

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；a += b 等价于 a = a + b
- ◀ -= 赋值运算符，自减运算；a -= b 等价于 a = a - b
- ◀ *= 赋值运算符，自乘运算；a *= b 等价于 a = a * b
- ◀ /= 赋值运算符，自除运算；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



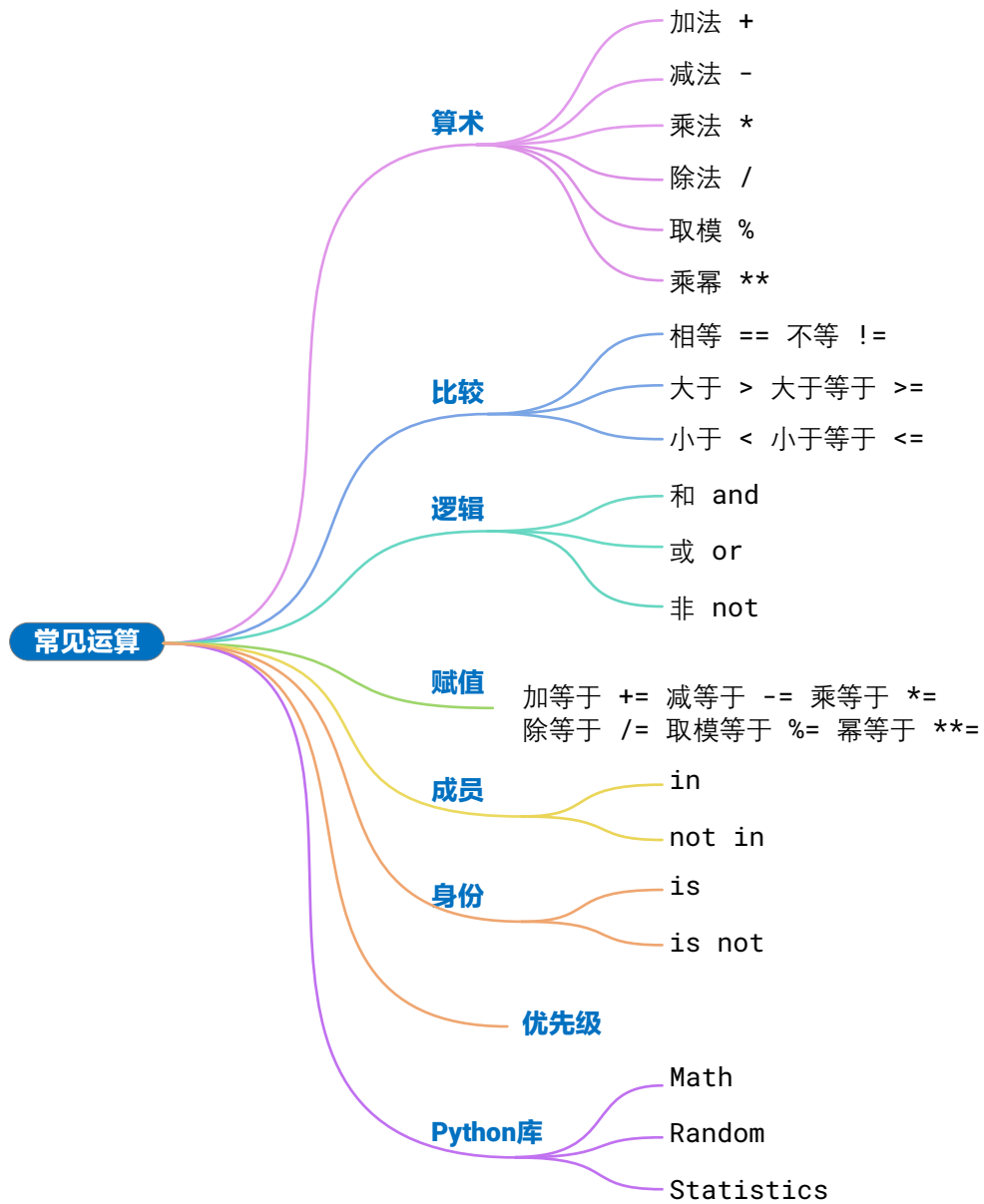
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

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 (/)、取模 (%)、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。

图 1 总结 Python 中常见的运算符，大家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。

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。比如，代码 1 的 **a** 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。

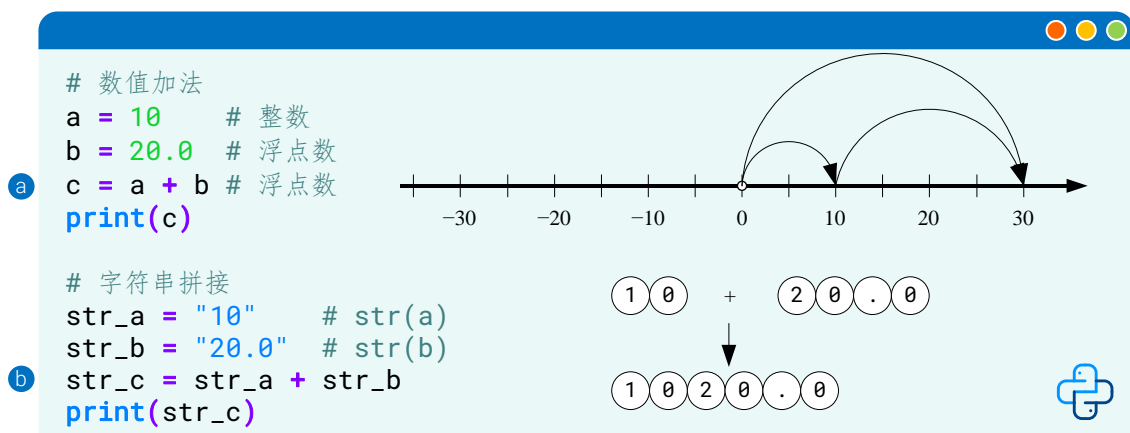
如果一个操作数是浮点数，而另一个是复数，则浮点数会被转换为复数，然后进行加法运算。运算结果为复数。

⚠ 注意，整数或浮点数不能和字符串数字相加，比如 `2 + '1'` 会报错，错误信息为 `TypeError: unsupported operand type(s) for +: 'int' and 'str'`。

代码 1 的 **b** 展示的就是上一章讲过的字符串拼接。请大家先将两个字符串用 `float()` 转化为浮点数，再完成加法运算。

⚠ 注意，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'`。

请大家在 JupyterLab 中自行练习代码 1。



代码 1. 加法; Bk1_Ch06_01.ipynb

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。

代码 2 中 **a** 完成整数和浮点数的乘法运算，结果还是一个浮点数。

b 和 **c** 则完成字符串的复制运算，而不是算术乘法运算。

幂运算符 ** 用于将一个数值的幂次方，比如 `2**3` 的结果为 8。

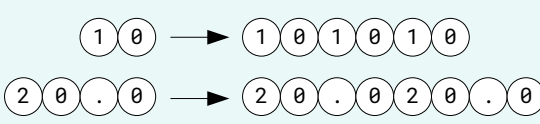

⚠ 注意，NumPy 数组完成**矩阵乘法** (matrix multiplication) 时用的运算符为 @。

```

# 数值乘法
a = 10      # 整数
b = 20.0    # 浮点数
a c = a * b  # 浮点数
print(c)

# 字符串复制
str_a = "10"    # str(a)
str_b = "20.0"  # str(b)
b str_c = str_a * 3
c str_d = str_b * 2
print(str_c)
print(str_d)

```


代码 2. 乘法;  Bk1_Ch06_02.ipynb

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 `\\`。
- ▶ `\'` 表示单引号。
- ▶ `\"` 表示双引号。
- ▶ `\` 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。注意，取模主要是用于编程中。取余则更多表示数学概念。两者最大区别在于对负数求余数的处理上。



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。

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

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。这和小学数学学的运算法则完全一致。

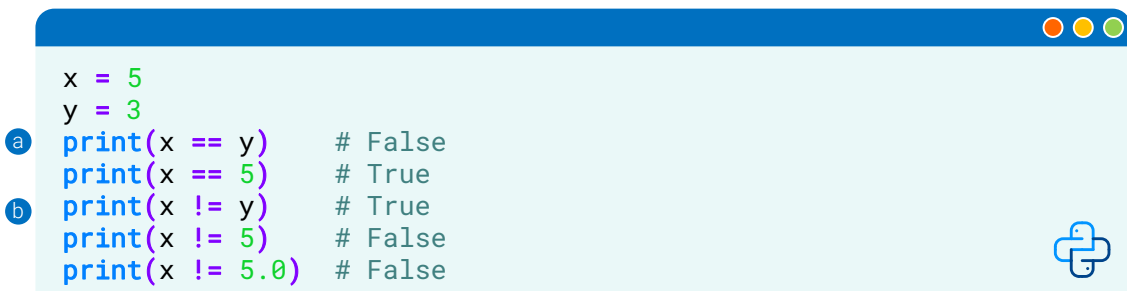
6.3 比较运算符

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。

请大家在 JupyterLab 中自行练习代码 3。



```
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
```

代码 3. 相等、不等;  Bk1_Ch06_03.ipynb

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。

请大家在 JupyterLab 中自行练习代码 4。

```

x = 5
y = 3

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

```

代码 4. 大于、大于等于; Bk1_Ch06_03.ipynb

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`。

请大家在 JupyterLab 中自行练习代码 5。

```

x = 5
y = 3

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

```

代码 5. 小于、小于等于; Bk1_Ch06_03.ipynb

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 2 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 2. 真值表

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>

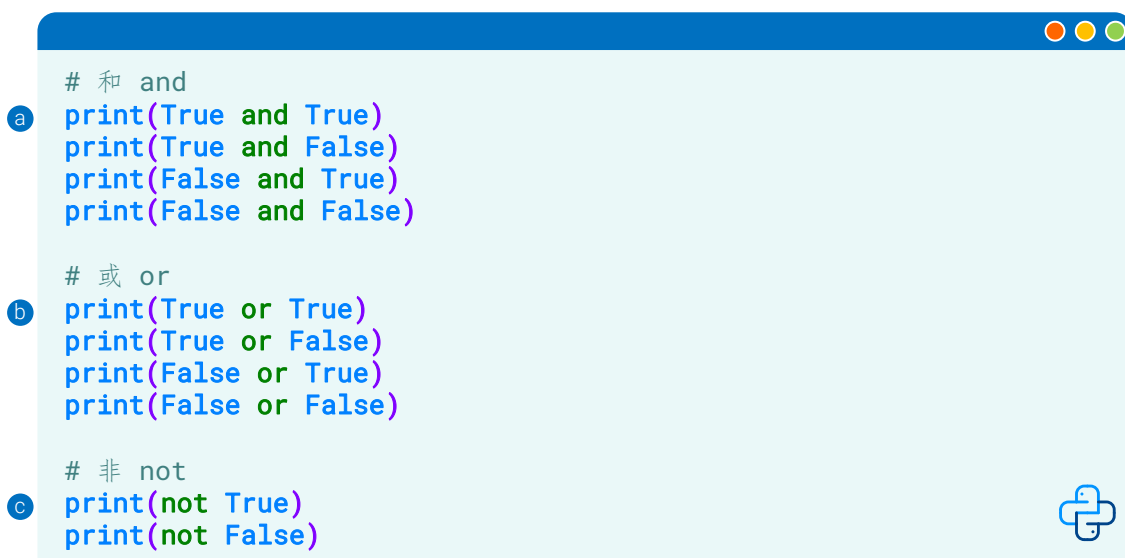
本书配套微视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

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

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`。

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。

请大家在 JupyterLab 中自行练习代码 6。并尝试有选择地把 `True` 替换成 `1`，把 `False` 替换成 `0`，再完成这些逻辑运算并查看结果。



```

# 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)

```

代码 6. 逻辑运算符;  BK1_Ch06_04.ipynb

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。

请大家在 JupyterLab 中自行练习代码 7。


```

a a = 5
  print(a)
b a += 3 # 等同于 a = a + 3, 此时 a 的值为 8
  print(a)
c a -= 3 # 等同于 a = a - 3, 此时 a 的值为 5
  print(a)
d a *= 2 # 等同于 a = a * 2, 此时 a 的值为 10
  print(a)
e a /= 5 # 等同于 a = a / 5, 此时 a 的值为 2.0
  print(a)
f a %= 3 # 等同于 a = a % 3, 此时 a 的值为 2.0
  print(a)
g a **= 3 # 等同于 a = a ** 3, 此时 a 的值为 8.0
  print(a)

```

代码 7. 赋值运算;  Bk1_Ch06_05.ipynb

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`。

代码 8 展示成员运算符的示例，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

```

代码 8. 成员运算;  Bk1_Ch06_06.ipynb

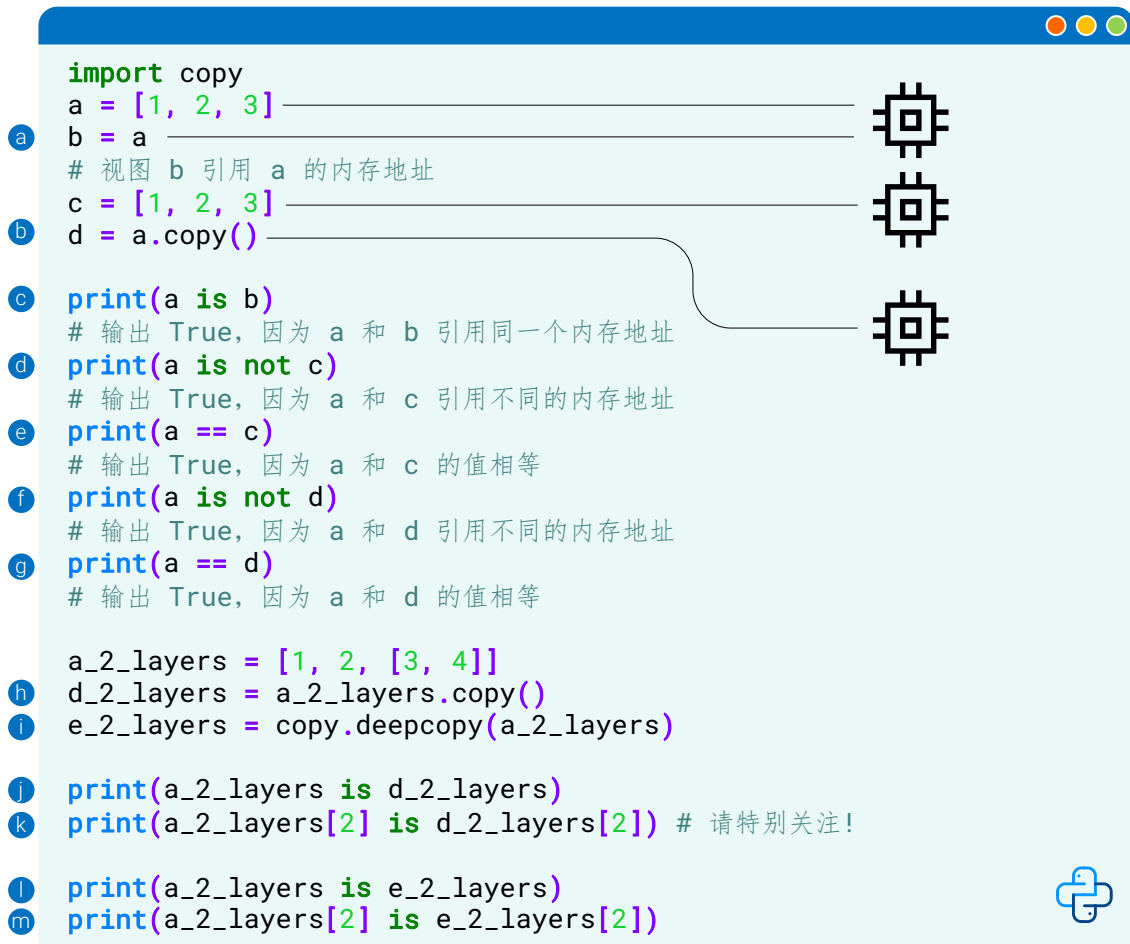
6.7 身份运算符

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。

请大家自行练习代码 9。请特别注意代码 9 中 **k** 的结果，并解释原因。



```

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])

```

代码 9. 身份运算;  Bk1_Ch06_07.ipynb

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。

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

- ▶ 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 |
|--------------------------------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <code>math.pi</code> | π | 圆周率, $\pi = 3.141592\dots$ |
| <code>math.e</code> | e | $e = 2.718281\dots$ |
| <code>math.inf</code> | ∞ | 正无穷 (positive infinity), <code>-math.inf</code> 为负无穷 |
| <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) |

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.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> | $\operatorname{arcsinh} x$ | 反双曲正弦 (inverse hyperbolic sine) |
| <code>math.atanh(x)</code> | $\operatorname{arctanh} x$ | 反双曲正切 (inverse hyperbolic tangent) |
| <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> | $\operatorname{erf} x = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2) dt$ | 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) |

代码 10 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 4 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。



也用列表生成式创建和 `x_array` 元素数量一致的全 0 列表。



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 3 (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。

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。

⚠ 注意，如果数据本身就是离散的，我们最好不用折线将它们连起来。这时可以采用散点图、火柴梗图等可视化方案。



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，`0` 是文本注释的纵轴坐标，'`0`' 是要显示的文本字符串。

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 。

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。



用于显示在创建的图形。



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。

① 同样利用 `plt.plot()` 绘制正弦函数 $f(x) = \sin(x)$ 的“折线 + 散点图”。

② 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，5 号字体为 10.5 `pt`，小五号字为 9 `pt`。鸢尾花书正文文字大小为 10 `pt`。为了缩减空间、节省用纸，《编程不难》中嵌入的代码文字字体采用 Roboto Mono Light，字号为 9pt，即小五号字。

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。本书后会介绍如何用 NumPy 向量化 (vectorize) 上述。

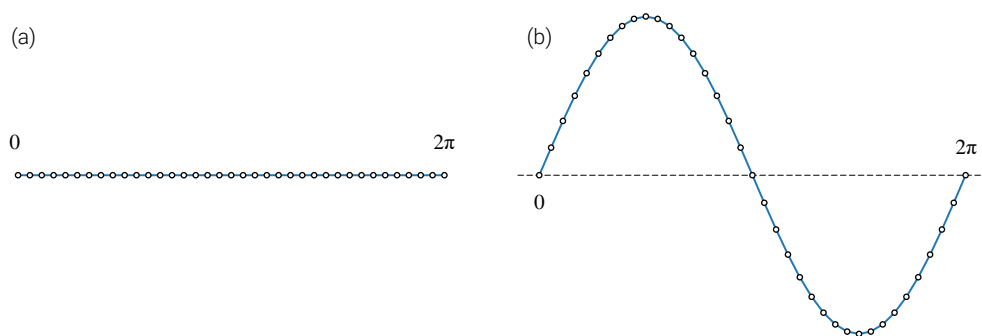


图 3. 可视化等差数列，正弦函数

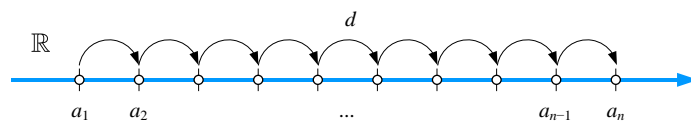


图 4. 实数轴上看等差数列

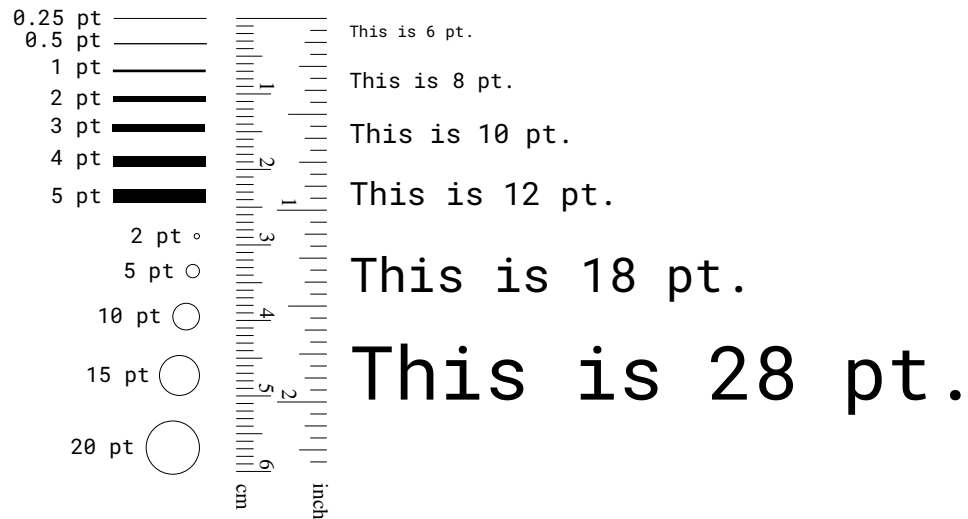


图 5. 线宽、字体等大小单位 point (pt)

```

# 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))

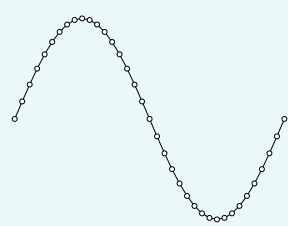
# 等差数列  $a_n = a_1 + d(n - 1)$ 
# 数列元素数量
e num = 37
# 计算公差
f step = (x_end - x_start) / (num - 1)
# 生成等差数列列表
g x_array = [x_start + i * step for i in range(num)]
# 生成等长全0列表, 等价于 zero_array = [0] * len(x_array)
h zero_array = [0 for i in range(num)]

# 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()

```



代码 10. 利用 `math.sin()` 计算数列列表正弦值并可视化; Bk1_Ch06_08.ipynb

6.10 聊聊 random 库和 statistics 库

`random` 是 Python 标准库中的一个模块, 提供了伪随机数生成器, 通常用于模拟随机事件、生成随机数据、进行随机采样等任务。在进行科学实验、模拟和游戏开发等领域, 它是一个非常有用的工

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

具。如果需要更高级的随机数生成或概率分布模拟，可以考虑使用 NumPy 或 SciPy 等第三方库，它们提供了更多的随机数生成和统计分析功能。

statistics 是 Python 标准库中的一个模块，用于执行统计计算和操作，包括平均值、中位数、标准差、方差、众数等。这个模块提供了一些基本的统计函数，适用于处理数值数据。

⚠ 注意，statistics 模块适用于处理小规模数据集，对于大型数据集和更复杂的统计分析，通常需要使用专门的数据科学库，如 NumPy、SciPy 或 Pandas。

下面，我们举几个例子，介绍如何使用 random 和 statistics 这两个 Python 库。

表 2. random 库中常用函数

| random 库函数 | 描述 |
|-----------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| random.random() | 返回一个 0 到 1 之间的随机浮点数 |
| random.randint(a, b) | 返回一个在区间 [a, b] 内的随机整数 |
| random.uniform(a, b) | 返回一个在区间 [a, b] 内的随机浮点数 |
| random.gauss(mu, sigma) | 生成一个服从一元高斯分布 (univariate Gaussian distribution) 的随机数，其中 mu 是均值，sigma 是标准差 |
| random.seed(seed) | 使用给定的种子 seed 初始化随机数生成器，这有助于结果可复刻 |
| random.choice(seq) | 从序列 (如列表、元组或字符串) 中随机选择一个元素并返回它 |
| random.choices(population, weights=None, k=k) | 从给定的 population 序列中随机选择 k 个元素，可以通过 weights 参数指定每个元素的权重，权重越高的元素被选中的概率越大。如果不指定权重，所有元素被选中的概率相等 |
| random.shuffle(sq) | 用于随机打乱序列 seq 中的元素顺序 |
| random.sample(population, k) | 从指定的 population 序列中随机选择 k 个不重复的元素，相当于从总体中采集 k 个不重复的样本 |
| random.betavariate(alpha, beta) | 用于生成一个服从 Beta 分布 (Beta distribution) 的随机数。Beta 分布是一个概率分布，其形状由两个参数 alpha 和 beta 来控制。《统计至简》将专门介绍这个概率分布，并在贝叶斯推断 (Bayesian inference) 中使用 Beta 分布 |
| random.expovariate(lambd) | 用于生成一个服从指数分布 (exponential distribution) 的随机数，lambd 是指数分布的一个参数。《统计至简》将介绍指数分布 |

表 3. statistics 库中常用函数

| statistics 库函数 | 描述 |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| statistics.mean() | 计算算术平均值 (arithmetic mean, average) |
| statistics.median() | 计算中位数 (median) |
| statistics.mode() | 计算众数 (mode) |
| statistics.quantiles() | 用于计算分位数 (quantile) 的函数。简单来说，分位数是指将一组数据按照大小顺序排列后，把数据分成若干部分的值，每一部分包含了一定比例的数据。通常，我们使用四分位数来分割数据集，这将数据集分为四个部分，分别包含 25%、50%、75%和 100%的数据。 |
| statistics.pstdev() | 计算数据的总体标准差 (population standard deviation) |
| statistics.stdev() | 计算数据的样本标准差 (sample standard deviation) |
| statistics.pvariace() | 计算数据的总体方差 (population variance) |
| statistics.variance() | 计算数据的样本方差 (sample variance) |
| statistics.covariance() | 计算数据的样本协方差 (sample covariance) |
| statistics.correlation() | 计算数据的皮尔逊相关性系数 (Pearson's correlation coefficient) |
| statistics.linear_regression() | 计算一元线性回归函数斜率 (slope) 和截距 (intercept) |

质地均匀抛硬币

这段代码模拟了抛硬币的实验，并记录每次抛硬币后的结果，然后计算当前所有结果均值。如图 6 (a) 所示为前 100 次投硬币结果，正面为 1，反面为 0。

图 6 (b) 反映了均值随时间的演化过程。随着抛硬币次数的增加，均值逐渐趋于 0.5，这是因为硬币正反面出现的概率是相等的。

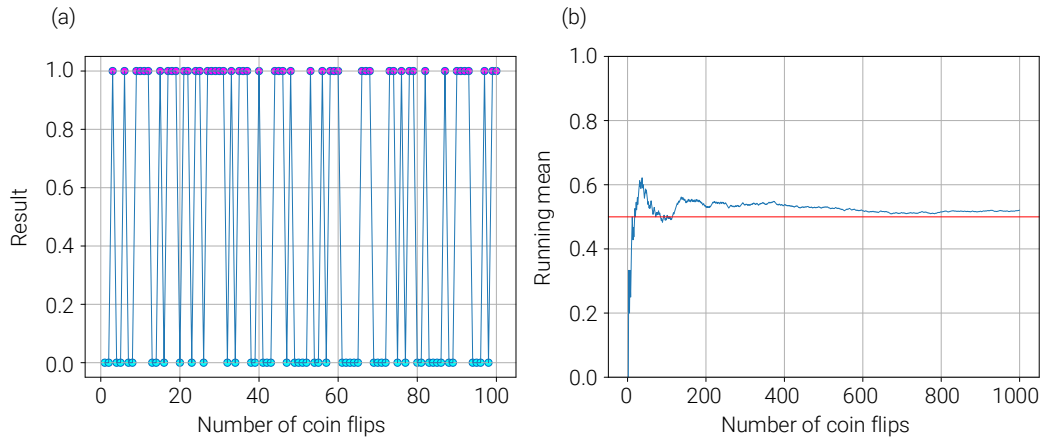


图 6. 抛均匀硬币模拟结果; (a) 前 100 次投掷结果; (b) 平均值随投掷次数变化

下面我们聊一下代码 11。

a 导入 Python 的 `random` 库。 **b** 导入 Python 的 `statistics` 库。 **c** 定义变量储存抛硬币数量。

d 定义空列表用来储存抛硬币结果。 **e** 定义空列表用来储存平均值。

f 用 `for` 循环遍历每次抛硬币。注意，下划线 `_` 是一个占位符，表示一个不需要使用的变量。在这个 `for` 循环中，`_` 表示迭代变量不会被使用，我们仅仅关注循环次数。

g 用 `random.randint(0,1)` 返回一个在区间 `[0, 1]` 内的随机整数，即 0 和 1。两者个整数有相同概率。

h 用 `append()` 方法在列表末尾添加新元素。

i 利用 `statistics.mean()` 计算当前所有结果的平均值。 **j** 将当前均值添加到列表中。

k 利用 `matplotlib.pyplot.scatter()`，简做 `plt.scatter()`，绘制前 100 次抛硬币结果散点图。

其中，`range(1, visual_num + 1)` 是散点的横轴位置。`range(1, visual_num + 1)` 生成一个从 1 到 `visual_num` 的整数序列。`results[0:visual_num]` 取出列表前 100 个元素，它们是散点的纵轴位置。

参数 `c=results[0:visual_num]` 用于指定散点的颜色的参数。由于结果仅有 0 和 1 两个值，因此最终会用两个颜色来展示散点。

参数 `marker="o"` 将散点的形状设置为圆圈。

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

参数 `cmap='cool'` 用于指定颜色映射。它决定了如何将 0 和 1 映射到不同的颜色。'cool' 是一种预定义的颜色映射，它将 0 映射为天蓝色，1 映射为粉紫色。

- l 利用 `matplotlib.pyplot.plot()`，简做 `plt.plot()`，绘制结果折线图。
- m 用折线图可视化均值随抛掷次数变化过程。
- n 利用 `matplotlib.pyplot.axhline()`，简做 `plt.axhline()`，绘制水平线。

```

a import random
b import statistics
  import matplotlib.pyplot as plt

  # 抛硬币实验的次数
c num_flips = 1000

  # 用于存储每次抛硬币的结果
d results = []
  # 用于存储每次抛硬币后的均值
e running_means = []


f for _ in range(num_flips):
  # 随机抛硬币，1代表正面 (H)，0代表反面 (T)
g   result_idx = random.randint(0, 1)
h   results.append(result_idx)

  # 计算当前所有结果的均值
i   mean_idx = statistics.mean(results)
j   running_means.append(mean_idx)

  # 可视化前100次结果均值随次数变化
  visual_num = 100
k plt.scatter(range(1, visual_num + 1), results[0:visual_num],
              c=results[0:visual_num], marker="o", cmap = 'cool')
l plt.plot(range(1, visual_num + 1), results[0:visual_num])
  plt.xlabel("Number of coin flips")
  plt.ylabel("Result")
  plt.grid(True)
  plt.show()

  # 可视化均值随次数变化
m plt.plot(range(1, num_flips + 1), running_means)
n plt.axhline(0.5, color = 'r')
  plt.xlabel("Number of coin flips")
  plt.ylabel("Running Mean")
  plt.grid(True)
  plt.ylim(0,1)
  plt.show()

```

代码 11. 质地均匀的硬币;  Bk1_Ch06_09.ipynb

硬币“头重脚轻”

假设硬币不均匀，抛掷结果为 1 的概率为 0.6，为 0 的概率为 0.4。图 7 (a) 所示为前 100 次投掷结果。图 7 (b) 所示为均值随抛掷次数变化。

代码 12 了列表生成式获得硬币结果列表，以及均值列表。^a 利用 `random.choices()` 从一个包含两个元素的序列 `[0, 1]` 中随机选择一个元素，并且为每个元素指定了相应的权重。选择 0 的概率为 0.4，选择 1 的概率为 0.6。

请大家修改代码 11 自行绘制图 7 两图。

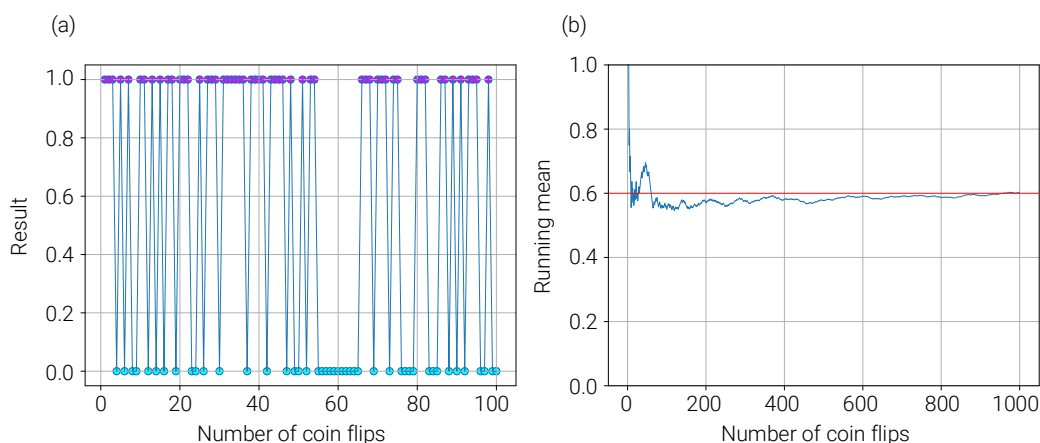



图 7. 抛头重脚轻硬币模拟结果；(a) 前 100 次投掷结果；(b) 平均值随投掷次数变化

```
import random
import statistics
import matplotlib.pyplot as plt

# 抛硬币实验的次数
num_flips = 1000

# 模拟抛硬币实验，硬币头重脚轻
# 用于存储每次抛硬币的结果
a results = [random.choices([0, 1], [0.4, 0.6])[0]
              for _ in range(num_flips)]

# 用于存储每次抛硬币后的均值
b running_means = [statistics.mean(results[0:idx+1])
                   for idx in range(num_flips)]
```

代码 12. 头重脚轻的硬币；  Bk1_Ch06_10.ipynb

混合两个一元高斯分布随机数

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

图 8 所示为混合了两个服从不同一元高斯分布随机数组的**直方图** (histogram)。直方图是一种用于可视化数据分布的图表类型，通常用于展示数据的**频率** (frequency)、**概率** (probability) 或 **概率密度** (probability density 或 density)。

频率直方图的纵轴表示数据集中每个数值或数值范围的出现次数。每个数据点或数据范围对应一个柱状条，柱状条的高度表示该数据点或数据范围在数据集中出现的次数。直方图所有柱状条的高度之和为样本的数量。比如，这个例子中样本数据一共有 1000 个样本，如果直方图纵轴为频率，则所有柱状条的高度之和为 1000。

概率直方图的纵轴表示每个数据点或数据范围在数据集中出现的概率。直方图所有柱状条的高度之和为 1。

如图 8 所示，概率密度直方图的纵轴表示每个数据点或数据范围的概率密度。这幅图中，所有柱状条的面积之和为 1。

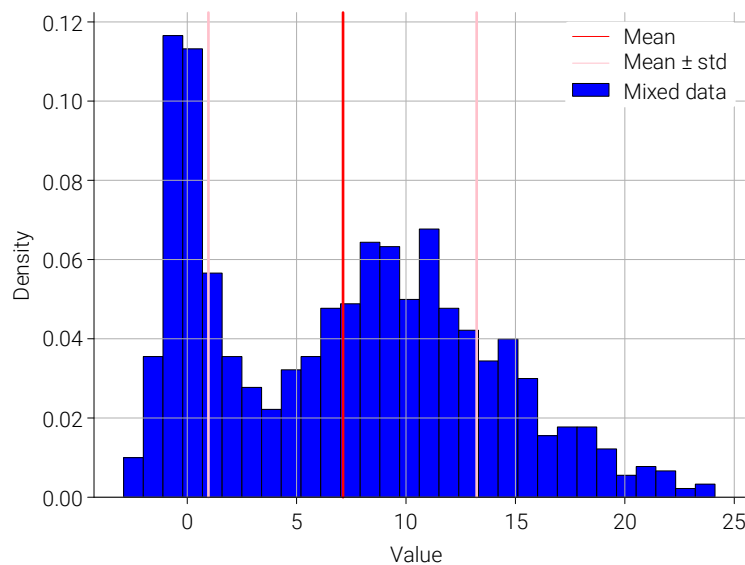


图 8. 混合样本数据的直方图

下面聊聊代码 13。

a 用多重赋值语句，分别定义了均值、标准差、样本数量。**b** 利用 `random.gauss()` 和列表生成式生成 300 个服从特定正态分布的随机数。

c 也用多重赋值语句，定义了第二组均值、标准差、样本数量。**d** 生成了另外一组 700 个随机数，它们服从第二个正态分布。

e 用列表加法将两个列表拼接。

f 利用 `statistics.mean()` 计算 1000 个样本数据均值。

g 利用 `statistics.stdev()` 计算 1000 个样本数据标准差。

h 利用 `matplotlib.pyplot.hist()`，简做 `plt.hist()`，绘制样本数据直方图。

参数 `bins=30` 指定直方图中柱状条的数量为 30。每个柱状条代表数据的一个范围。

参数 `density=True` 表示直方图的纵轴将表示概率密度，即所有柱状条的面积总和等于 1。如果设置为 `False`，纵轴将表示频率，即所有柱状条的高度总和为样本数，即 1000。

参数 `edgecolor='black'` 设置柱状条边框颜色的为黑色。

参数 `alpha=0.7` 设置柱状条的透明度的，取值范围为 0 到 1。0 表示完全透明，1 表示完全不透明。

参数 `color='blue'` 设置柱状条的填充颜色为蓝色。

参数 `label='Mixed Data'` 设置直方图的标签，用于在图例中标识直方图的内容。

① 用 `matplotlib.pyplot.axvline()`，简做 `plt.axvline()`，绘制竖直参考线，用来展示均值位置。并设置参考线颜色为红色，以及图例标签。请大家用 `linewidth` 或 `lw`，设置线宽，用 `linestyle` 或 `ls` 设置线条类型。

② 和 ③ 绘制 $\mu \pm \sigma$ 参考线。


④ 展示图例。

```
import random
import statistics
import matplotlib.pyplot as plt
random.seed(0) # 方便复刻结果
# 生成300个服从N(0, 1**2)的随机数
a mean1, std1, size1 = 0, 1, 300
b data1 = [random.gauss(mean1, std1) for _ in range(size1)]

# 生成700个服从N(10, 5**2)的随机数
c mean2, std2, size2 = 10, 5, 700
d data2 = [random.gauss(mean2, std2) for _ in range(size2)]

# 将两组随机数混合
e mixed_data = data1 + data2
f mean_loc = statistics.mean(mixed_data)
g std_loc = statistics.stdev(mixed_data)

# 绘制混合数据的直方图
h plt.hist(mixed_data, bins=30, density=True, edgecolor='black',
           alpha=0.7, color='blue', label='Mixed data')
i plt.axvline(mean_loc, color='red', label='Mean')
j plt.axvline(mean_loc + std_loc,
             color='pink', label='Mean + std')
k plt.axvline(mean_loc - std_loc, color='pink')
plt.xlabel('Value')
plt.ylabel('Density')
l plt.legend()
plt.title('Histogram of Mixed Data')
plt.grid(True)
plt.show()
```

代码 13. 混合两个一元高斯分布随机数;  Bk1_Ch06_11.ipynb

线性回归

我们在本书第 1 章提过**线性回归** (linear regression) 这个概念。简单来说，线性回归是一种统计学和机器学习中常用的方法，用于建立自变量与因变量之间线性关系的模型。通过拟合一条直线，预测因变量的值。

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

图 9 (a) 所示为样本数据散点图，横轴对应自变量，纵轴对应因变量。显然，我们一眼就发现自变量和因变量之间似乎存在某种线性关系，也就是找到一条线解释两者关系。图 9 (b) 中的红色直线就是这条直线。

在 `statistics` 库中的 `linear_regression()` 函数可以帮助我们找到图 9 (b) 红色直线的斜率 (slope) 和截距 (intercept)。

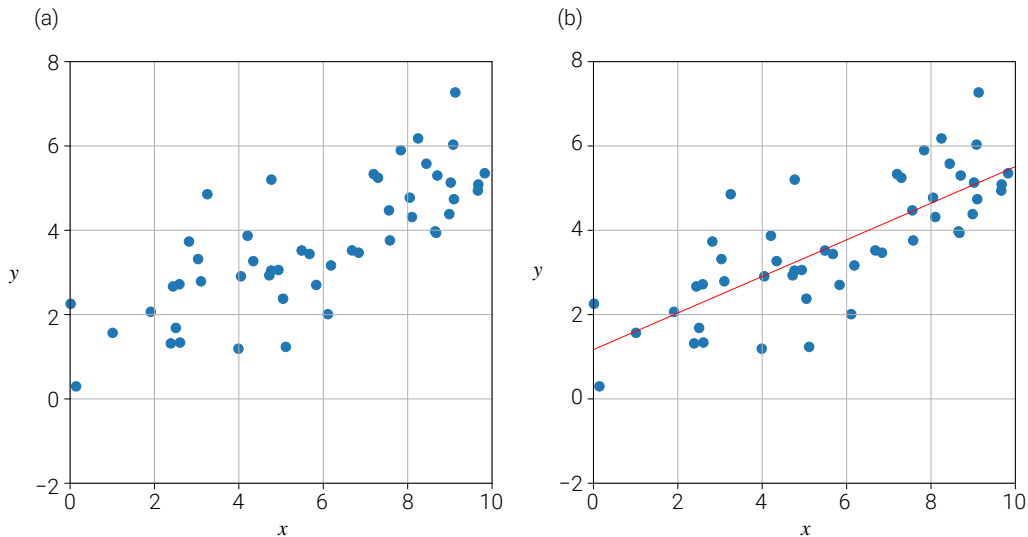


图 9. 线性回归

代码 14 完成线性回归运算并绘制图 9，下面我们聊一聊其中关键语句。

a 用列表生成式和 `random.uniform()` 产生在区间 $[0, 10)$ 内均匀分布的 50 个随机数。这些数字代表自变量的数据。

类似 **a**，**b** 用列表生成式和 `random.gauss()` 产生数据噪音 (noise)。

c 用列表生成式产生因变量数据，即 $0.5 \times x + 1 + \text{noise}$ 。其中，0.5 为斜率，1 为截距。

d 利用 `plt.subplots()` 产生图形对象 `fig`、轴对象 `ax`。

e 在轴对象 `ax` 上用 `.scatter()` 方法绘制散点图。

f 在轴对象 `ax` 上用 `.set_xlabel('x')` 和 `.set_ylabel('y')` 设置横纵轴标签。注意，**f** 有两句话，用半角分号 (semicolon) ; 分隔。

g 设置轴对象 `ax` 的纵横比例为相等。

h 利用 `set_xlim()` 和 `set_ylim()` 方法设置轴对象 `ax` 的横轴、纵轴取值范围，这两句也是用半角分号分隔。

i 设置轴对象 `ax` 的背景网格线。

j 调用 `statistics.linear_regression()` 函数计算一元线性回归模型的斜率和截距。

k 这一段利用 `while` 循环生成等差数列。等差数列的起始值为 0，结束值为 10，步长 (公差) 为 0.5。具体来说，`while` 循环开始时，`x_i` 被初始化为 `start`。在每次循环迭代中，`x_i` 被添加 (append 方法) 到列表 `x_array` 中，然后增加 `(+=) step` 的值。循环会一直执行，直到 `x_i` 大于 `end`，便停止 `while` 循环。

l 用列表生成式计算 `x_array` 预测值。

`statistics` 库中的 `linear_regression()` 函数仅仅能处理一元线性回归，而且不能做回归分析；因此，在实践中我们并不会使用这个函数。但是，本书第 8 章会利用这个函数介绍如何通过学习别人的源代码来提高编程能力，这就是为什么我们要在此处安排这部分内容的原因。

➡ 本书第 27 章会介绍 `Statsmodels` 中的回归分析函数，本书第 30 章将介绍 `Scikit-Learn` 中的各种回归分析工具。

```
# 导入包
import random
import statistics
import matplotlib.pyplot as plt
# 产生数据
num = 50
random.seed(0)
a x_data = [random.uniform(0, 10) for _ in range(num)]
# 噪音
b noise = [random.gauss(0,1) for _ in range(num)]
c y_data = [0.5 * x_data[idx] + 1 + noise[idx]
            for idx in range(num)]

# 绘制散点图
d fig, ax = plt.subplots()
e ax.scatter(x_data, y_data)
f ax.set_xlabel('x'); ax.set_ylabel('y')
g ax.set_aspect('equal', adjustable='box')
h ax.set_xlim(0,10); ax.set_ylim(-2,8)
i ax.grid()

# 一元线性回归
j slope, intercept = statistics.linear_regression(x_data, y_data)

# 生成一个等差数列
start, end, step = 0, 10, 0.5
k x_array = []
x_i = start

while x_i <= end:
    x_array.append(x_i)
    x_i += step

# 计算x_array预测值
l y_array_predicted = [slope * x_i + intercept for x_i in x_array]

# 可视化一元线性回归直线
fig, ax = plt.subplots()
ax.scatter(x_data, y_data)
ax.plot(x_array, y_array_predicted, color = 'r')
ax.set_xlabel('x'); ax.set_ylabel('y')
ax.set_aspect('equal', adjustable='box')
ax.set_xlim(0,10); ax.set_ylim(-2,8)
ax.grid()
```

代码 14. 线性回归; Bk1_Ch06_12.ipynb

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

聊聊统计

“统计”这个词有两个字“统”和“计”。

“统”的意思是汇总，相当于折叠、总结、降维、压扁。某个特征上的样本数据实在太多，信息细节不再重要，我们把这个特征“压扁”。

“计”的意思是记录、量化、计算。也就是，汇总的结果是具体的数字，不能模糊。

单一特征样本数据量化汇总的方式有很多，比如**计数** (count)、**求和** (sum)、**均值** (mean 或 average)、**中位数** (median)、**四分位** (quartile)、**百分位** (percentile)、**最大值** (maximum)、**最小值** (minimum)、**方差** (variance)、**标准差** (standard deviation)、**偏度** (skewness)、**峰度** (kurtosis) 等等。

对于二特征、多特征样本数据，除了上述统计量之外，我们还可以用**协方差** (covariance)、**协方差矩阵** (covariance matrix)、**相关性系数** (correlation)、**相关性系数矩阵** (correlation matrix) 等量化特征之间的关系。这是本书后续要介绍的内容。

描述统计 (descriptive statistics) 是使用数字、图表和总结性信息对数据进行概括和简化的方法，以帮助理解数据的特征、趋势和分布，而不进行深入的统计分析或推断。

当然除了统计描述，我们同样关心**统计推断** (statistical inference)。比如，前文的线性回归就是统计推断的重要方法之一。



有关统计可视化，我们会在本书第 12 章、第 20 章、第 23 章专门介绍。

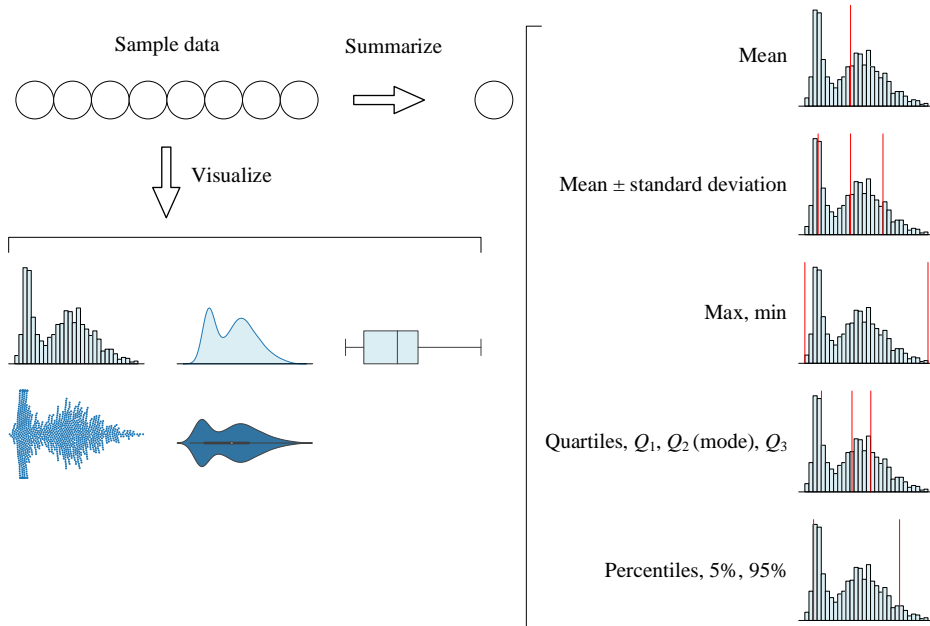


图 10. 统计描述，一元随机变量



请大家完成下面题目。

Q1. 首先请大家在 JupyterLab 中练习本章正文给出的示例代码。

Q2. 利用 `math` 库计算半径为 2 圆的周长和面积。

Q3. 利用 `math` 库绘制 $f(x) = \exp(x)$ 函数图像, x 的取值范围为 $[-3, 3]$ 。

Q4. 利用 `math` 库绘制高斯函数 $f(x) = \exp(-x^2)$ 图像, x 的取值范围为 $[-3, 3]$ 。

Q5. 利用 `random` 和 `statistics` 库函数完成如下实验。抛一枚质地均匀六面色子 1000 次, 色子点数为 $[1, 2, 3, 4, 5, 6]$ 。可视化均值随抛掷次数变化。

Q6. 利用 `random` 和 `statistics` 库函数完成如下实验。抛一枚质地不均匀色子 1000 次。点数 $[1, 2, 3, 4, 5, 6]$ 对应的概率分别为 $[0.2, 0.16, 0.16, 0.16, 0.16, 0.16]$ 。可视化点数均值随抛掷次数变化。

* 不提供答案。



本章首先介绍了几种常见 Python 运算符。需要大家注意的是各种运算符的优先级, 以及如何在本书后续的控制结构中使用比较、逻辑运算符。

此外, 本书最后介绍了三个 Python 内置库——`math`、`random`、`statistics`。书中例子可以帮助大家回顾很多重要数学概念, 并且利用可视化方案帮大家更直观理解。

继续学习本书后续内容, 大家会发现, 我们很少使用 `math`、`random`、`statistics`, 因为它们不方便向量化运算。Python 第三方库, 比如 `NumPy`、`SciPy`、`Pandas`、`Statsmodels` 提供了更方便、更高效、更丰富的运算函数。