

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



- ◀ + 算术运算符，加法；将两个数值相加或连接两个字符串
- ◀ - 算术运算符，减法；从一个数值中减去另一个数值
- ◀ * 算术运算符，乘法；将两个数值相乘
- ◀ / 算术运算符，除法；将一个数值除以另一个数值，得到浮点数结果
- ◀ % 算术运算符，取余数；计算两个数相除后的余数
- ◀ ** 算术运算符，乘幂；将一个数值的指数幂次方
- ◀ == 比较运算符，等于；判断两个值是否相等，返回一个布尔值 (True 或 False)
- ◀ != 比较运算符，不等于；判断两个值是否不相等，返回一个布尔值 (True 或 False)
- ◀ > 比较运算符，大于；判断左边的值是否大于右边的值，返回一个布尔值 (True 或 False)
- ◀ < 比较运算符，小于；判断左边的值是否小于右边的值，返回一个布尔值 (True 或 False)
- ◀ >= 比较运算符，大于等于；判断左边的值是否大于或等于右边的值，返回一个布尔值 (True 或 False)
- ◀ <= 比较运算符，小于等于；判断左边的值是否小于或等于右边的值，返回一个布尔值 (True 或 False)
- ◀ and 逻辑运算符，与；判断两个条件是否同时为真，如果两个条件都为真，则返回 True；否则返回 False
- ◀ or 逻辑运算符，或；判断两个条件是否有一个为真，如果至少有一个条件为真，返回 True；否则返回 False
- ◀ not 逻辑运算符，非；对一个条件进行取反，如果条件为真，则返回 False；如果条件为假，则返回 True
- ◀ = 赋值运算符，等于；将等号右侧的值赋给左侧的变量，即将右侧的值存储到左侧的变量中
- ◀ += 赋值运算符，自加运算；将变量与右侧的值相加，并将结果赋值给该变量，例如，`a += b` 等价于 `a = a + b`
- ◀ -= 赋值运算符，自减运算；将变量与右侧的值相减，并将结果赋值给该变量，例如，`a -= b` 等价于 `a = a - b`
- ◀ *= 赋值运算符，自乘运算；将变量与右侧的值相乘，并将结果赋值给该变量，例如，`a *= b` 等价于 `a = a * b`
- ◀ /= 赋值运算符，自除运算；将变量与右侧的值相除，并将结果赋值给该变量，例如，`a /= b` 等价于 `a = a / b`
- ◀ in 成员运算符；检查某个值是否存在于指定的序列（如列表、元组、字符串等）中，如果存在则返回 True，否则返回 False
- ◀ not in 成员运算符；检查某个值是否不存在于指定的序列（如列表、元组、字符串等）中，如果不存在则返回 True，否则返回 False。
- ◀ is 身份运算符；检查两个变量是否引用同一个对象，如果是则返回 True，否则返回 False
- ◀ is not 身份运算符；检查两个变量是否不引用同一个对象，如果不是则返回 True，否则返回 False



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

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

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。

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'`。

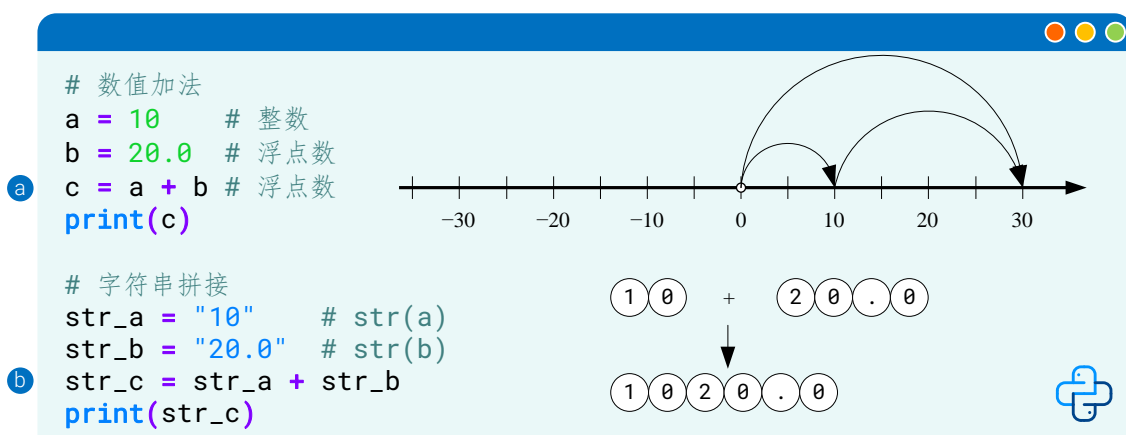


图 2. 加法

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 @。

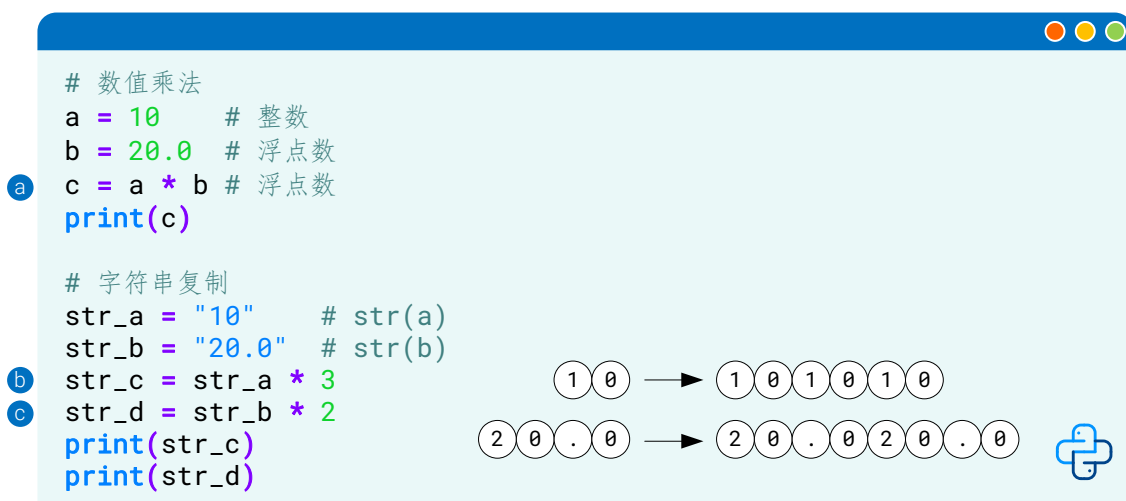


图 3. 乘法

除法运算符 / 用于将两个数值相除，结果为浮点数。

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

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。
- ▶ \\ 反斜线；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
```

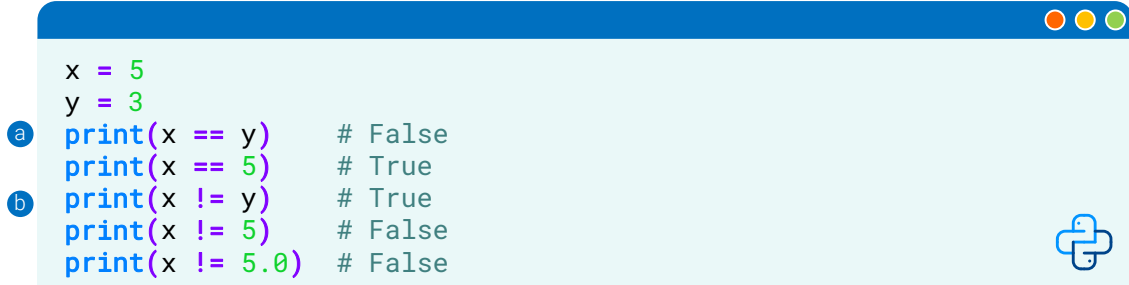
根据运算符的优先级规则，乘法运算 * 具有更高的优先级，因此先执行乘法，然后再进行加法。所以结果是 14。如果我们想先执行加法运算，然后再进行乘法运算，可以使用圆括号来改变优先级。

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`。

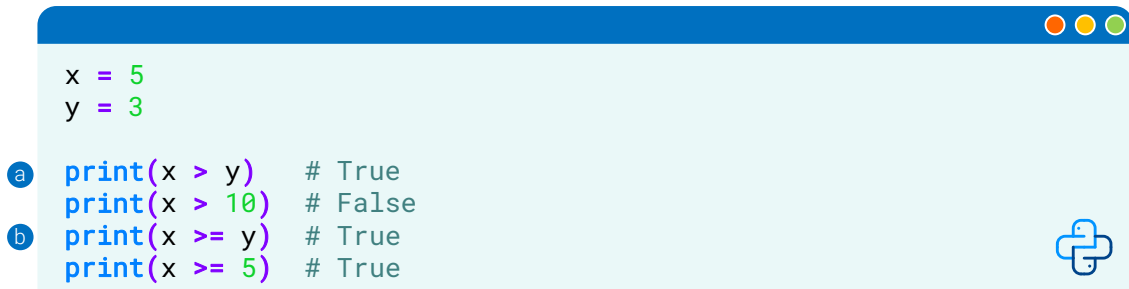


```
x = 5
y = 3
a print(x == y)      # False
  print(x == 5)      # True
b print(x != y)      # True
  print(x != 5)      # False
  print(x != 5.0)    # False
```

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`。

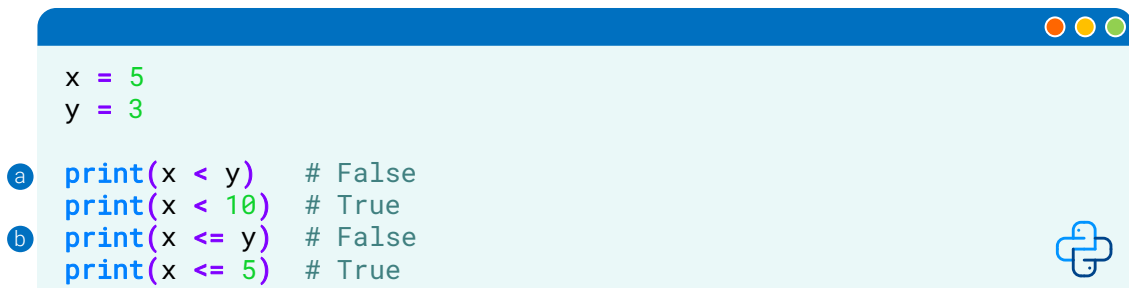


```
x = 5
y = 3
a print(x > y)       # True
  print(x > 10)      # False
b print(x >= y)      # True
  print(x >= 5)      # True
```

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`。



```
x = 5
y = 3
a print(x < y)       # False
  print(x < 10)      # True
b print(x <= y)      # False
  print(x <= 5)      # True
```

图 6. 小于、小于等于

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

6.4 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	A	B	A or B	A	not A
True	True	True	True	True	True	True	False
True	False	False	True	False	True	False	True
False	True	False	False	True	True		
False	False	False	False	False	False		

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
a print(True and True)
  print(True and False)
  print(False and True)
  print(False and False)

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

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

```

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。

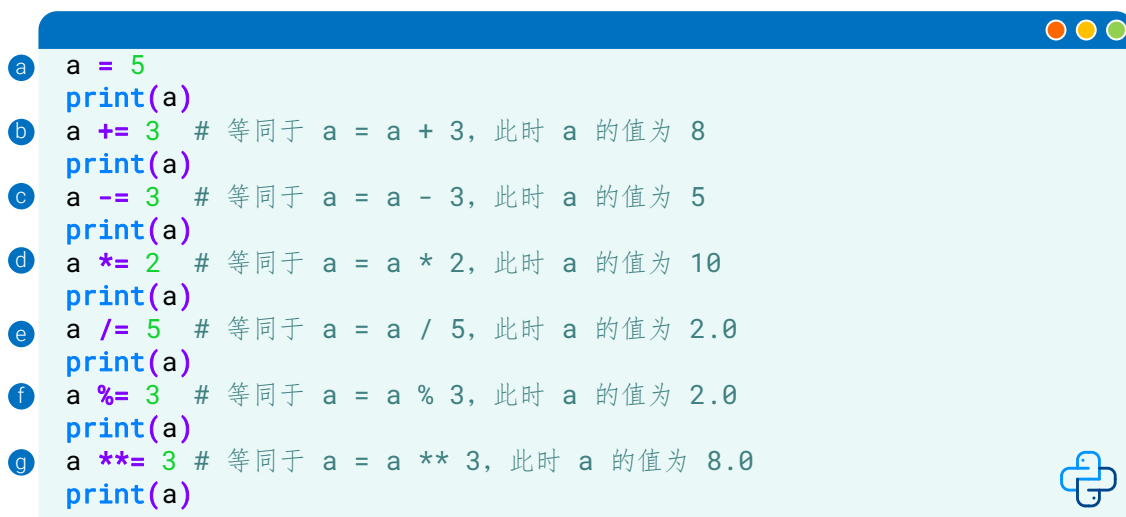


图 9. 赋值运算

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。

```

# 定义一个列表
a my_list = [1, 2, 3, 4, 5]

# 判断元素是否在列表中
b print(3 in my_list) # True
  print(6 in my_list) # False

# 判断元素是否不在列表中
c print(3 not in my_list) # False
  print(6 not in my_list) # True

```

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

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

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

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

l print(a_2_layers is e_2_layers)
m print(a_2_layers[2] is e_2_layers[2])

```

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

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。

6.9 聊聊 math 库

本节简单聊一聊 math 库。math 库是 Python 标准库之一，提供了许多数学函数和常量，用于执行各种基本数学运算。表 1 总结 math 库中常用的函数。

注意，如果需要向量化运算或使用更高级的数学操作，请使用 NumPy 或 SciPy 等第三方库。

大家可以在本书第 15 章找到表 1 中很多函数的图像。

表 1. math 库中常用函数

math 库函数	数学符号	描述
math.pi	π	圆周率, $\pi = 3.141592\dots$
math.e	e	$e = 2.718281\dots$
math.inf	∞	正无穷 (positive infinity), $-\text{math.inf}$ 为负无穷

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

<code>math.nan</code>	NaN	非数 (not a number)
<code>math.ceil(x)</code>	$\lceil x \rceil$	向上取整 (ceiling of x)
<code>math.floor(x)</code>	$\lfloor x \rfloor$	向下取整 (floor of x)
<code>math.comb(n, k)</code>	C_n^k	组合数 (combination), 输入均为整数 int 式子描述为 the number of ways to choose k items from n items without repetition and without order
<code>math.perm(n, k)</code>	P_n^k	排列数 (permutation), 输入均为整数 int 式子描述为 the number of ways to choose k items from n items without repetition and with order
<code>math.fabs(x)</code>	$ x $	绝对值 (absolute value)
<code>math.factorial(n)</code>	$n!$	阶乘 (factorial), 输入为整数 int
<code>math.sqrt(x)</code>	\sqrt{x}	平方根 (square root)
<code>math.cbrt(x)</code>	$\sqrt[3]{x}$	立方根 (cube root)
<code>math.exp(x)</code>	$\exp(x) = e^x$	指数 (natural exponential) 式子描述 e raised to the power x
<code>math.log(x)</code>	$\ln x$	自然对数 (natural logarithm)
<code>math.dist(p, q)</code>	$\ p - q\ $	欧几里得距离 (Euclidean distance)
<code>math.hypot(x1, x2, x3, ...)</code>	$\ x\ = \sqrt{x_1^2 + x_2^2 + x_3^2 + \dots}$	距离原点的欧几里得距离 (Euclidean distance from origin)
<code>math.sin(x)</code>	$\sin x$	正弦 (sine), 输入为弧度
<code>math.cos(x)</code>	$\cos x$	余弦 (cosine), 输入为弧度
<code>math.tan(x)</code>	$\tan x$	正切 (tangent), 输入为弧度
<code>math.asin(x)</code>	$\arcsin x$	反正弦 (arc sine), 结果在 $-\pi/2$ 和 $\pi/2$ 之间
<code>math.acos(x)</code>	$\arccos x$	反余弦 (arc cosine), 结果在 0 和 π 之间
<code>math.atan(x)</code>	$\arctan x$	反正切 (arc tangent), 结果在 $-\pi/2$ 和 $\pi/2$ 之间
<code>math.atan2(y, x)</code>	$\arctan\left(\frac{y}{x}\right)$	反正切 (arc tangent), 结果在 $-\pi$ 和 π 之间
<code>math.cosh(x)</code>	$\cosh x$	双曲余弦 (hyperbolic cosine)
<code>math.sinh(x)</code>	$\sinh x$	双曲正弦 (hyperbolic sine)
<code>math.tanh(x)</code>	$\tanh x$	双曲正切 (hyperbolic tangent)
<code>math.acosh(x)</code>	$\operatorname{arccosh} x$	反双曲余弦 (inverse hyperbolic cosine)
<code>math.asinh(x)</code>	$\operatorname{arsinh} x$	反双曲正弦 (inverse hyperbolic sine)
<code>math.atanh(x)</code>	$\operatorname{artanh} x$	反双曲正切 (inverse hyperbolic tangent)

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

<code>math.radians(x)</code>	$\frac{x}{180} \times \pi$	将角度 (degrees) 转换为弧度 (radians)
<code>math.degrees(x)</code>	$\frac{x}{\pi} \times 180$	将弧度转换为角度
<code>math.erf(x)</code>	$\text{erf } x = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt$	误差函数 (error function)
<code>math.gamma(x)</code>	$\Gamma(x) = (x-1)!$ * 仅当 x 为正整数	Gamma 函数 (gamma function)

图 14 代码给了一个用 `math.sin()` 计算等差数列列表 (list) 每个元素的正弦值。下面介绍其中主要语句。

a 利用 `import math` 将 `math` 库引入到 Python 程序中，这样我们可以在后面语句中使用 `math` 模块中的各种数学函数和常量。

b 导入 Matplotlib 库的 `pyplot` 模块，`as` 后面是模块的别名 `plt`。简单来说，`matplotlib.pyplot` 是 Matplotlib 众多子模块之一。后续代码接着用这个子模块绘图、标注等等操作。

c 利用 `math.sin()` 计算 `sin(0)`。

d 是一个 Python 赋值语句，将变量 `x_end` 的值设置为数学常量 `2*math.pi`。`math.pi` 是 Python 标准库中 `math` 模块中的一个常量，它代表圆周率 π ，它的值约为 3.1415926。然后，下一句代码计算 `sin(2π)`。

e 定义了等差数列 (arithmetic progression) 元素的数量，`x_start` 为数列第一项，`x_end` 为数列最后一项。**f** 计算等差数列的公差。图 13 所示为在实数轴上看等差数列。

g 利用列表生成式 (list comprehension) 生成等差数列列表 `x_array`。其中，`for i in range(num)` 是列表生成式的 `for` 循环部分。`range(num)` 生成的整数序列，其中 `num` 是要生成的元素数量，这个序列将会包括从 0 到 `num-1` 的整数值。因此，这个 `for` 循环将会执行 `num` 次，每次使用一个新的 `i` 值来生成一个新的列表元素。而列表的元素为 `x_start + i * step`，即等差数列的每一项。这句的最终结果是一个由 37 个元素构成的列表 `x_array`。列表本身就是一个等差数列，数列的第一项为 0，最后一项为 2π 。

本书第 7 章将在 `for` 循环中专门介绍列表生成式。

大家会发现本书后续经常用 `numpy.arange()` 和 `numpy.linspace()` 生成等差数列。

h 也用列表生成式创建和 `x_array` 元素数量一致的全 0 列表。

i 利用 `matplotlib.pyplot.plot()`，简做 `plt.plot()`，绘制“折线 + 散点图”。`x_array` 为散点横轴坐标，`zero_array` 散点的纵轴坐标。将这些散点顺序连线，我们便获得折线；这个例子中的折线恰好为直线。如图 12 (b) 所示，将子图散点顺序连线我们得到正弦曲线。这条曲线看上去“光滑”，而本质上也是折线。这提醒了我们，只有散点足够密集，也就是颗粒度够高时，折线才看上去更细腻、顺滑。特别是，当曲线特别复杂时，我们需要更高颗粒度。

默认情况下，`matplotlib.pyplot.plot()` 只绘制折线，不突出显示散点。

参数 `marker='.'` 指定散点标记样式。

参数 `markersize=8` 指定散点标记大小的参数。

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

参数 `markerfacecolor='w'` 指定散点标记内部颜色, 'w'代表白色。

参数 `markeredgecolor='k'` 指定散点标记边缘颜色'k'代表黑色。

① 利用 `matplotlib.pyplot.text()`, 简做 `plt.text()`, 在图中添加文本注释 (annotation)。其中, `x_start` 是文本注释的横轴坐标, `y` 是文本注释的纵轴坐标, `text` 是要显示的文本字符串。

② 也是用 `plt.text()` 在图中添加文本数值, 文本坐标不同, 文本本身也不同。`r'2π'` 是一个包含 LaTeX 表达式的字符串。r 字符前缀表示原始字符串 (raw string), 以确保 LaTeX 表达式中的反斜杠不被转义。\$ 符号用于标识 LaTeX 表达式的开始和结束。在图中, 文本最终打印效果为 2π 。

③ 这行代码的作用是在当前的 Matplotlib 图形中关闭坐标轴的显示, 从而在图形中不显示坐标轴刻度、标签和框线。

本书第 10 章将专门介绍一幅图中各种组成元素。

④ 用于显示在创建的图形。

⑤ 还是用列表生成式创建一个列表, 这个列表每个元素是 `x_array` 等差数列列表对应元素的正弦值。

⑥ 同样利用 `plt.plot()` 绘制正弦函数 $f(x) = \sin(x)$ 的“折线 + 散点图”。

⑦ 利用 `plt.axhline()` 在图形中添加一条水平的参考线。

参数 `y=0` 是参考线的水平位置。

参数 `color='k'` 是参考线的颜色设置。

参数 `linestyle='--'` 是参考线的线型设置, '--' 表示虚线。

参考 `linewidth=0.25` 是参考线的线宽设置。在这里, 线宽被设置为 0.25 个单位。在 Matplotlib 中, `linewidth` 的单位是点 (point), 通常表示为 pt。1 pt 等于 1/72 英寸。点用于度量线宽的标准单位, 通常用于印刷和出版领域。文字大小也可以用 pt 表示, 比如鸢尾花书正文文字大小为 10 pt。而 5 号字体为 10.5 pt, 小五号字为 9 pt。

大家可能已经发现这段代码的最大问题, 就是我们反复利用列表生成式 (本质上是 for 循环) 生成各种序列, 也就是 for 循环中一个个运算。本书后文会介绍如何用 NumPy 向量化 (vectorize) 上述。

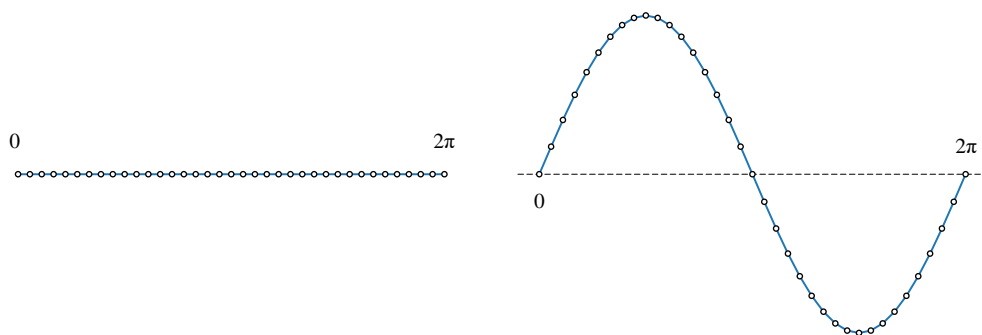


图 12. 可视化等差数列, 正弦函数

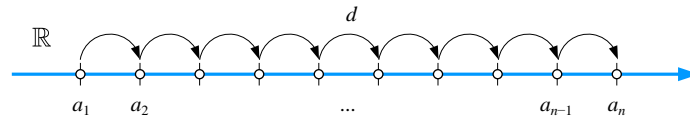


图 13. 实数轴上看等差数列

```

# 导入包
a import math
b import matplotlib.pyplot as plt

# 计算正弦值
x_start = 0 # 弧度值
c print(math.sin(x_start))
d x_end = 2*math.pi # 弧度值
  print(math.sin(x_end))

# 等差数列 a_n = a_1 + d(n - 1)
# 数列元素数量
e num = 37
# 计算公差
f step = (x_end - x_start) / (num - 1)
# 生成等差数列列表
g x_array = [x_start + i * step for i in range(num)]
# 生成等长全0列表, 等价于 zero_array = [0] * len(x_array)
h zero_array = [0 for i in range(num)]

# 可视化等差数列
i plt.plot(x_array, zero_array, marker = '.',
           markersize = 8,
           markerfacecolor="w",
           markeredgecolor='k')
j plt.text(x_start, 0, '0')
k plt.text(x_end, 0, r'$2\pi$')
l plt.axis('off')
m plt.show()

# 正弦 sin(x) 列表
n y_array = [math.sin(x_idx) for x_idx in x_array]

# 可视化正弦函数
o plt.plot(x_array, y_array, marker = '.',
           markersize = 8,
           markerfacecolor="w",
           markeredgecolor='k')
  plt.text(x_start, -0.1, '0')
  plt.text(x_end, 0.1, r'$2\pi$')
p plt.axhline(y=0, color='k', linestyle='--', linewidth=0.25)
  plt.axis('off')
  plt.show()

```

图 14. 利用 math.sin() 计算数列列表正弦



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。