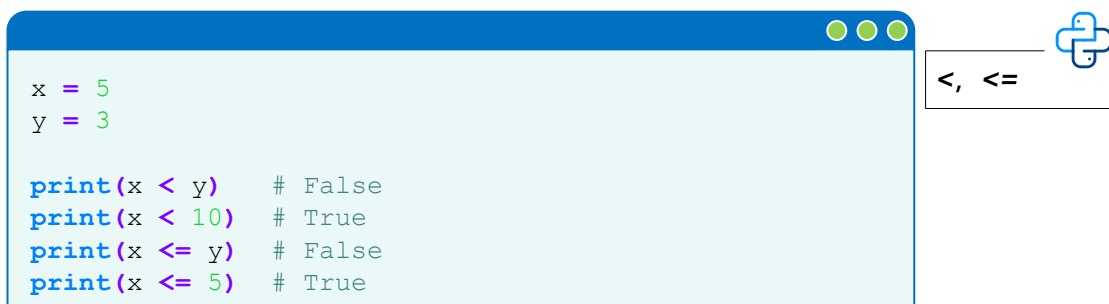


图 6. 小于、小于等于

6.4 逻辑运算符

Python 中有三种逻辑运算符，分别为 and、or 和 not，这些逻辑运算符可用于布尔类型的操作数上。这三种逻辑运算符实际上体现的是真值表 (truth table) 的逻辑。

如图 7 所示，真值表是一个逻辑表格，用于列出逻辑表达式的所有可能的输入组合和对应的输出结果。它展示了在不同的输入情况下，逻辑表达式的真值 True 或假值 False。下面对每种逻辑运算符进行详细的讲解。

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

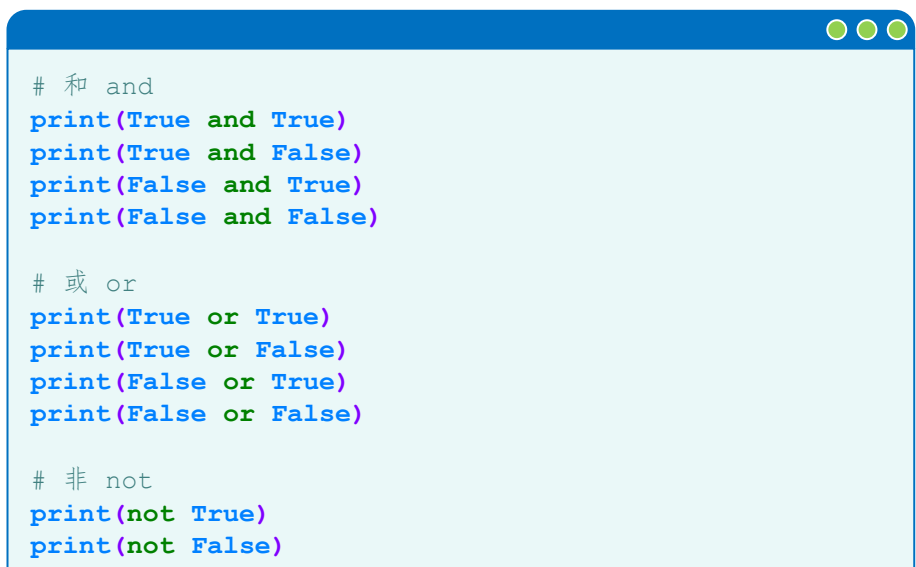
A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

A	not A
True	False
False	True

图 7. 真值表

和运算符 `and` 当左右两边的操作数都为 `True` 时，返回 `True`，否则返回 `False`。或运算符 `or` 当左右两边的操作数至少有一个为 `True` 时，返回 `True`，否则返回 `False`。取非运算符 `not` 对一个布尔类型的操作数取反，如果操作数为 `True`，返回 `False`，否则返回 `True`。请大家在 JupyterLab 自行练习图 8。


逻辑运算符常用于条件判断、循环控制等语句中。通过组合不同的逻辑运算符，可以实现复杂的逻辑表达式。



```
# 和 and
print(True and True)
print(True and False)
print(False and True)
print(False and False)

# 或 or
print(True or True)
print(True or False)
print(False or True)
print(False or False)

# 非 not
print(not True)
print(not False)
```



逻辑运算符

图 8. 逻辑运算符

6.5 赋值运算符

Python 中的赋值运算符用于将值分配给变量，下面逐一讲解。

等号 `=` 将右侧的值赋给左侧的变量。加等于 `+=` 将右侧的值加到左侧的变量上，并将结果赋给左侧的变量。

减等于 `-=` 将右侧的值从左侧的变量中减去，并将结果赋给左侧的变量。

乘等于 `*=` 将右侧的值乘以左侧的变量，并将结果赋给左侧的变量。

除等于 `/=` 将左侧的变量除以右侧的值，并将结果赋给左侧的变量。

取模等于 `%=` 将左侧的变量对右侧的值取模，并将结果赋给左侧的变量。

幂等于 `**=` 将左侧的变量的值提高到右侧的值的幂，并将结果赋给左侧的变量。

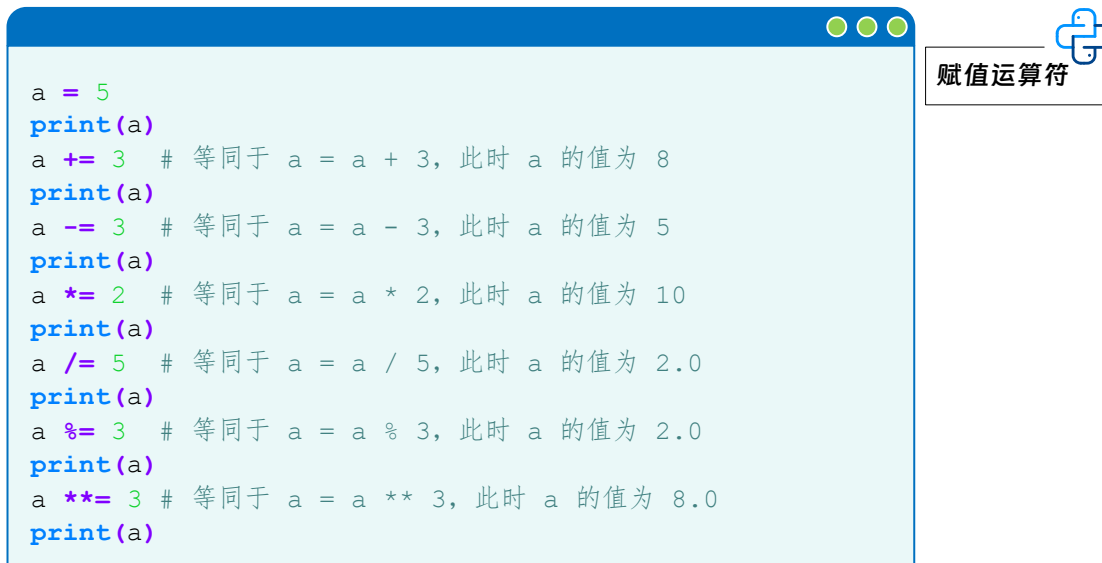


图 9. 赋值运算

6.6 成员运算符

Python 中成员运算符用于测试是否存在于序列中。共有两个成员运算符：a) in：如果在序列中找到值，返回 True，否则返回 False。b) not in：如果在序列中没有找到值，返回 True，否则返回 False。

图 10 是成员运算符的示例代码，请大家在 JupyterLab 中自行练习。

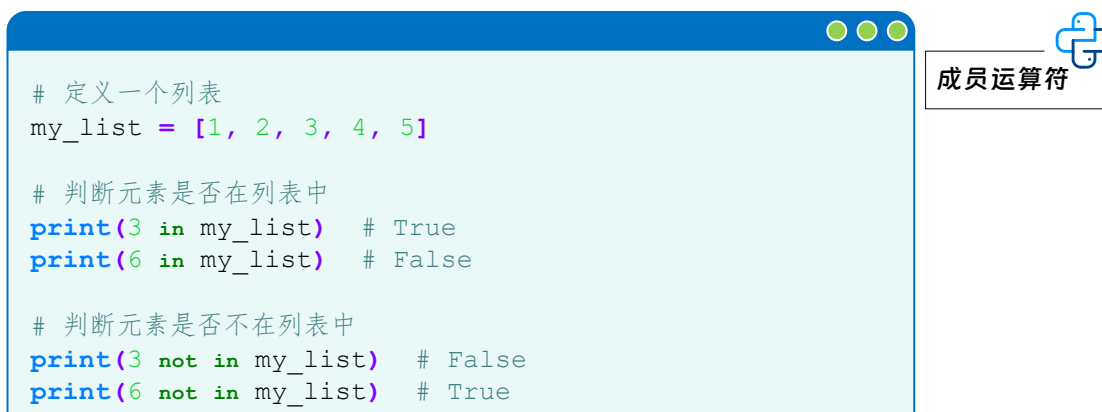


图 10. 成员运算

6.7 身份运算符

Python 身份运算符包括 `is` 和 `is not`，用于判断两个对象是否引用同一个内存地址。请大家回顾上一章介绍的视图、浅复制、深复制这三个概念。简单来说，浅复制只复制对象的一层内容，不涉及到嵌套的可变对象。深复制创建一个全新的对象，并递归地复制原始对象及其嵌套的可变对象。每个对象的副本都是独立的，修改原始对象或其嵌套对象不会影响深复制的对象。深复制涉及到多层嵌套的可变对象，确保每个对象都被复制。

请大家自行练习图 11 给出代码。

```
import copy
a = [1, 2, 3]
b = a
# 视图 b 引用 a 的内存地址
c = [1, 2, 3]
d = a.copy()

print(a is b)
# 输出 True, 因为 a 和 b 引用同一个内存地址
print(a is not c)
# 输出 True, 因为 a 和 c 引用不同的内存地址
print(a == c)
# 输出 True, 因为 a 和 c 的值相等
print(a is not d)
# 输出 True, 因为 a 和 d 引用不同的内存地址
print(a == d)
# 输出 True, 因为 a 和 d 的值相等

a_2_layers = [1, 2, [3, 4]]
d_2_layers = a_2_layers.copy()
e_2_layers = copy.deepcopy(a_2_layers)

print(a_2_layers is d_2_layers)
print(a_2_layers[2] is d_2_layers[2]) # 请特别关注

print(a_2_layers is e_2_layers)
print(a_2_layers[2] is e_2_layers[2])
```

身份运算符

图 11. 身份运算

6.8 优先级

在 Python 中，不同类型的运算符优先级是不同的，当一个表达式中有多个运算符时，会按照优先级的顺序依次计算，可以使用括号改变运算顺序。下面是 Python 中常见的运算符优先级列表，从高到低排列：

- ▶ 括号运算符：(), 用于改变运算顺序。
- ▶ 正负号运算符：+x, -x, 用于对数字取正负。
- ▶ 算术运算符：**, *, /, //, %, 用于数字的算术运算。
- ▶ 位运算符：~, &, |, ^, <<, >>, 用于二进制位的运算。
- ▶ 比较运算符：<, <=, >, >=, ==, !=, 用于比较大小关系。
- ▶ 身份运算符：is, is not, 用于判断两个对象是否相同。
- ▶ 成员运算符：in, not in, 用于判断一个元素是否属于一个集合。
- ▶ 逻辑运算符：not, and, or, 用于逻辑运算。

这部分我们不再展开介绍，如果后续用到的话，请大家自行学习。



什么是位运算符？

Python 提供了一组位运算符 (bitwise operator)，用于在二进制级别对整数进行操作。这些位运算符将整数的二进制表示作为操作数，并对每个位进行逻辑运算。



请大家完成下面 1 道题目。

Q1. 本章的唯一的题目就是请大家在 JupyterLab 中练习本章正文给出的示例代码。

* 不提供答案。