# 练习答案

### 第2章

##### 练习2-2 多条简单消息

将一条消息存储到变量中，将其打印出来；再将变量的值修改为一条新消息，并将其打印出来。

msg **=** "I love learning to use Python."

**print**(msg)

msg **=** "It's really satisfying!"

**print**(msg)

输出：

I love learning to use Python.

It's really satisfying!

##### 练习2-5 名言

找位你钦佩的名人说的名言，将这个名人的姓名和他的名言打印出来。输出应类似于下面这样（包括引号）：

*Albert Einstein once said, “A person who never made a mistake never tried anything new.”*

**print**('Albert Einstein once said, "A person who never made a mistake')

**print**('never tried anything new."')

输出：

Albert Einstein once said, "A person who never made a mistake

never tried anything new."

##### 练习2-7 剔除人名中的空白

存储一个人名，并在其开头和末尾都包含一些空白字符。务必至少使用字符组合"\t"和"\n"各一次。

打印这个人名，以显示其开头和末尾的空白。然后，分别使用剔除函数lstrip()、rstrip()和strip()对人名进行处理，并将结果打印出来。

name **=** "\tEric Matthes\n"

**print**("Unmodified:")

**print**(name)

**print**("\nUsing lstrip():")

**print**(name**.**lstrip())

**print**("\nUsing rstrip():")

**print**(name**.**rstrip())

**print**("\nUsing strip():")

**print**(name**.**strip())

输出：

Unmodified:

Eric Matthes

Using lstrip():

Eric Matthes

Using rstrip():

Eric Matthes

Using strip():

Eric Matthes

##### 练习2-9 最喜欢的数

将你最喜欢的数存储在一个变量中，再使用这个变量创建一条消息，指出你最喜欢的数，然后将这条消息打印出来。

fav\_num **=** 42

msg **=** "My favorite number is " **+** str(fav\_num) **+** "."

**print**(msg)

输出：

My favorite number is 42.

### 第3章

##### 练习3-1 姓名

将一些朋友的姓名存储在一个列表中，并将其命名为names。依次访问该列表中的每个元素，从而将每个朋友的姓名都打印出来。

names **=** ['ron', 'tyler', 'dani']

**print**(names[0])

**print**(names[1])

**print**(names[2])

输出：

ron

tyler

dani

##### 练习3-2 问候语

继续使用练习3-1中的列表，但不打印每个朋友的姓名，而是打印一条消息。每条消息都包含相同的问候语，但抬头为相应朋友的姓名。

names **=** ['ron', 'tyler', 'dani']

msg **=** "Hello, " **+** names[0]**.**title() **+** "!"

**print**(msg)

msg **=** "Hello, " **+** names[1]**.**title() **+** "!"

**print**(msg)

msg **=** "Hello, " **+** names[2]**.**title() **+** "!"

**print**(msg)

输出：

Hello, Ron!

Hello, Tyler!

Hello, Dani!

##### 练习3-4 嘉宾名单

如果你可以邀请任何人一起共进晚餐（无论是故去的还是在世的），你会邀请哪些人？请创建一个列表，其中包含至少3个你想邀请的人；然后，使用这个列表打印消息，邀请这些人来与你共进晚餐。

guests **=** ['guido van rossum', 'jack turner', 'lynn hill']

name **=** guests[0]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

输出：

Guido Van Rossum, please come to dinner.

Jack Turner, please come to dinner.

Lynn Hill, please come to dinner.

##### 练习3-5 修改嘉宾名单

你刚得知有位嘉宾无法赴约，因此需要另外邀请一位嘉宾。

* 以你完成练习3-4时编写的程序为基础，在程序末尾添加一条print语句，指出哪位嘉宾无法赴约。
* 修改嘉宾名单，将无法赴约的嘉宾的姓名替换为新邀请的嘉宾的姓名。
* 再次打印一系列消息，将名单中的每位嘉宾发出邀请。

*# 邀请一些嘉宾与你共进晚餐。*

guests **=** ['guido van rossum', 'jack turner', 'lynn hill']

name **=** guests[0]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**("\nSorry, " **+** name **+** " can't make it to dinner.")

*# Jack无法赴约，转而邀请Gary。*

**del**(guests[1])

guests**.**insert(1, 'gary snyder')

*# 重新打印邀请函。*

name **=** guests[0]**.**title()

**print**("\n" **+** name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

输出：

Guido Van Rossum, please come to dinner.

Jack Turner, please come to dinner.

Lynn Hill, please come to dinner.

Sorry, Jack Turner can't make it to dinner.

Guido Van Rossum, please come to dinner.

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.

##### 练习3-6 添加嘉宾

你刚找到了一个更大的餐桌，可容纳更多的嘉宾。请想想你还想邀请哪三位嘉宾。

以你完成练习3-4或3-5时编写的程序为基础，在程序末尾添加一条print语句，指出你找到了一个更大的餐桌。

* 使用insert()将一位新嘉宾添加到名单开头。
* 使用insert()将另一位新嘉宾添加到名单中间。
* 使用append()将最后一位新嘉宾添加到名单末尾。
* 打印一系列消息，向名单中的每位嘉宾发出邀请。

*# 邀请一些嘉宾与你共进晚餐。*

guests **=** ['guido van rossum', 'jack turner', 'lynn hill']

name **=** guests[0]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**("\nSorry, " **+** name **+** " can't make it to dinner.")

*# Jack无法赴约，转而邀请Gary。*

**del**(guests[1])

guests**.**insert(1, 'gary snyder')

*# 重新打印邀请函。*

name **=** guests[0]**.**title()

**print**("\n" **+** name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

*# 找到了更大的餐桌，再邀请一些嘉宾。*

**print**("\nWe got a bigger table!")

guests**.**insert(0, 'frida kahlo')

guests**.**insert(2, 'reinhold messner')

guests**.**append('elizabeth peratrovich')

name **=** guests[0]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[3]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[4]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[5]**.**title()

**print**(name **+** ", please come to dinner.")

输出：

Guido Van Rossum, please come to dinner.

Jack Turner, please come to dinner.

Lynn Hill, please come to dinner.

Sorry, Jack Turner can't make it to dinner.

Guido Van Rossum, please come to dinner.

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.

We got a bigger table!

Frida Kahlo, please come to dinner.

Guido Van Rossum, please come to dinner.

Reinhold Messner, please come to dinner.

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.

Elizabeth Peratrovich, please come to dinner.

##### 练习3-7 缩减名单

你刚得知新购买的餐桌无法及时送达，因此只能邀请两位嘉宾。

* 以你完成练习3-6时编写的程序为基础，在程序末尾添加一行代码，打印一条你只能邀请两位嘉宾共进晚餐的消息。
* 使用pop()不断地删除名单中的嘉宾，直到只有两位嘉宾为止。每次从名单中弹出一位嘉宾时，都打印一条消息，让该嘉宾知悉你很抱歉，无法邀请他来共进晚餐。
* 对于余下的两位嘉宾中的每一位，都打印一条消息，指出他依然在邀请之列。
* 使用del将最后两位嘉宾从名单中删除，让名单变成空的。打印该名单，核实程序结束时名单确实是空的。

*# 邀请一些嘉宾与你共进晚餐。*

guests **=** ['guido van rossum', 'jack turner', 'lynn hill']

name **=** guests[0]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**("\nSorry, " **+** name **+** " can't make it to dinner.")

*# Jack无法赴约，转而邀请Gary。*

**del**(guests[1])

guests**.**insert(1, 'gary snyder')

*# 重新打印邀请函。*

name **=** guests[0]**.**title()

**print**("\n" **+** name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

*# 找到了更大的餐桌，再邀请一些嘉宾。*

**print**("\nWe got a bigger table!")

guests**.**insert(0, 'frida kahlo')

guests**.**insert(2, 'reinhold messner')

guests**.**append('elizabeth peratrovich')

name **=** guests[0]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[2]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[3]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[4]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[5]**.**title()

**print**(name **+** ", please come to dinner.")

*# 糟糕，餐桌无法及时送达！*

**print**("\nSorry, we can only invite two people to dinner.")

name **=** guests**.**pop()

**print**("Sorry, " **+** name**.**title() **+** " there's no room at the table.")

name **=** guests**.**pop()

**print**("Sorry, " **+** name**.**title() **+** " there's no room at the table.")

name **=** guests**.**pop()

**print**("Sorry, " **+** name**.**title() **+** " there's no room at the table.")

name **=** guests**.**pop()

**print**("Sorry, " **+** name**.**title() **+** " there's no room at the table.")

*# 应该只剩下两位嘉宾了，向他们发出邀请。*

name **=** guests[0]**.**title()

**print**(name **+** ", please come to dinner.")

name **=** guests[1]**.**title()

**print**(name **+** ", please come to dinner.")

*# 清空名单。*

**del**(guests[0])

**del**(guests[0])

*# 核实名单是空的。*

**print**(guests)

输出：

Guido Van Rossum, please come to dinner.

Jack Turner, please come to dinner.

Lynn Hill, please come to dinner.

Sorry, Jack Turner can't make it to dinner.

Guido Van Rossum, please come to dinner.

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.

We got a bigger table!

Frida Kahlo, please come to dinner.

Guido Van Rossum, please come to dinner.

Reinhold Messner, please come to dinner.

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.

Elizabeth Peratrovich, please come to dinner.

Sorry, we can only invite two people to dinner.

Sorry, Elizabeth Peratrovich there's no room at the table.

Sorry, Lynn Hill there's no room at the table.

Sorry, Gary Snyder there's no room at the table.

Sorry, Reinhold Messner there's no room at the table.

Frida Kahlo, please come to dinner.

Guido Van Rossum, please come to dinner.

[]

##### 练习3-8. 放眼世界

想出至少5个你渴望去旅游的地方。

* 将这些地方存储在一个列表中，并确保其中的元素不是按字母顺序排列的。
* 按原始排列顺序打印该列表。不要考虑输出是否整洁的问题，只管打印原始Python列表。
* 使用sorted()按字母顺序打印这个列表，同时不修改它。
* 再次打印该列表，核实排列顺序未变。
* 使用sorted()按与字母顺序相反的顺序打印这个列表，同时不修改它。
* 再次打印该列表，核实排列顺序未变。
* 使用reverse()修改列表元素的排列顺序。打印该列表，核实排列顺序确实变了。
* 使用reverse()再次修改列表元素的排列顺序。打印该列表，核实已恢复到原来的排列顺序。
* 使用sort()修改该列表，使其元素按字母顺序排列。打印该列表，核实排列顺序确实变了。
* 使用sort()修改该列表，使其元素按与字母顺序相反的顺序排列。打印该列表，核实排列顺序确实变了。

locations **=** ['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

**print**("Original order:")

**print**(locations)

**print**("\nAlphabetical:")

**print**(sorted(locations))

**print**("\nOriginal order:")

**print**(locations)

**print**("\nReverse alphabetical:")

**print**(sorted(locations, reverse**=**True))

**print**("\nOriginal order:")

**print**(locations)

**print**("\nReversed:")

locations**.**reverse()

**print**(locations)

**print**("\nOriginal order:")

locations**.**reverse()

**print**(locations)

**print**("\nAlphabetical")

locations**.**sort()

**print**(locations)

**print**("\nReverse alphabetical")

locations**.**sort(reverse**=**True)

**print**(locations)

输出：

Original order:

['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

Alphabetical:

['andes', 'guam', 'himalaya', 'labrador', 'tierra del fuego']

Original order:

['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

Reverse alphabetical:

['tierra del fuego', 'labrador', 'himalaya', 'guam', 'andes']

Original order:

['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

Reversed:

['guam', 'labrador', 'tierra del fuego', 'andes', 'himalaya']

Original order:

['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

Alphabetical

['andes', 'guam', 'himalaya', 'labrador', 'tierra del fuego']

Reverse alphabetical

['tierra del fuego', 'labrador', 'himalaya', 'guam', 'andes']

### 第4章

##### 练习4-1 比萨

想出至少三种你喜欢的比萨，将其名称存储在一个列表中，再使用for循环将每种比萨的名称都打印出来。

* 修改这个for循环，使其打印包含比萨名称的句子，而不仅仅是比萨的名称。对于每种比萨，都显示一行输出，如I like pepperoni pizza。
* 在程序末尾添加一行代码，它不在for循环中，指出你有多喜欢比萨。输出应包含针对每种比萨的消息，还有一个总结性句子，如I really love pizza!。

favorite\_pizzas **=** ['pepperoni', 'hawaiian', 'veggie']

*# 打印所有比萨的名称。*

**for** pizza **in** favorite\_pizzas:

**print**(pizza)

**print**("\n")

*# 针对每种比萨打印一个句子。*

**for** pizza **in** favorite\_pizzas:

**print**("I really love " **+** pizza **+** " pizza!")

**print**("\nI really love pizza!")

输出：

pepperoni

hawaiian

veggie

I really love pepperoni pizza!

I really love hawaiian pizza!

I really love veggie pizza!

I really love pizza!

##### 练习4-3 数到20

使用一个for循环打印数1~20（含）。

numbers **=** list(range(1, 21))

**for** number **in** numbers:

**print**(number)

输出：

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

##### 练习4-5 计算1～1000000的总和

创建一个列表，其中包含数1～1000000，再使用min()和max()核实该列表确实是从1开始，到100000结束的。另外，对这个列表调用函数sum()，看看Python将一百万个数字相加需要多长时间。

numbers **=** list(range(1, 1000001))

**print**(min(numbers))

**print**(max(numbers))

**print**(sum(numbers))

输出：

1

1000000

500000500000

##### 练习4-7 3的倍数

创建一个列表，其中包含3~30能被3整除的数；再使用一个for循环将这个列表中的数字都打印出来。

threes **=** list(range(3, 31, 3))

**for** number **in** threes:

**print**(number)

输出：

3

6

9

12

15

18

21

24

27

30

##### 练习4-8 立方

将同一个数字乘三次被称为立方。例如，在Python中，2的立方用2\*\*3表示。请创建一个列表，其中包含前10个整数（即1～10）的立方，再使用一个for循环将这些立方数都打印出来。

cubes **=** []

**for** number **in** range(1, 11):

cube **=** number**\*\***3

cubes**.**append(cube)

**for** cube **in** cubes:

**print**(cube)

输出：

1

8

27

64

125

216

343

512

729

1000

##### 练习4-9 立方解析

使用列表解析生成一个列表，其中包含前10个整数的立方。

cubes **=** [number**\*\***3 **for** number **in** range(1,11)]

**for** cube **in** cubes:

**print**(cube)

输出：

1

8

27

64

125

216

343

512

729

1000

##### 练习4-11 你的比萨和我的比萨

在你为完成练习4-1而编写的程序中，创建比萨列表的副本，并将其存储到变量friend\_pizzas中，再完成如下任务：

* 在原来的比萨列表中添加一种比萨。
* 在列表friend\_pizzas中添加另一种比萨。
* 核实你有两个不同的列表。为此，打印消息My favorite pizzas are:，再使用一个for循环打印第一个列表；打印消息My friend’s favorite pizzas are:，再使用一个for循环打印第二个列表。核实新增的比萨被添加到正确的列表中。

favorite\_pizzas **=** ['pepperoni', 'hawaiian', 'veggie']

friend\_pizzas **=** favorite\_pizzas[:]

favorite\_pizzas**.**append("meat lover's")

friend\_pizzas**.**append('pesto')

**print**("My favorite pizzas are:")

**for** pizza **in** favorite\_pizzas:

**print**("- " **+** pizza)

**print**("\nMy friend's favorite pizzas are:")

**for** pizza **in** friend\_pizzas:

**print**("- " **+** pizza)

输出：

My favorite pizzas are:

- pepperoni

- hawaiian

- veggie

- meat lover's

My friend's favorite pizzas are:

- pepperoni

- hawaiian

- veggie

- pesto

##### 练习4-13 自助餐

有一家自助式餐馆，只提供五种简单的食品。请想出五种简单的食品，并将其存储在一个元组中。

* 使用一个for循环将该餐馆提供的五种食品都打印出来。
* 尝试修改其中的一个元素，核实Python确实会拒绝你这样做。
* 餐馆调整了菜单，替换了它提供的其中两种食品。请编写一个这样的代码块：给元组变量赋值，并使用一个for循环将新元组的每个元素都打印出来。

menu\_items **=** (

'rockfish sandwich', 'halibut nuggets', 'smoked salmon chowder',

'salmon burger', 'crab cakes',

)

**print**("You can choose from the following menu items:")

**for** item **in** menu\_items:

**print**("- " **+** item)

menu\_items **=** (

'rockfish sandwich', 'halibut nuggets', 'smoked salmon chowder',

'black cod tips', 'king crab legs',

)

**print**("\nOur menu has been updated.")

**print**("You can now choose from the following items:")

**for** item **in** menu\_items:

**print**("- " **+** item)

输出：

You can choose from the following menu items:

- rockfish sandwich

- halibut nuggets

- smoked salmon chowder

- salmon burger

- crab cakes

Our menu has been updated.

You can now choose from the following items:

- rockfish sandwich

- halibut nuggets

- smoked salmon chowder

- black cod tips

- king crab legs

### 第5章

##### 练习5-3 外星人颜色#1

假设在游戏中刚射杀了一个外星人，请创建一个名为alien\_color的变量，并将其设置为'green'、'yellow'或'red'。

* 编写一条if语句，检查外星人是否是绿色的；如果是，就打印一条消息，指出玩家获得了5个点。
* 编写这个程序的两个版本，在一个版本中上述测试通过了，而在另一个版本中未通过（未通过测试时没有输出）。

能够通过测试的版本：

alien\_color **=** 'green'

**if** alien\_color **==** 'green':

**print**("You just earned 5 points!")

输出：

You just earned 5 points!

不能通过测试的版本：

alien\_color **=** 'red'

**if** alien\_color **==** 'green':

**print**("You just earned 5 points!")

（没有输出）

##### 练习5-4 外星人颜色#2

像练习5-3那样设置外星人的颜色，并编写一个if-else结构。

* 如果外星人是绿色的，就打印一条消息，指出玩家因射杀该外星人获得了5个点。
* 如果外星人不是绿色的，就打印一条消息，指出玩家获得了10个点。
* 编写这个程序的两个版本，在一个版本中执行if代码块，而在另一个版本中执行else代码块。

执行if代码块的版本：

alien\_color **=** 'green'

**if** alien\_color **==** 'green':

**print**("You just earned 5 points!")

**else**:

**print**("You just earned 10 points!")

输出：

You just earned 5 points!

执行else代码块的版本：

alien\_color **=** 'yellow'

**if** alien\_color **==** 'green':

**print**("You just earned 5 points!")

**else**:

**print**("You just earned 10 points!")

输出：

You just earned 10 points!

##### 练习5-5 外星人颜色#3

将练习5-4中的if-else结构改为if-elif-else结构。

* 如果外星人是绿色的，就打印一条消息，指出玩家获得了5个点。
* 如果外星人是黄色的，就打印一条消息，指出玩家获得了10个点。
* 如果外星人是红色的，就打印一条消息，指出玩家获得了15个点。
* 编写这个程序的三个版本，它们分别在外星人为绿色、黄色和红色时打印一条消息。

外星人为红色的版本：

alien\_color **=** 'red'

**if** alien\_color **==** 'green':

**print**("You just earned 5 points!")

**elif** alien\_color **==** 'yellow':

**print**("You just earned 10 points!")

**else**:

**print**("You just earned 15 points!")

输出：

You just earned 15 points!

##### 练习5-6 人生的不同阶段

设置变量age的值，再编写一个if-elif-else结构，根据age的值判断处于人生的哪个阶段：

* 如果一个人的年龄小于2岁，就打印一条消息，指出他是婴儿。
* 如果一个人的年龄为2（含）~4岁，就打印一条消息，指出他正蹒跚学步（toddler）。
* 如果一个人的年龄为4（含）~13岁，就打印一条消息，指出他是儿童。
* 如果一个人的年龄为13（含）~20岁，就打印一条消息，指出他是少年。
* 如果一个人的年龄为20（含）~65岁，就打印一条消息，指出他是成年人。
* 如果一个人的年龄超过65（含）岁，就打印一条消息，指出他是老年人。

age **=** 17

**if** age **<** 2:

**print**("You're a baby!")

**elif** age **<** 4:

**print**("You're a toddler!")

**elif** age **<** 13:

**print**("You're a kid!")

**elif** age **<** 20:

**print**("You're a teenager!")

**elif** age **<** 65:

**print**("You're an adult!")

**else**:

**print**("You're an elder!")

输出：

You're a teenager!

##### 练习5-7 喜欢的水果

创建一个列表，其中包含你喜欢的水果，再编写一系列独立的if语句，检查列表中是否包含特定的水果。

* 将该列表命名为favorite\_fruits，并在其中包含三种水果。
* 编写5条if语句，每条都检查某种水果是否包含在列表中，如果包含在列表中，就打印一条消息，如You really like bananas!。

favorite\_fruits **=** ['blueberries', 'salmonberries', 'peaches']

**if** 'bananas' **in** favorite\_fruits:

**print**("You really like bananas!")

**if** 'apples' **in** favorite\_fruits:

**print**("You really like apples!")

**if** 'blueberries' **in** favorite\_fruits:

**print**("You really like blueberries!")

**if** 'kiwis' **in** favorite\_fruits:

**print**("You really like kiwis!")

**if** 'peaches' **in** favorite\_fruits:

**print**("You really like peaches!")

输出：

You really like blueberries!

You really like peaches!

##### 练习5-8 以特殊方式跟管理员打招呼

创建一个至少包含5个用户名的列表，且其中一个用户名为'admin'。想象你要编写代码，在每位用户登录网站后都打印一条问候消息。遍历用户名列表，并向每位用户打印一条问候消息：

* 如果用户名为'admin'，就打印一条特殊的问候消息，如Hello admin, would you like to see a status report?。
* 否则，打印一条普通的问候消息，如Hello Eric, thank you for logging in again。

usernames **=** ['eric', 'willie', 'admin', 'erin', 'ever']

**for** username **in** usernames:

**if** username **==** 'admin':

**print**("Hello admin, would you like to see a status report?")

**else**:

**print**("Hello " **+** username **+** ", thank you for logging in again!")

输出：

Hello eric, thank you for logging in again!

Hello willie, thank you for logging in again!

Hello admin, would you like to see a status report?

Hello erin, thank you for logging in again!

Hello ever, thank you for logging in again!

练习5-9 处理没有用户的情形

在为完成练习5-8编写的程序中，添加一条if语句，检查用户名列表是否为空。

* 如果为空，就打印消息We need to find some users!。
* 删除列表中的所有用户名，核实将打印正确的消息。

usernames **=** []

**if** usernames:

**for** username **in** usernames:

**if** username **==** 'admin':

**print**("Hello admin, would you like to see a status report?")

**else**:

**print**("Hello " **+** username **+** ", thank you for logging in again!")

**else**:

**print**("We need to find some users!")

输出：

We need to find some users!

##### 练习5-10 检查用户名

按下面说的编写一个程序，模拟网站确保每位用户的用户名都独一无二的方式。

* 创建一个至少包含5个用户名的列表，并将其命名为current\_users。
* 再创建一个包含5个用户名的列表，将其命名为new\_users，并确保其中有一两个用户名也包含在列表current\_users中。
* 遍历列表new\_users，对于其中的每个用户名，都检查它是否已被使用。如果是这样，就打印一条消息，指出需要输入别的用户名；否则，打印一条消息，指出这个用户名未被使用。
* 确保比较是不区分大消息；换句话说，如果用户名'John'已被使用，应拒绝用户名'JOHN'。

current\_users **=** ['eric', 'willie', 'admin', 'erin', 'Ever']

new\_users **=** ['sarah', 'Willie', 'PHIL', 'ever', 'Iona']

current\_users\_lower **=** [user**.**lower() **for** user **in** current\_users]

**for** new\_user **in** new\_users:

**if** new\_user**.**lower() **in** current\_users\_lower:

**print**("Sorry " **+** new\_user **+** ", that name is taken.")

**else**:

**print**("Great, " **+** new\_user **+** " is still available.")

输出：

Great, sarah is still available.

Sorry Willie, that name is taken.

Great, PHIL is still available.

Sorry ever, that name is taken.

Great, Iona is still available.

注意：如果你还不熟悉列表解析，可像下面这样使用循环来生成列表current\_users\_lower：

current\_users\_lower **=** []

**for** user **in** current\_users:

current\_users\_lower**.**append(user**.**lower())

##### 练习5-11 序数

序数表示位置，如1st和2nd。大多数序数都以th结尾，只有1、2和3例外。

在一个列表中存储数字1~9。

遍历这个列表。

在循环中使用一个if-elif-else结构，以打印每个数字对应的序数。输出内容应为1st、2nd、3rd、4th、5th、6th、7th、8th和9th，但每个序数都独占一行。

numbers **=** list(range(1,10))

**for** number **in** numbers:

**if** number **==** 1:

**print**("1st")

**elif** number **==** 2:

**print**("2nd")

**elif** number **==** 3:

**print**("3rd")

**else**:

**print**(str(number) **+** "th")

输出：

1st

2nd

3rd

4th

5th

6th

7th

8th

9th

### 第6章

##### 练习6-1 人

使用一个字典来存储一个熟人的信息，包括名、姓、年龄和居住的城市。该字典应包含键first\_name、last\_name、age和city。将存储在该字典中的每项信息都打印出来。

person **=** {

'first\_name': 'eric',

'last\_name': 'matthes',

'age': 43,

'city': 'sitka',

}

**print**(person['first\_name'])

**print**(person['last\_name'])

**print**(person['age'])

**print**(person['city'])

输出：

eric

matthes

43

sitka

##### 练习6-2 喜欢的数

使用一个字典来存储一些人们喜欢的数。请想出5个人的名字，并将这些名字用作字典中的键；想出每个人喜欢的一个数，并将这些数作为值存储在字典中。打印每个人的名字和喜欢的数。为让这个程序更有趣，通过询问朋友确保数据是真实的。

favorite\_numbers **=** {

'mandy': 42,

'micah': 23,

'gus': 7,

'hank': 1000000,

'maggie': 0,

}

num **=** favorite\_numbers['mandy']

**print**("Mandy's favorite number is " **+** str(num) **+** ".")

num **=** favorite\_numbers['micah']

**print**("Micah's favorite number is " **+** str(num) **+** ".")

num **=** favorite\_numbers['gus']

**print**("Gus's favorite number is " **+** str(num) **+** ".")

num **=** favorite\_numbers['hank']

**print**("Hank's favorite number is " **+** str(num) **+** ".")

num **=** favorite\_numbers['maggie']

**print**("Maggie's favorite number is " **+** str(num) **+** ".")

输出：

Mandy's favorite number is 42.

Micah's favorite number is 23.

Gus's favorite number is 7.

Hank's favorite number is 1000000.

Maggie's favorite number is 0.

##### 练习6-3 词汇表

Python字典可用于模拟现实生活中的字典，但为避免混淆，我们将后者称为词汇表。

* 想出你在前面学过的5个编程词汇，将它们用作词汇表中的键，并将它们的含义作为值存储在词汇表中。
* 以整洁的方式打印每个词汇及其含义。为此，你可以先打印词汇，在它后面加上一个冒号，再打印词汇的含义；也可在一行打印词汇，再使用换行符（\n）插入一个空行，然后在下一行以缩进的方式打印词汇的含义。

glossary **=** {

'string': 'A series of characters.',

'comment': 'A note in a program that the Python interpreter ignores.',

'list': 'A collection of items in a particular order.',

'loop': 'Work through a collection of items, one at a time.',

'dictionary': "A collection of key-value pairs.",

}

word **=** 'string'

**print**("\n" **+** word**.**title() **+** ": " **+** glossary[word])

word **=** 'comment'

**print**("\n" **+** word**.**title() **+** ": " **+** glossary[word])

word **=** 'list'

**print**("\n" **+** word**.**title() **+** ": " **+** glossary[word])

word **=** 'loop'

**print**("\n" **+** word**.**title() **+** ": " **+** glossary[word])

word **=** 'dictionary'

**print**("\n" **+** word**.**title() **+** ": " **+** glossary[word])

输出：

String: A series of characters.

Comment: A note in a program that the Python interpreter ignores.

List: A collection of items in a particular order.

Loop: Work through a collection of items, one at a time.

Dictionary: A collection of key-value pairs.

##### 练习6-4 词汇表2

现在你知道了如何遍历字典，请整理你为完成练习6.3而编写的代码，将其中的一系列print语句替换为一个遍历字典中键和值的循环。确定该循环正确无误后，再在词汇表中添加5个Python术语。当你再次运行这个程序时，这些新术语及其含义将自动包含在输出中。

glossary **=** {

'string': 'A series of characters.',

'comment': 'A note in a program that the Python interpreter ignores.',

'list': 'A collection of items in a particular order.',

'loop': 'Work through a collection of items, one at a time.',

'dictionary': "A collection of key-value pairs.",

'key': 'The first item in a key-value pair in a dictionary.',

'value': 'An item associated with a key in a dictionary.',

'conditional test': 'A comparison between two values.',

'float': 'A numerical value with a decimal component.',

'boolean expression': 'An expression that evaluates to True or False.',

}

**for** word, definition **in** glossary**.**items():

**print**("\n" **+** word**.**title() **+** ": " **+** definition)

输出：

Dictionary: A collection of key-value pairs.

String: A series of characters.

Boolean Expression: An expression that evaluates to True or False.

Comment: A note in a program that the Python interpreter ignores.

Value: An item associated with a key in a dictionary.

Loop: Work through a collection of items, one at a time.

List: A collection of items in a particular order.

Conditional Test: A comparison between two values.

Key: The first item in a key-value pair in a dictionary.

Float: A numerical value with a decimal component.

##### 练习6-5 河流

河流：创建一个字典，在其中存储三条大河流及其流经的国家。其中一个键-值对可能是'nile': 'egypt'。

* 使用循环为每条河流打印一条消息，如The Nile runs through Egypt。
* 使用循环将该字典中每条河流的名字都打印出来。
* 使用循环将该字典包含的每个国家的名字都打印出来。

rivers **=** {

'nile': 'egypt',

'mississippi': 'united states',

'fraser': 'canada',

'kuskokwim': 'alaska',

'yangtze': 'china',

}

**for** river, country **in** rivers**.**items():

**print**("The " **+** river**.**title() **+** " flows through " **+** country**.**title() **+** ".")

**print**("\nThe following rivers are included in this data set:")

**for** river **in** rivers**.**keys():

**print**("- " **+** river**.**title())

**print**("\nThe following countries are included in this data set:")

**for** country **in** rivers**.**values():

**print**("- " **+** country**.**title())

输出：

The Mississippi flows through United States.

The Yangtze flows through China.

The Fraser flows through Canada.

The Nile flows through Egypt.

The Kuskokwim flows through Alaska.

The following rivers are included in this data set:

- Mississippi

- Yangtze

- Fraser

- Nile

- Kuskokwim

The following countries are included in this data set:

- United States

- China

- Canada

- Egypt

- Alaska

注意：有些人将阿拉斯加视为美国的一个附属国。

##### 练习6-6 调查

在6.3.1节编写的程序favorite\_languages.py中：

* 创建一个应该会接受调查的人员名单，其中有些人已包含在字典中，而其他人未包含在字典中。
* 遍历这个人员名单，对于已参与调查的人，打印一条消息表示感谢。对于还未参与调查的人，打印一条消息邀请他参与调查。

favorite\_languages **=** {

'jen': 'python',

'sarah': 'c',

'edward': 'ruby',

'phil': 'python',

}

**for** name, language **in** favorite\_languages**.**items():

**print**(name**.**title() **+** "'s favorite language is " **+**

language**.**title() **+** ".")

**print**("\n")

coders **=** ['phil', 'josh', 'david', 'becca', 'sarah', 'matt', 'danielle']

**for** coder **in** coders:

**if** coder **in** favorite\_languages**.**keys():

**print**("Thank you for taking the poll, " **+** coder**.**title() **+** "!")

**else**:

**print**(coder**.**title() **+** ", what's your favorite programming language?")

输出：

Jen's favorite language is Python.

Sarah's favorite language is C.

Phil's favorite language is Python.

Edward's favorite language is Ruby.

Thank you for taking the poll, Phil!

Josh, what's your favorite programming language?

David, what's your favorite programming language?

Becca, what's your favorite programming language?

Thank you for taking the poll, Sarah!

Matt, what's your favorite programming language?

Danielle, what's your favorite programming language?

##### 练习6-7 人

在为完成练习6.1而编写的程序中，再创建两个表示人的字典，然后将这三个字典都存储在一个名为people的列表中。遍历这个列表，将其中每个人的所有信息都打印出来。

*# 创建一个用于存储人的空列表。*

people **=** []

*# 定义一些人并将他们添加到前述列表中。*

person **=** {

'first\_name': 'eric',

'last\_name': 'matthes',

'age': 43,

'city': 'sitka',

}

people**.**append(person)

person **=** {

'first\_name': 'ever',

'last\_name': 'matthes',

'age': 5,

'city': 'sitka',

}

people**.**append(person)

person **=** {

'first\_name': 'willie',

'last\_name': 'matthes',

'age': 8,

'city': 'sitka',

}

people**.**append(person)

*# 显示列表包含的每个字典中的信息。*

**for** person **in** people:

name **=** person['first\_name']**.**title() **+** " " **+** person['last\_name']**.**title()

age **=** str(person['age'])

city **=** person['city']**.**title()

**print**(name **+** ", of " **+** city **+** ", is " **+** age **+** " years old.")

输出：

Eric Matthes, of Sitka, is 43 years old.

Ever Matthes, of Sitka, is 5 years old.

Willie Matthes, of Sitka, is 8 years old.

##### 练习6-8 宠物

创建多个字典，对于每个字典，都使用一个宠物的名称来给它命名；在每个字典中，包含宠物的类型及其主人的名字。将这些字典存储在一个名为pets的列表中，再遍历该列表，并将有关每个宠物的所有信息都打印出来。

*# 创建一个用于存储宠物的空列表。*

pets **=** []

*# 定义各个宠物并将其存储到列表中。*

pet **=** {

'animal type': 'python',

'name': 'john',

'owner': 'guido',

'weight': 43,

'eats': 'bugs',

}

pets**.**append(pet)

pet **=** {

'animal type': 'chicken',

'name': 'clarence',

'owner': 'tiffany',

'weight': 2,

'eats': 'seeds',

}

pets**.**append(pet)

pet **=** {

'animal type': 'dog',

'name': 'peso',

'owner': 'eric',

'weight': 37,

'eats': 'shoes',

}

pets**.**append(pet)

*# 显示每个宠物的信息。*

**for** pet **in** pets:

**print**("\nHere's what I know about " **+** pet['name']**.**title() **+** ":")

**for** key, value **in** pet**.**items():

**print**("\t" **+** key **+** ": " **+** str(value))

输出：

Here's what I know about John:

weight: 43

animal type: python

name: john

owner: guido

eats: bugs

Here's what I know about Clarence:

weight: 2

animal type: chicken

name: clarence

owner: tiffany

eats: seeds

Here's what I know about Peso:

weight: 37

animal type: dog

name: peso

owner: eric

eats: shoes

##### 练习6-9 喜欢的地方

创建一个名为favorite\_places的字典。在这个字典中，将三个人的名字用作键；对于其中的每个人，都存储他喜欢的1-3个地方。为让这个练习更有趣些，让一些朋友指出他们喜欢的几个地方。遍历这个字典，并将其中每个人的名字及其喜欢的地方打印出来。

favorite\_places **=** {

'eric': ['bear mountain', 'death valley', 'tierra del fuego'],

'erin': ['hawaii', 'iceland'],

'ever': ['mt. verstovia', 'the playground', 'south carolina']

}

**for** name, places **in** favorite\_places**.**items():

**print**("\n" **+** name**.**title() **+** " likes the following places:")

**for** place **in** places:

**print**("- " **+** place**.**title())

输出：

Ever likes the following places:

- Mt. Verstovia

- The Playground

- South Carolina

Erin likes the following places:

- Hawaii

- Iceland

Eric likes the following places:

- Bear Mountain

- Death Valley

- Tierra Del Fuego

##### 练习6-10 喜欢的数

修改为完成练习6-2而编写的程序，让每个人都可以有多个喜欢的数，然后将每个人的名字及其喜欢的数打印出来。

favorite\_numbers **=** {

'mandy': [42, 17],

'micah': [42, 39, 56],

'gus': [7, 12],

}

**for** name, numbers **in** favorite\_numbers**.**items():

**print**("\n" **+** name**.**title() **+** " likes the following numbers:")

**for** number **in** numbers:

**print**(" " **+** str(number))

输出：

Micah likes the following numbers:

42

39

56

Mandy likes the following numbers:

42

17

Gus likes the following numbers:

7

12

##### 练习6-11 城市

创建一个名为cities的字典，在其中将三个城市名用作键；对于每座城市，都创建一个字典，并在其中包含该城市所属的国家、人口约数以及一个有关该城市的事实。在表示每座城市的字典中，应包含country、population和fact等键。将每座城市的名字以及有关它们的信息都打印出来。

cities **=** {

'santiago': {

'country': 'chile',

'population': 6158080,

'nearby mountains': 'andes',

},

'talkeetna': {

'country': 'alaska',

'population': 876,

'nearby mountains': 'alaska range',

},

'kathmandu': {

'country': 'nepal',

'population': 1003285,

'nearby mountains': 'himilaya',

}

}

**for** city, city\_info **in** cities**.**items():

country **=** city\_info['country']**.**title()

population **=** city\_info['population']

mountains **=** city\_info['nearby mountains']**.**title()

**print**("\n" **+** city**.**title() **+** " is in " **+** country **+** ".")

**print**(" It has a population of about " **+** str(population) **+** ".")

**print**(" The " **+** mountains **+** " mountains are nearby.")

输出：

Santiago is in Chile.

It has a population of about 6158080.

The Andes mountains are nearby.

Kathmandu is in Nepal.

It has a population of about 1003285.

The Himilaya mountains are nearby.

Talkeetna is in Alaska.

It has a population of about 876.

The Alaska Range mountains are nearby.

### 第7章

注意：Sublime Text不能运行提示用户输入的程序。你可使用Sublime Text来编写提示用户输入的程序，但必须从终端运行它们。详情请参阅1.4节。

##### 练习7-1 汽车租赁

编写一个程序，询问用户要租赁什么样的汽车，并打印一条消息，如“Let me see if I can find you a Subaru.”。

car **=** input("What kind of car would you like? ")

**print**("Let me see if I can find you a " **+** car**.**title() **+** ".")

输出：

What kind of car would you like? **Toyota Tacoma**

Let me see if I can find you a Toyota Tacoma.

##### 练习7-2 餐馆订位

编写一个程序，询问用户有多少人用餐。如果超过8位，就打印一条消息，指出没有空桌；否则就指出有空桌。

party\_size **=** input("How many people are in your dinner party tonight? ")

party\_size **=** int(party\_size)

**if** party\_size **>** 8:

**print**("I'm sorry, you'll have to wait for a table.")

**else**:

**print**("Your table is ready.")

输出：

How many people are in your dinner party tonight? **12**

I'm sorry, you'll have to wait for a table.

或：

How many people are in your dinner party tonight? **6**

Your table is ready.

##### 练习7-3 10的整数倍

让用户输入一个数字，并指出这个数字是否是10的整数倍。

number **=** input("Give me a number, please: ")

number **=** int(number)

**if** number **%** 10 **==** 0:

**print**(str(number) **+** " is a multiple of 10.")

**else**:

**print**(str(number) **+** " is not a multiple of 10.")

输出：

Give me a number, please: **23**

23 is not a multiple of 10.

或：

Give me a number, please: **90**

90 is a multiple of 10.

##### 练习7-4 比萨配料

编写一个循环，提示用户输入一系列的比萨配料，并在用户输入'quit'时结束循环。每当用户输入一种配料后，都打印一条消息，说我们会在比萨中添加这种配料。

prompt **=** "\nWhat topping would you like on your pizza?"

prompt **+=** "\nEnter 'quit' when you are finished: "

**while** True:

topping **=** input(prompt)

**if** topping **!=** 'quit':

**print**(" I'll add " **+** topping **+** " to your pizza.")

**else**:

**break**

输出：

What topping would you like on your pizza?

Enter 'quit' when you are finished: **pepperoni**

I'll add pepperoni to your pizza.

What topping would you like on your pizza?

Enter 'quit' when you are finished: **sausage**

I'll add sausage to your pizza.

What topping would you like on your pizza?

Enter 'quit' when you are finished: **bacon**

I'll add bacon to your pizza.

What topping would you like on your pizza?

Enter 'quit' when you are finished: **quit**

##### 练习7-5 电影票

有家电影院根据观众的年龄收取不同的票价：不到3岁的观众免费；3~12岁的观众为10美元；超过12岁的观众为15美元。请编写一个循环，在其中询问用户的年龄，并指出其票价。

prompt **=** "How old are you?"

prompt **+=** "\nEnter 'quit' when you are finished. "

**while** True:

age **=** input(prompt)

**if** age **==** 'quit':

**break**

age **=** int(age)

**if** age **<** 3:

**print**(" You get in free!")

**elif** age **<** 13:

**print**(" Your ticket is $10.")

**else**:

**print**(" Your ticket is $15.")

输出：

How old are you?

Enter 'quit' when you are finished. **2**

You get in free!

How old are you?

Enter 'quit' when you are finished. **3**

Your ticket is $10.

How old are you?

Enter 'quit' when you are finished. **12**

Your ticket is $10.

How old are you?

Enter 'quit' when you are finished. **18**

Your ticket is $15.

How old are you?

Enter 'quit' when you are finished. **quit**

##### 练习7-8 熟食店

创建一个名为sandwich\_orders的列表，在其中包含各种三明治的名字；再创建一个名为finished\_sandwiches的空列表。遍历列表sandwich\_orders，对于其中的每种三明治，都打印一条消息，如I made your tuna sandwich，并将其移到列表finished\_sandwiches。所有三明治都制作好后，打印一条消息，将这些三明治列出来。

sandwich\_orders **=** ['veggie', 'grilled cheese', 'turkey', 'roast beef']

finished\_sandwiches **=** []

**while** sandwich\_orders:

current\_sandwich **=** sandwich\_orders**.**pop()

**print**("I'm working on your " **+** current\_sandwich **+** " sandwich.")

finished\_sandwiches**.**append(current\_sandwich)

**print**("\n")

**for** sandwich **in** finished\_sandwiches:

**print**("I made a " **+** sandwich **+** " sandwich.")

输出：

I'm working on your roast beef sandwich.

I'm working on your turkey sandwich.

I'm working on your grilled cheese sandwich.

I'm working on your veggie sandwich.

I made a roast beef sandwich.

I made a turkey sandwich.

I made a grilled cheese sandwich.

I made a veggie sandwich.

##### 练习7-9 五香烟熏牛肉卖完了

使用为完成练习7-8而创建的列表sandwich\_orders，并确保'pastrami'在其中至少出现了三次。在程序开头附近添加这样的代码：打印一条消息，指出熟食店的五香烟熏牛肉（pastrami）卖完了；再使用一个while循环将列表sandwich\_orders中的'pastrami'都删除。确认最终的列表finished\_sandwiches中未包含'pastrami'。

sandwich\_orders **=** [

'pastrami', 'veggie', 'grilled cheese', 'pastrami',

'turkey', 'roast beef', 'pastrami']

finished\_sandwiches **=** []

**print**("I'm sorry, we're all out of pastrami today.")

**while** 'pastrami' **in** sandwich\_orders:

sandwich\_orders**.**remove('pastrami')

**print**("\n")

**while** sandwich\_orders:

current\_sandwich **=** sandwich\_orders**.**pop()

**print**("I'm working on your " **+** current\_sandwich **+** " sandwich.")

finished\_sandwiches**.**append(current\_sandwich)

**print**("\n")

**for** sandwich **in** finished\_sandwiches:

**print**("I made a " **+** sandwich **+** " sandwich.")

输出：

I'm sorry, we're all out of pastrami today.

I'm working on your roast beef sandwich.

I'm working on your turkey sandwich.

I'm working on your grilled cheese sandwich.

I'm working on your veggie sandwich.

I made a roast beef sandwich.

I made a turkey sandwich.

I made a grilled cheese sandwich.

I made a veggie sandwich.

##### 练习7-10 梦想的度假胜地

编写一个程序，调查用户梦想的度假胜地。使用类似于“If you could visit one place in the world, where would you go?”的提示，并编写一个打印调查结果的代码块。

name\_prompt **=** "\nWhat's your name? "

place\_prompt **=** "If you could visit one place in the world, where would it be? "

continue\_prompt **=** "\nWould you like to let someone else respond? (yes/no) "

*# 调查结果将存储在形如{name: place}的字典中。*

responses **=** {}

**while** True:

*# 询问用户想去哪里度假。*

name **=** input(name\_prompt)

place **=** input(place\_prompt)

*# 存储调查结果。*

responses[name] **=** place

*# 询问是否还有其他人要参与调查。*

repeat **=** input(continue\_prompt)

**if** repeat **!=** 'yes':

**break**

*# 显示调查结果。*

**print**("\n--- Results ---")

**for** name, place **in** responses**.**items():

**print**(name**.**title() **+** " would like to visit " **+** place**.**title() **+** ".")

输出：

What's your name? **eric**

If you could visit one place in the world, where would it be? **tierra del fuego**

Would you like to let someone else respond? (yes/no) **yes**

What's your name? **erin**

If you could visit one place in the world, where would it be? **iceland**

Would you like to let someone else respond? (yes/no) **yes**

What's your name? **ever**

If you could visit one place in the world, where would it be? **death valley**

Would you like to let someone else respond? (yes/no) **no**

--- Results ---

Ever would like to visit Death Valley.

Erin would like to visit Iceland.

Eric would like to visit Tierra Del Fuego.

### 第8章

##### 练习8-1 消息

编写一个名为display\_message()的函数，它打印一个句子，指出你在本章学的是什么。调用这个函数，确认显示的消息正确无误。

**def** **display\_message**():

"""显示一条消息，指出你在本章学的是什么。"""

msg **=** "I'm learning to store code in functions."

**print**(msg)

display\_message()

输出：

I'm learning to store code in functions.

##### 练习8-2 喜欢的图书

编写一个名为favorite\_book()的函数，其中包含一个名为title的形参。这个函数打印一条消息，如One of my favorite books is Alice in Wonderland。调用这个函数，并将一本图书的名称作为实参传递给它。

**def** **favorite\_book**(title):

"""显示一条消息，指出喜欢的一本图书。"""

**print**(title **+** " is one of my favorite books.")

favorite\_book('The Abstract Wild')

输出：

The Abstract Wild is one of my favorite books.

练习8-3 T恤

编写一个名为make\_shirt()的函数，它接受一个尺码以及要印到T恤上的字样。这个函数应打印一个句子，概要地说明T恤衫的尺码和字样。

使用位置实参调用这个函数来制作一件T恤；再使用关键字实参来调用这个函数。

**def** **make\_shirt**(size, message):

"""指出要制作的T恤什么样。"""

**print**("\nI'm going to make a " **+** size **+** " t-shirt.")

**print**('It will say, "' **+** message **+** '"')

make\_shirt('large', 'I love Python!')

make\_shirt(message**=**"Readability counts.", size**=**'medium')

输出：

I'm going to make a large t-shirt.

It will say, "I love Python!"

I'm going to make a medium t-shirt.

It will say, "Readability counts."

##### 练习8-4 大号T恤

修改函数make\_shirt()，使其在默认情况下制作一件印有字样I love Python的大号T恤。调用这个函数来制作如下T恤：一件印有默认字样的大号T恤、一件印有默认字样的中号T恤、一件印有其他字样的T恤（尺码无关紧要）。

**def** **make\_shirt**(size**=**'large', message**=**'I love Python!'):

"""指出要制作的T恤什么样。"""

**print**("\nI'm going to make a " **+** size **+** " t-shirt.")

**print**('It will say, "' **+** message **+** '"')

make\_shirt()

make\_shirt(size**=**'medium')

make\_shirt('small', 'Programmers are loopy.')

输出：

I'm going to make a large t-shirt.

It will say, "I love Python!"

I'm going to make a medium t-shirt.

It will say, "I love Python!"

I'm going to make a small t-shirt.

It will say, "Programmers are loopy."

##### 练习8-5 城市

编写一个名为describe\_city()的函数，它接受一座城市的名字以及该城市所属的国家。这个函数应打印一个简单的句子，如Reykjavik is in Iceland。给用于存储国家的形参指定默认值。为三座不同的城市调用这个函数，且其中至少有一座城市不属于默认国家。

**def** **describe\_city**(city, country**=**'chile'):

"""描绘城市。"""

msg **=** city**.**title() **+** " is in " **+** country**.**title() **+** "."

**print**(msg)

describe\_city('santiago')

describe\_city('reykjavik', 'iceland')

describe\_city('punta arenas')

输出：

Santiago is in Chile.

Reykjavik is in Iceland.

Punta Arenas is in Chile.

##### 练习8-6 城市名

编写一个名为city\_country()的函数，它接受城市的名称及其所属的国家。这个函数应返回一个格式类似于下面的字符串：

“Santiago, Chile”

至少使用三个城市-国家对调用这个函数，并打印它返回的值。

**def** **city\_country**(city, country):

"""返回一个类似于'Santiago, Chile'的字符串。"""

**return**(city**.**title() **+** ", " **+** country**.**title())

city **=** city\_country('santiago', 'chile')

**print**(city)

city **=** city\_country('ushuaia', 'argentina')

**print**(city)

city **=** city\_country('longyearbyen', 'svalbard')

**print**(city)

输出：

Santiago, Chile

Ushuaia, Argentina

Longyearbyen, Svalbard

##### 练习8-7 专辑

编写一个名为make\_album()的函数，它创建一个描述音乐专辑的字典。这个函数应接受歌手的名字和专辑名，并返回一个包含这两项信息的字典。使用这个函数创建三个表示不同专辑的字典，并打印每个返回的值，以核实字典正确地存储了专辑的信息。

给函数make\_album()添加一个可选形参，以便能够存储专辑包含的歌曲数。如果调用这个函数时指定了歌曲数，就将这个值添加到表示专辑的字典中。调用这个函数，并至少在一次调用中指定专辑包含的歌曲数。

简单版：

**def** **make\_album**(artist, title):

"""创建一个包含专辑信息的字典。"""

album\_dict **=** {

'artist': artist**.**title(),

'title': title**.**title(),

}

**return** album\_dict

album **=** make\_album('metallica', 'ride the lightning')

**print**(album)

album **=** make\_album('beethoven', 'ninth symphony')

**print**(album)

album **=** make\_album('willie nelson', 'red-headed stranger')

**print**(album)

输出：

{'title': 'Ride The Lightning', 'artist': 'Metallica'}

{'title': 'Ninth Symphony', 'artist': 'Beethoven'}

{'title': 'Red-Headed Stranger', 'artist': 'Willie Nelson'}

包含歌曲数的版本：

**def** **make\_album**(artist, title, tracks**=**0):

"""创建一个包含专辑信息的字典。"""

album\_dict **=** {

'artist': artist**.**title(),

'title': title**.**title(),

}

**if** tracks:

album\_dict['tracks'] **=** tracks

**return** album\_dict

album **=** make\_album('metallica', 'ride the lightning')

**print**(album)

album **=** make\_album('beethoven', 'ninth symphony')

**print**(album)

album **=** make\_album('willie nelson', 'red-headed stranger')

**print**(album)

album **=** make\_album('iron maiden', 'piece of mind', tracks**=**8)

**print**(album)

输出：

{'artist': 'Metallica', 'title': 'Ride The Lightning'}

{'artist': 'Beethoven', 'title': 'Ninth Symphony'}

{'artist': 'Willie Nelson', 'title': 'Red-Headed Stranger'}

{'tracks': 8, 'artist': 'Iron Maiden', 'title': 'Piece Of Mind'}

##### 练习8-8 用户的专辑

在为完成练习8-7编写的程序中，编写一个while循环，让用户输入一个专辑的歌手和名称。获取这些信息后，使用它们来调用函数make\_album()并将创建的字典打印出来。在这个while循环中，务必要提供退出途径。

**def** **make\_album**(artist, title, tracks**=**0):

"""创建一个包含专辑信息的字典。"""

album\_dict **=** {

'artist': artist**.**title(),

'title': title**.**title(),

}

**if** tracks:

album\_dict['tracks'] **=** tracks

**return** album\_dict

*# 生成提示语。*

title\_prompt **=** "\nWhat album are you thinking of? "

artist\_prompt **=** "Who's the artist? "

*# 让用户知道如何退出。*

**print**("Enter 'quit' at any time to stop.")

**while** True:

title **=** input(title\_prompt)

**if** title **==** 'quit':

**break**

artist **=** input(artist\_prompt)

**if** artist **==** 'quit':

**break**

album **=** make\_album(artist, title)

**print**(album)

**print**("\nThanks for responding!")

输出：

Enter 'quit' at any time to stop.

What album are you thinking of? **number of the beast**

Who's the artist? **iron maiden**

{'artist': 'Iron Maiden', 'title': 'Number Of The Beast'}

What album are you thinking of? **touch of class**

Who's the artist? **angel romero**

{'artist': 'Angel Romero', 'title': 'Touch Of Class'}

What album are you thinking of? **rust in peace**

Who's the artist? **megadeth**

{'artist': 'Megadeth', 'title': 'Rust In Peace'}

What album are you thinking of? **quit**

Thanks for responding!

##### 练习8-9 魔术师

创建一个包含魔术师名字的列表，将其传递给一个名为show\_magicians()的函数，这个函数打印列表中每个魔术师的名字。

**def** **show\_magicians**(magicians):

"""打印列表中每个魔术师的名字。"""

**for** magician **in** magicians:

**print**(magician**.**title())

magicians **=** ['harry houdini', 'david blaine', 'teller']

show\_magicians(magicians)

输出：

Harry Houdini

David Blaine

Teller

##### 练习8-10 了不起的魔术师

在你为完成练习8-9而编写的程序中，编写一个名为make\_great()的函数，对魔术师列表进行修改，在每个魔术师的名字中都加入字样the Great。调用函数show\_magicians()，确认魔术师列表确实变了。

""""""

**def** **show\_magicians**(magicians):

"""打印列表中每个魔术师的名字。"""

**for** magician **in** magicians:

**print**(magician)

**def** **make\_great**(magicians):

"""在每个魔术师的名字中都加入字样the Great。"""

*# 新建一个列表，用于存储修改后的魔术师名字。*

great\_magicians **=** []

*# 在每个魔术师的名字中都加入字样the Great，并将其添加到great\_magicians中。*

**while** magicians:

magician **=** magicians**.**pop()

great\_magician **=** magician **+** ' the Great'

great\_magicians**.**append(great\_magician)

*# 将修改后的名字放回到列表magicians中。*

**for** great\_magician **in** great\_magicians:

magicians**.**append(great\_magician)

magicians **=** ['Harry Houdini', 'David Blaine', 'Teller']

show\_magicians(magicians)

**print**("\n")

make\_great(magicians)

show\_magicians(magicians)

输出：

Harry Houdini

David Blaine

Teller

Teller the Great

David Blaine the Great

Harry Houdini the Great

##### 练习8-11 不变的魔术师

修改你为完成练习8-10而编写的程序，在调用函数make\_great()时，向它传递魔术师列表的拷贝。由于不想修改原始列表，请返回修改后的列表，并将其存储到另一个列表中。分别使用这两个列表来调用show\_magicians()，确认一个列表包含的是原来的魔术师名字，而另一个列表包含的是添加了the Great字样的魔术师名字。

**def** **show\_magicians**(magicians):

"""打印列表中每个魔术师的名字。"""

**for** magician **in** magicians:

**print**(magician)

**def** **make\_great**(magicians):

"""在每个魔术师的名字中都加入字样the Great。"""

*# 新建一个列表，用于存储修改后的魔术师名字。*

great\_magicians **=** []

*# 在每个魔术师的名字中都加入字样the Great，并将其添加到great\_magicians中。*

**while** magicians:

magician **=** magicians**.**pop()

great\_magician **=** magician **+** ' the Great'

great\_magicians**.**append(great\_magician)

*# 将修改后的名字放回到列表magicians中。*

**for** great\_magician **in** great\_magicians:

magicians**.**append(great\_magician)

**return** magicians

magicians **=** ['Harry Houdini', 'David Blaine', 'Teller']

show\_magicians(magicians)

**print**("\nGreat magicians:")

great\_magicians **=** make\_great(magicians[:])

show\_magicians(great\_magicians)

**print**("\nOriginal magicians:")

show\_magicians(magicians)

输出：

Harry Houdini

David Blaine

Teller

Great magicians:

Teller the Great

David Blaine the Great

Harry Houdini the Great

Original magicians:

Harry Houdini

David Blaine

Teller

##### 练习8-12 三明治

编写一个函数，它接受顾客要在三明治中添加的一系列食材。这个函数只有一个形参（它收集函数调用中提供的所有食材），并打印一条消息，对顾客点的三明治进行概述。调用这个函数三次，每次都提供不同数量的实参。

**def** **make\_sandwich**(**\***items):

"""使用指定的食材制作三明治。"""

**print**("\nI'll make you a great sandwich:")

**for** item **in** items:

**print**(" ...adding " **+** item **+** " to your sandwich.")

**print**("Your sandwich is ready!")

make\_sandwich('roast beef', 'cheddar cheese', 'lettuce', 'honey dijon')

make\_sandwich('turkey', 'apple slices', 'honey mustard')

make\_sandwich('peanut butter', 'strawberry jam')

输出：

I'll make you a great sandwich:

...adding roast beef to your sandwich.

...adding cheddar cheese to your sandwich.

...adding lettuce to your sandwich.

...adding honey dijon to your sandwich.

Your sandwich is ready!

I'll make you a great sandwich:

...adding turkey to your sandwich.

...adding apple slices to your sandwich.

...adding honey mustard to your sandwich.

Your sandwich is ready!

I'll make you a great sandwich:

...adding peanut butter to your sandwich.

...adding strawberry jam to your sandwich.

Your sandwich is ready!

##### 练习8-14 汽车

编写一个函数，将一辆汽车的信息存储在一个字典中。这个函数总是接受制造商和型号，还接受任意数量的关键字实参。这样调用这个函数：提供必不可少的信息，还有两个名称-值对，如颜色和选装配件。这个函数必须能够像下面这样进行调用：

car = make\_car('subaru', 'outback', color='blue', tow\_package=True)

打印返回的字典，确认正确地处理了所有的信息。

**def** **make\_car**(manufacturer, model, **\*\***options):

"""创建一个表示汽车的字典。"""

car\_dict **=** {

'manufacturer': manufacturer**.**title(),

'model': model**.**title(),

}

**for** option, value **in** options**.**items():

car\_dict[option] **=** value

**return** car\_dict

my\_outback **=** make\_car('subaru', 'outback', color**=**'blue', tow\_package**=**True)

**print**(my\_outback)

my\_accord **=** make\_car('honda', 'accord', year**=**1991, color**=**'white',

headlights**=**'popup')

**print**(my\_accord)

输出：

{'manufacturer': 'Subaru', 'color': 'blue', 'tow\_package': True, 'model': 'Outback'}

{'year': 1991, 'manufacturer': 'Honda', 'color': 'white', 'headlights': 'popup', 'model': 'Accord'}

##### 练习8-15 打印模型

打印模型：将示例printing\_models.py中的函数放在另一个名为printing\_functions.py的文件中；在printing\_models.py的开头编写一条import语句，并修改这个文件以使用导入的函数。

printing\_functions.py:

"""与打印3D模型相关的函数。"""

**def** **print\_models**(unprinted\_designs, completed\_models):

"""

模拟打印每个设计，直到没有未打印的设计为止。

打印每个设计后，都将其移到列表completed\_models中。

"""

**while** unprinted\_designs:

current\_design **=** unprinted\_designs**.**pop()

*# 模拟根据设计制作3D打印模型的过程。*

**print**("Printing model: " **+** current\_design)

completed\_models**.**append(current\_design)

**def** **show\_completed\_models**(completed\_models):

"""显示打印好的所有模型。"""

**print**("\nThe following models have been printed:")

**for** completed\_model **in** completed\_models:

**print**(completed\_model)

printing\_models.py:

import printing\_functions **as** pf

unprinted\_designs **=** ['iphone case', 'robot pendant', 'dodecahedron']

completed\_models **=** []

pf**.**print\_models(unprinted\_designs, completed\_models)

pf**.**show\_completed\_models(completed\_models)

输出：

Printing model: dodecahedron

Printing model: robot pendant

Printing model: iphone case

The following models have been printed:

dodecahedron

robot pendant

iphone case

### 第9章

##### 练习9-1 餐馆

创建一个名为Restaurant类，其方法\_\_init\_\_()设置两个属性：restaurant\_name和cuisine\_type。创建一个名为describe\_restaurant()的方法和一个名为open\_restaurant()的方法，其中前者打印前述两项信息，而后者打印一条消息，指出餐馆正在营业。

根据这个类创建一个名为restaurant的实例，分别打印其两个属性，再调用前述两个方法。

**class** **Restaurant**():

"""一个表示餐馆的类。"""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""初始化餐馆。"""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

**def** **describe\_restaurant**(self):

"""显示餐馆信息摘要。"""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""显示一条消息，指出餐馆正在营业。"""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

restaurant **=** Restaurant('the mean queen', 'pizza')

**print**(restaurant**.**name)

**print**(restaurant**.**cuisine\_type)

restaurant**.**describe\_restaurant()

restaurant**.**open\_restaurant()

输出：

The Mean Queen

pizza

The Mean Queen serves wonderful pizza.

The Mean Queen is open. Come on in!

##### 练习9-2 三家餐馆

根据你为完成练习9-1而编写类创建三个实例，并对每个实例调用方法describe\_restaurant()。

**class** **Restaurant**():

"""一个表示餐馆的类。"""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""初始化餐馆。"""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

**def** **describe\_restaurant**(self):

"""显示餐馆信息摘要概述。"""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""显示一条消息，指出餐馆正在营业。"""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

mean\_queen **=** Restaurant('the mean queen', 'pizza')

mean\_queen**.**describe\_restaurant()

ludvigs **=** Restaurant("ludvig's bistro", 'seafood')

ludvigs**.**describe\_restaurant()

mango\_thai **=** Restaurant('mango thai', 'thai food')

mango\_thai**.**describe\_restaurant()

输出：

The Mean Queen serves wonderful pizza.

Ludvig'S Bistro serves wonderful seafood.

Mango Thai serves wonderful thai food.

##### 练习9-3 用户

创建一个名为User的类，其中包含属性first\_name和last\_name，还有用户简介通常会存储的其他几个属性。在类User中定义一个名为describe\_user()的方法，它打印用户信息摘要；再定义一个名为greet\_user()的方法，它向用户发出个性化的问候。

创建多个表示不同用户的实例，并对每个实例都调用上述两个方法。

**class** **User**():

"""一个表示用户的简单类。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化用户。"""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

**def** **describe\_user**(self):

"""显示用户信息摘要。"""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""它向用户发出个性化的问候。"""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

eric **=** User('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric**.**greet\_user()

willie **=** User('willie', 'burger', 'willieburger', 'wb@example.com', 'alaska')

willie**.**describe\_user()

willie**.**greet\_user()

输出：

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

Welcome back, e\_matthes!

Willie Burger

Username: willieburger

Email: wb@example.com

Location: Alaska

Welcome back, willieburger!

##### 练习9-4 就餐人数

在为完成练习9-1而编写的程序中，添加一个名为number\_served的属性，并将其默认值设置为0。根据这个类创建一个名为restaurant的实例；打印有多少人在这家餐馆就餐过，然后修改这个值并再次打印它。

添加一个名为set\_number\_served()的方法，它让你能够设置就餐人数。调用这个方法并向它传递一个值，然后再次打印这个值。

添加一个名为increment\_number\_served()的方法，它让你能够增加就餐人数。调用这个方法并向它传递一个这样的值：你认为这家餐馆每天可能接待的就餐人数。

**class** **Restaurant**():

"""一个表示餐馆的类。"""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""初始化餐馆。"""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

self**.**number\_served **=** 0

**def** **describe\_restaurant**(self):

"""显示餐馆信息摘要。"""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""显示一条消息，指出餐馆正在营业。"""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

**def** **set\_number\_served**(self, number\_served):

"""让用户能够设置就餐人数。"""

self**.**number\_served **=** number\_served

**def** **increment\_number\_served**(self, additional\_served):

"""让用户能够增加就餐人数。"""

self**.**number\_served **+=** additional\_served

restaurant **=** Restaurant('the mean queen', 'pizza')

restaurant**.**describe\_restaurant()

**print**("\nNumber served: " **+** str(restaurant**.**number\_served))

restaurant**.**number\_served **=** 430

**print**("Number served: " **+** str(restaurant**.**number\_served))

restaurant**.**set\_number\_served(1257)

**print**("Number served: " **+** str(restaurant**.**number\_served))

restaurant**.**increment\_number\_served(239)

**print**("Number served: " **+** str(restaurant**.**number\_served))

输出：

The Mean Queen serves wonderful pizza.

Number served: 0

Number served: 430

Number served: 1257

Number served: 1496

##### 练习9-5 尝试登录次数

在为完成练习9-3而编写的User类中，添加一个名为login\_attempts的属性。编写一个名为increment\_login\_attempts()的方法，它将属性login\_attempts的值加1。再编写一个名为reset\_login\_attempts()的方法，它将属性login\_ attempts的值重置为0。

根据User类创建一个实例，再调用方法increment\_login\_attempts()多次。打印属性login\_attempts的值，确认它被正确地递增；然后，调用方法reset\_login\_attempts()，并再次打印属性login\_attempts的值，确认它被重置为0。

**class** **User**():

"""一个表示用户的简单类。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化用户"""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""显示用户信息摘要。"""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""向用户发出个性化问候。"""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""将属性login\_attempts的值加1。"""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""将login\_attempts重置为0。"""

self**.**login\_attempts **=** 0

eric **=** User('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric**.**greet\_user()

**print**("\nMaking 3 login attempts...")

eric**.**increment\_login\_attempts()

eric**.**increment\_login\_attempts()

eric**.**increment\_login\_attempts()

**print**(" Login attempts: " **+** str(eric**.**login\_attempts))

**print**("Resetting login attempts...")

eric**.**reset\_login\_attempts()

**print**(" Login attempts: " **+** str(eric**.**login\_attempts))

输出：

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

Welcome back, e\_matthes!

Making 3 login attempts...

Login attempts: 3

Resetting login attempts...

Login attempts: 0

##### 练习9-6 冰淇淋小店

冰淇淋小店是一种特殊的餐馆。编写一个名为IceCreamStand的类，让它继承你为完成练习9-1或9-4而编写的Restaurant类。这两个版本的Restaurant类都可以，挑选你更喜欢的那个即可。添加一个名为flavors的属性，用于存储一个由各种口味的冰淇淋组成的列表。编写一个显示这些冰淇淋的方法。创建一个IceCreamStand实例，并调用这个方法。

**class** **Restaurant**():

"""一个表示餐馆的类。"""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""初始化餐馆。"""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

self**.**number\_served **=** 0

**def** **describe\_restaurant**(self):

"""显示餐馆信息摘要。"""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""显示一条消息，指出餐馆正在营业。"""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

**def** **set\_number\_served**(self, number\_served):

"""让用户能够设置就餐人数。"""

self**.**number\_served **=** number\_served

**def** **increment\_number\_served**(self, additional\_served):

"""让用户能够增加就餐人数。"""

self**.**number\_served **+=** additional\_served

**class** **IceCreamStand**(Restaurant):

"""一个表示冰淇淋小店的类。"""

**def** **\_\_init\_\_**(self, name, cuisine\_type**=**'ice\_cream'):

"""初始化冰淇淋小店。"""

super()**.**\_\_init\_\_(name, cuisine\_type)

self**.**flavors **=** []

**def** **show\_flavors**(self):

"""显示出售的冰淇淋品种。"""

**print**("\nWe have the following flavors available:")

**for** flavor **in** self**.**flavors:

**print**("- " **+** flavor**.**title())

big\_one **=** IceCreamStand('The Big One')

big\_one**.**flavors **=** ['vanilla', 'chocolate', 'black cherry']

big\_one**.**describe\_restaurant()

big\_one**.**show\_flavors()

输出：

The Big One serves wonderful ice\_cream.

We have the following flavors available:

- Vanilla

- Chocolate

- Black Cherry

##### 练习9-7 管理员

管理员是一种特殊的用户。编写一个名为Admin的类，让它继承你为完成练习9-3或9-5而编写的User 类。添加一个名为privileges的属性，用于存储一个由字符串（如"can add post"、"can delete post"、"can ban user"等）组成的列表。编写一个名为show\_privileges()的方法，它显示管理员的权限。创建一个Admin实例，并调用这个方法。

**class** **User**():

"""一个表示用户的简单类。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化用户。"""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""显示用户信息摘要。"""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""向用户发出个性化问候。"""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""将属性login\_attempts的值加1。"""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""将login\_attempts重置为0。"""

self**.**login\_attempts **=** 0

**class** **Admin**(User):

"""有管理权限的用户。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化管理员。"""

super()**.**\_\_init\_\_(first\_name, last\_name, username, email, location)

self**.**privileges **=** []

**def** **show\_privileges**(self):

"""显示当前管理员的权限。"""

**print**("\nPrivileges:")

**for** privilege **in** self**.**privileges:

**print**("- " **+** privilege)

eric **=** Admin('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric**.**privileges **=** [

'can reset passwords',

'can moderate discussions',

'can suspend accounts',

]

eric**.**show\_privileges()

输出：

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

Privileges:

- can reset passwords

- can moderate discussions

- can suspend accounts

##### 练习9-8 权限

编写一个名为Privileges的类，它只有一个属性——privileges，其中存储了练习9-7所说的字符串列表。将方法show\_privileges()移到这个类中。在Admin类中，将一个Privileges实例用作其属性。创建一个Admin实例，并使用方法show\_privileges()来显示其权限。

**class** **User**():

"""一个表示用户的简单类。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化用户。"""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""显示用户信息摘要。"""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""向用户发出个性化问候。"""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""将属性login\_attempts的值加1。"""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""将login\_attempts重置为0。"""

self**.**login\_attempts **=** 0

**class** **Admin**(User):

"""有管理权限的用户。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化管理员。"""

super()**.**\_\_init\_\_(first\_name, last\_name, username, email, location)

*# 将权限集初始化为空。*

self**.**privileges **=** Privileges()

**class** **Privileges**():

"""一个存储管理员权限的类。"""

**def** **\_\_init\_\_**(self, privileges**=**[]):

self**.**privileges **=** privileges

**def** **show\_privileges**(self):

**print**("\nPrivileges:")

**if** self**.**privileges:

**for** privilege **in** self**.**privileges:

**print**("- " **+** privilege)

**else**:

**print**("- This user has no privileges.")

eric **=** Admin('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric**.**privileges**.**show\_privileges()

**print**("\nAdding privileges...")

eric\_privileges **=** [

'can reset passwords',

'can moderate discussions',

'can suspend accounts',

]

eric**.**privileges**.**privileges **=** eric\_privileges

eric**.**privileges**.**show\_privileges()

输出：

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

Privileges:

- This user has no privileges.

Adding privileges...

Privileges:

- can reset passwords

- can moderate discussions

- can suspend accounts

##### 练习9-9 电瓶升级

在本节最后一个electric\_car.py版本中，给Battery类添加一个名为upgrade\_battery()的方法。这个方法检查电瓶容量，如果它不是85，就将它设置为85。创建一辆电瓶容量为默认值的电动汽车，调用方法get\_range()，然后对电瓶进行升级，并再次调用get\_range()。你将看到这辆汽车的续航里程增加了。

**class** **Car**():

"""一次模拟汽车的简单尝试。"""

**def** **\_\_init\_\_**(self, manufacturer, model, year):

"""初始化描述汽车的属性。"""

self**.**manufacturer **=** manufacturer

self**.**model **=** model

self**.**year **=** year

self**.**odometer\_reading **=** 0

**def** **get\_descriptive\_name**(self):

"""返回整洁的描述性信息。"""

long\_name **=** str(self**.**year) **+** ' ' **+** self**.**manufacturer **+** ' ' **+** self**.**model

**return** long\_name**.**title()

**def** **read\_odometer**(self):

"""打印一条指出汽车里程的消息。"""

**print**("This car has " **+** str(self**.**odometer\_reading) **+** " miles on it.")

**def** **update\_odometer**(self, mileage):

"""

将里程表读数设置为指定的值。

禁止将里程表读数往回调。

"""

**if** mileage **>=** self**.**odometer\_reading:

self**.**odometer\_reading **=** mileage

**else**:

**print**("You can't roll back an odometer!")

**def** **increment\_odometer**(self, miles):

"""将里程表读数增加指定的量。"""

self**.**odometer\_reading **+=** miles

**class** **Battery**():

"""一次模拟电动汽车电瓶的简单尝试。"""

**def** **\_\_init\_\_**(self, battery\_size**=**60):

"""初始化电瓶的属性。"""

self**.**battery\_size **=** battery\_size

**def** **describe\_battery**(self):

"""打印一条描述电瓶容量的消息。"""

**print**("This car has a " **+** str(self**.**battery\_size) **+** "-kWh battery.")

**def** **get\_range**(self):

"""打印一条消息，指出电瓶的续航里程。"""

**if** self**.**battery\_size **==** 60:

range **=** 140

**elif** self**.**battery\_size **==** 85:

range **=** 185

message **=** "This car can go approximately " **+** str(range)

message **+=** " miles on a full charge."

**print**(message)

**def** **upgrade\_battery**(self):

"""在可能的情况下将电瓶升级。"""

**if** self**.**battery\_size **==** 60:

self**.**battery\_size **=** 85

**print**("Upgraded the battery to 85 kWh.")

**else**:

**print**("The battery is already upgraded.")

**class** **ElectricCar**(Car):

"""电动汽车的独特之处。"""

**def** **\_\_init\_\_**(self, manufacturer, model, year):

"""

初始化父类的属性，再初始化电动汽车特有的属性。

"""

super()**.**\_\_init\_\_(manufacturer, model, year)

self**.**battery **=** Battery()

**print**("Make an electric car, and check the battery:")

my\_tesla **=** ElectricCar('tesla', 'model s', 2016)

my\_tesla**.**battery**.**describe\_battery()

**print**("\nUpgrade the battery, and check it again:")

my\_tesla**.**battery**.**upgrade\_battery()

my\_tesla**.**battery**.**describe\_battery()

**print**("\nTry upgrading the battery a second time.")

my\_tesla**.**battery**.**upgrade\_battery()

my\_tesla**.**battery**.**describe\_battery()

输出：

Make an electric car, and check the battery:

This car has a 60-kWh battery.

Upgrade the battery, and check it again:

Upgraded the battery to 85 kWh.

This car has a 85-kWh battery.

Try upgrading the battery a second time.

The battery is already upgraded.

This car has a 85-kWh battery.

##### 练习9-10 导入Restaurant类

将最新的Restaurant类存储在一个模块中。在另一个文件中，导入Restaurant类、创建一个Restaurant实例并调用Restaurant的一个方法，以确认import语句正确无误。

*restaurant.py:*

"""一个表示餐馆的类。"""

**class** **Restaurant**():

"""一个表示餐馆的类。"""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""初始化餐馆。"""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

self**.**number\_served **=** 0

**def** **describe\_restaurant**(self):

"""显示餐馆信息摘要。"""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""显示一条消息，指出餐馆正在营业。"""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

**def** **set\_number\_served**(self, number\_served):

"""让用户能够设置就餐人数。"""

self**.**number\_served **=** number\_served

**def** **increment\_number\_served**(self, additional\_served):

"""让用户能够增加就餐人数。"""

self**.**number\_served **+=** additional\_served

*my\_restaurant.py:*

from restaurant import Restaurant

channel\_club **=** Restaurant('the channel club', 'steak and seafood')

channel\_club**.**describe\_restaurant()

channel\_club**.**open\_restaurant()

输出：

The Channel Club serves wonderful steak and seafood.

The Channel Club is open. Come on in!

练习9-11 导入Admin类

以为完成练习9-8而做的工作为基础。将User、Privileges和Admin类存储在一个模块中，再创建一个文件，在其中创建一个Admin实例并对其调用方法show\_privileges()，以确认一切都能正确地运行。

*user.py:*

"""一系列模拟用户的类。"""

**class** **User**():

"""一个表示用户的简单类。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化用户。"""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""显示用户信息摘要。"""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""向用户发出个性化问候。"""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""将属性login\_attempts的值加1。"""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""将login\_attempts重置为0。"""

self**.**login\_attempts **=** 0

**class** **Admin**(User):

"""有管理权限的用户。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化管理员。"""

super()**.**\_\_init\_\_(first\_name, last\_name, username, email, location)

*# 将权限集初始化为空。*

self**.**privileges **=** Privileges([])

**class** **Privileges**():

"""存储管理员的权限。"""

**def** **\_\_init\_\_**(self, privileges):

"""初始化权限对象。"""

self**.**privilege **=** privileges

**def** **show\_privileges**(self):

"""显示当前管理员的权限。"""

**for** privilege **in** self**.**privileges:

**print**("- " **+** privilege)

*my\_user.py:*

from user import Admin

eric **=** Admin('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric\_privileges **=** [

'can reset passwords',

'can moderate discussions',

'can suspend accounts',

]

eric**.**privileges**.**privileges **=** eric\_privileges

**print**("\nThe admin " **+** eric**.**username **+** " has these privileges: ")

eric**.**privileges**.**show\_privileges()

输出：

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

The admin e\_matthes has these privileges:

- can reset passwords

- can moderate discussions

- can suspend accounts

##### 练习9-12 多个模块

将User类存储在一个模块中，并将Privileges和Admin类存储在另一个模块中。再创建一个文件，在其中创建一个Admin实例并对其调用方法show\_privileges()，以确认一切都依然能够正确地运行。

*user.py:*

"""一个模拟用户的类。"""

**class** **User**():

"""一个表示用户的简单类。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化用户。"""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""显示用户信息摘要。"""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""向用户发出个性化问候。"""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""将属性login\_attempts的值加1。"""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""将login\_attempts重置为0。"""

self**.**login\_attempts **=** 0

*admin.py:*

"""一系列模拟管理员的类。"""

from user import User

**class** **Admin**(User):

"""有管理权限的用户。"""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""初始化管理员。"""

super()**.**\_\_init\_\_(first\_name, last\_name, username, email, location)

*# 将权限集初始化为空。*

self**.**privileges **=** Privileges([])

**class** **Privileges**():

"""存储管理员的权限。"""

**def** **\_\_init\_\_**(self, privileges):

"""初始化权限对象。"""

self**.**privilege **=** privileges

**def** **show\_privileges**(self):

"""显示当前管理员的权限。"""

**for** privilege **in** self**.**privileges:

**print**("- " **+** privilege)

*my\_admin.py*

from admin import Admin

eric **=** Admin('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric\_privileges **=** [

'can reset passwords',

'can moderate discussions',

'can suspend accounts',

]

eric**.**privileges**.**privileges **=** eric\_privileges

**print**("\nThe admin " **+** eric**.**username **+** " has these privileges: ")

eric**.**privileges**.**show\_privileges()

输出：

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

The admin e\_matthes has these privileges:

- can reset passwords

- can moderate discussions

- can suspend accounts

##### 练习9-13 使用OrderedDict

在练习6-4中，你使用了一个标准字典来表示词汇表，请使用OrderedDict类来重写这个程序，并确认输出的顺序与你在字典中添加键-值对的顺序一致。

注意：在Python 3.6中，字典按顺序存储键，但并非在所有Python版本中都如此。因此要以特定顺序存储键-值对，应使用OrderedDict。

from collections import OrderedDict

glossary **=** OrderedDict()

glossary['string'] **=** 'A series of characters.'

glossary['comment'] **=** 'A note in a program that the Python interpreter ignores.'

glossary['list'] **=** 'A collection of items in a particular order.'

glossary['loop'] **=** 'Work through a collection of items, one at a time.'

glossary['dictionary'] **=** "A collection of key-value pairs."

glossary['key'] **=** 'The first item in a key-value pair in a dictionary.'

glossary['value'] **=** 'An item associated with a key in a dictionary.'

glossary['conditional test'] **=** 'A comparison between two values.'

glossary['float'] **=** 'A numerical value with a decimal component.'

glossary['boolean expression'] **=** 'An expression that evaluates to True or False.'

**for** word, definition **in** glossary**.**items():

**print**("\n" **+** word**.**title() **+** ": " **+** definition)

输出：

String: A series of characters.

Comment: A note in a program that the Python interpreter ignores.

List: A collection of items in a particular order.

Loop: Work through a collection of items, one at a time.

Dictionary: A collection of key-value pairs.

Key: The first item in a key-value pair in a dictionary.

Value: An item associated with a key in a dictionary.

Conditional Test: A comparison between two values.

Float: A numerical value with a decimal component.

Boolean Expression: An expression that evaluates to True or False.

##### 练习9-14 骰子

模块random包含以各种方式生成随机数的函数，其中的randint()返回一个位于指定范围内的整数，例如，下面的代码返回一个1~6的整数：

from random import randint

x **=** randint(1, 6)

请创建一个Die类，它包含一个名为sides的属性，该属性的默认值为6。编写一个名为roll\_die()的方法，它打印位于1和骰子面数之间的随机数。创建一个6面的骰子再掷10次。。

创建一个10面的骰子和一个20面的骰子，并对它们都掷10次。

"""一个表示骰子的类。"""

**def** **\_\_init\_\_**(self, sides**=**6):

"""初始化骰子。"""

self**.**sides **=** sides

**def** **roll\_die**(self):

"""返回一个位于1和骰子面数之间的随机数。"""

**return** randint(1, self**.**sides)

*# 创建一个6面的骰子，再掷10次并显示结果。*

d6 **=** Die()

results **=** []

**for** roll\_num **in** range(10):

result **=** d6**.**roll\_die()

results**.**append(result)

**print**("10 rolls of a 6-sided die:")

**print**(results)

*# 创建一个10面的骰子，再掷10次并显示结果。*

d10 **=** Die(sides**=**10)

results **=** []

**for** roll\_num **in** range(10):

result **=** d10**.**roll\_die()

results**.**append(result)

**print**("\n10 rolls of a 10-sided die:")

**print**(results)

*# 创建一个20面的骰子，再掷10次并显示结果。*

d20 **=** Die(sides**=**20)

results **=** []

**for** roll\_num **in** range(10):

result **=** d20**.**roll\_die()

results**.**append(result)

**print**("\n10 rolls of a 20-sided die:")

**print**(results)

输出：

10 rolls of a 6-sided die:

[5, 5, 6, 3, 6, 4, 2, 2, 1, 1]

10 rolls of a 10-sided die:

[8, 9, 8, 10, 7, 1, 3, 5, 3, 4]

10 rolls of a 20-sided die:

[4, 3, 18, 17, 3, 1, 13, 12, 5, 14]

### 第10章

##### 练习10-1 Python学习笔记

在文本编辑器中新建一个文件，写几句话来总结一下你至此学到的Python知识，其中每一行都以In Python you can打头。将这个文件命名为learning\_python.txt，并将其存储到为完成本章练习而编写的程序所在的目录中。编写一个程序，它读取这个文件，并将你所写的内容打印三次：第一次打印时读取整个文件；第二次打印时遍历文件对象；第三次打印时将各行存储在一个列表中，再在with代码块外打印它们。

*learning\_python.txt:*

In Python you can store as much information as you want.

In Python you can connect pieces of information.

In Python you can model real-world situations.

*learning\_python.py:*

filename **=** 'learning\_python.txt'

**print**("--- Reading in the entire file:")

**with** open(filename) **as** f:

contents **=** f**.**read()

**print**(contents)

**print**("\n--- Looping over the lines:")

**with** open(filename) **as** f:

**for** line **in** f:

**print**(line**.**rstrip())

**print**("\n--- Storing the lines in a list:")

**with** open(filename) **as** f:

lines **=** f**.**readlines()

**for** line **in** lines:

**print**(line**.**rstrip())

输出：

--- Reading in the entire file:

In Python you can store as much information as you want.

In Python you can connect pieces of information.

In Python you can model real-world situations.

--- Looping over the lines:

In Python you can store as much information as you want.

In Python you can connect pieces of information.

In Python you can model real-world situations.

--- Storing the lines in a list:

In Python you can store as much information as you want.

In Python you can connect pieces of information.

In Python you can model real-world situations.

##### 练习10-2 C语言学习笔记

你可使用方法replace()将字符串中的特定单词都替换为另一个单词。下面是一个简单的示例，演示了如何将句子中的'dog'替换为'cat'：

**>>>** message **=** "I really like dogs."

**>>>** message**.**replace('dog', 'cat')

'I really like cats.'

读取你刚创建的文件learning\_python.txt中的每一行，将其中的Python都替换为另一门语言的名称，如C。将修改后的各行都打印到屏幕上。

filename **=** 'learning\_python.txt'

**with** open(filename) **as** f:

lines **=** f**.**readlines()

**for** line **in** lines:

*# 删除末尾的换行符，再将Python替换为C。*

line **=** line**.**rstrip()

**print**(line**.**replace('Python', 'C'))

输出：

In C you can store as much information as you want.

In C you can connect pieces of information.

In C you can model real-world situations.

可在一行代码中依次调用rstrip()和replace()，这被称为方法串接。在下面的代码中，删除行尾的换行符再将Python替换为C，输出与前面的代码相同。

filename **=** 'learning\_python.txt'

**with** open(filename) **as** f:

lines **=** f**.**readlines()

**for** line **in** lines:

*# 删除末尾的换行符，再将Python替换为C。*

**print**(line**.**rstrip()**.**replace('Python', 'C'))

##### 练习10-3访客

编写一个程序，提示用户输入其名字；用户做出响应后，将其名字写入到文件guest.txt中。

name **=** input("What's your name? ")

filename **=** 'guest.txt'

**with** open(filename, 'w') **as** f:

f**.**write(name)

输出：

*guest.txt:*

eric

##### 练习10-4 访客名单

编写一个while循环，提示用户输入其名字。用户输入其名字后，在屏幕上打印一句问候语，并将一条到访记录添加到文件guest\_book.txt中。确保这个文件中的每条记录都独占一行。

filename **=** 'guest\_book.txt'

**print**("Enter 'quit' when you are finished.")

**while** True:

name **=** input("\nWhat's your name? ")

**if** name **==** 'quit':

**break**

**else**:

**with** open(filename, 'a') **as** f:

f**.**write(name **+** "\n")

**print**("Hi " **+** name **+** ", you've been added to the guest book.")

输出：

Enter 'quit' when you are finished.

What's your name? **eric**

Hi eric, you've been added to the guest book.

What's your name? **willie**

Hi willie, you've been added to the guest book.

What's your name? **ever**

Hi ever, you've been added to the guest book.

What's your name? **erin**

Hi erin, you've been added to the guest book.

What's your name? **quit**

*guest\_book.txt:*

eric

willie

ever

erin

##### 10-5 关于编程的调查

编写一个while循环，询问用户为何喜欢编程。每当用户输入一个原因后，都将其添加到一个存储所有原因的文件中。

filename **=** 'programming\_poll.txt'

responses **=** []

**while** True:

response **=** input("\nWhy do you like programming? ")

responses**.**append(response)

continue\_poll **=** input("Would you like to let someone else respond? (y/n) ")

**if** continue\_poll **!=** 'y':

**break**

**with** open(filename, 'a') **as** f:

**for** response **in** responses:

f**.**write(response **+** "\n")

输出：

Why do you like programming? **Programmers can build almost anything they can imagine.**

Would you like to let someone else respond? (y/n) **y**

Why do you like programming? **It's really fun, and really satisfying.**

Would you like to let someone else respond? (y/n) **y**

Why do you like programming? **It just never gets old.**

Would you like to let someone else respond? (y/n) **n**

*programming\_poll.txt:*

Programmers can build almost anything they can imagine.

It's really fun, and really satisfying.

It just never gets old.

##### 练习10-6 加法运算

提示用户提供数值输入时，常出现的一个问题是，用户提供的是文本而不是数字。在这种情况下，当你尝试将输入转换为整数时，将引发TypeError异常。编写一个程序，提示用户输入两个数字，再将它们相加并打印结果。在用户输入的任何一个值不是数字时都捕获TypeError异常，并打印一条友好的错误消息。对你编写的程序进行测试：先输入两个数字，再输入一些文本而不是数字。

**try**:

x **=** input("Give me a number: ")

x **=** int(x)

y **=** input("Give me another number: ")

y **=** int(y)

**except** ValueError:

**print**("Sorry, I really needed a number.")

**else**:

sum **=** x **+** y

**print**("The sum of " **+** str(x) **+** " and " **+** str(y) **+** " is " **+** str(sum) **+** ".")

用户输入两个整数时的输出：

Give me a number: **23**

Give me another number: **47**

The sum of 23 and 47 is 70.

用户输入非数值时的输出：

Give me a number: **23**

Give me another number: **fred**

Sorry, I really needed a number.

##### 练习10-7 加法计算器

将你为完成练习10-6而编写的代码放在一个while循环中，让用户犯错（输入的是文本而不是数）后能够继续输入数。

**print**("Enter 'q' at any time to quit.\n")

**while** True:

**try**:

x **=** input("\nGive me a number: ")

**if** x **==** 'q':

**break**

x **=** int(x)

y **=** input("Give me another number: ")

**if** y **==** 'q':

**break**

y **=** int(y)

**except** ValueError:

**print**("Sorry, I really needed a number.")

**else**:

sum **=** x **+** y

**print**("The sum of " **+** str(x) **+** " and " **+** str(y) **+** " is " **+** str(sum) **+** ".")

输出：

Enter 'q' at any time to quit.

Give me a number: **23**

Give me another number: **47**

The sum of 23 and 47 is 70.

Give me a number: **three**

Sorry, I really needed a number.

Give me a number: **3**

Give me another number: **five**

Sorry, I really needed a number.

Give me a number: **-12**

Give me another number: **20**

The sum of -12 and 20 is 8.

Give me a number: **q**

##### 练习10-8 猫和狗

创建两个文件——cats.txt和dogs.txt，在第一个文件中至少存储三只猫的名字，在第二个文件中至少存储三条狗的名字。编写一个程序，尝试读取这些文件，并将其内容打印到屏幕上。将这些代码放在一个try-except代码块中，以便在文件不存在时捕获FileNotFound错误，并打印一条友好的消息。将文件之一移到另一个地方，并确认except代码块中的代码将正确地执行。

*cats.txt:*

henry

clarence

mildred

*dogs.txt:*

willie

annahootz

summit

*cats\_and\_dogs.py:*

filenames **=** ['cats.txt', 'dogs.txt']

**for** filename **in** filenames:

**print**("\nReading file: " **+** filename)

**try**:

**with** open(filename) **as** f:

contents **=** f**.**read()

**print**(contents)

**except** FileNotFoundError:

**print**(" Sorry, I can't find that file.")

两个文件都存在时的输出：

Reading file: cats.txt

henry

clarence

mildred

Reading file: dogs.txt

willie

annahootz

summit

移走文件cats.txt后的输出：

Reading file: cats.txt

Sorry, I can't find that file.

Reading file: dogs.txt

willie

annahootz

summit

##### 练习10-9 沉默的猫和狗

修改你在练习10-8中编写的except代码块，让程序在文件不存在时一言不发。

filenames **=** ['cats.txt', 'dogs.txt']

**for** filename **in** filenames:

**try**:

**with** open(filename) **as** f:

contents **=** f**.**read()

**except** FileNotFoundError:

**pass**

**else**:

**print**("\nReading file: " **+** filename)

**print**(contents)

两个文件都存在时的输出：

Reading file: cats.txt

henry

clarence

mildred

Reading file: dogs.txt

willie

annahootz

summit

移走文件cats.txt后的输出：

Reading file: dogs.txt

willie

annahootz

summit

##### 练习10-11 喜欢的数

编写一个程序，提示用户输入他喜欢的数，并使用json.dump()将这个数存储到文件中。再编写一个程序，从文件中读取这个值，并打印消息“I know your favorite number! It’s \_\_\_\_\_.”。

*favorite\_number\_write.py:*

import json

number **=** input("What's your favorite number? ")

**with** open('favorite\_number.json', 'w') **as** f:

json**.**dump(number, f)

**print**("Thanks! I'll remember that.")

输出：

What's your favorite number? **42**

Thanks! I'll remember that.

*favorite\_number\_read.py:*

import json

**with** open('favorite\_number.json') **as** f:

number **=** json**.**load(f)

**print**("I know your favorite number! It's " **+** str(number) **+** ".")

输出：

I know your favorite number! It's 42.

##### 练习10-12 记住喜欢的数

将练习10-11中的两个程序合而为一。如果存储了用户喜欢的数，就向用户显示它；否则提示用户输入他喜欢的数并将其存储到文件中。运行这个程序两次，看看它是否像预期的那样工作。

import json

**try**:

**with** open('favorite\_number.json') **as** f:

number **=** json**.**load(f)

**except** FileNotFoundError:

number **=** input("What's your favorite number? ")

**with** open('favorite\_number.json', 'w') **as** f:

json**.**dump(number, f)

**print**("Thanks, I'll remember that.")

**else**:

**print**("I know your favorite number! It's " **+** str(number) **+** ".")

第一次运行时的输出：

What's your favorite number? **42**

Thanks, I'll remember that.

第二次运行的输出：

I know your favorite number! It's 42.

##### 练习10-13 验证用户

最后一个remember\_me.py版本假设用户要么已输入其用户名，要么是首次运行该程序。我们应修改这个程序，以应对这样的情形：当前和最后一次运行该程序的用户并非同一个人。

为此，在greet\_user()中打印欢迎用户回来的消息前，询问他用户名是否是对的。如果不对，就调用get\_new\_username()让用户输入正确的用户名。

import json

**def** **get\_stored\_username**():

"""获取存储的用户名——如果存储了。"""

filename **=** 'username.json'

**try**:

**with** open(filename) **as** f\_obj:

username **=** json**.**load(f\_obj)

**except** FileNotFoundError:

**return** None

**else**:

**return** username

**def** **get\_new\_username**():

"""提示用户输入用户名。"""

username **=** input("What is your name? ")

filename **=** 'username.json'

**with** open(filename, 'w') **as** f\_obj:

json**.**dump(username, f\_obj)

**return** username

**def** **greet\_user**():

"""基于用户名问候用户。"""

username **=** get\_stored\_username()

**if** username:

correct **=** input("Are you " **+** username **+** "? (y/n) ")

**if** correct **==** 'y':

**print**("Welcome back, " **+** username **+** "!")

**else**:

username **=** get\_new\_username()

**print**("We'll remember you when you come back, " **+** username **+** "!")

**else**:

username **=** get\_new\_username()

**print**("We'll remember you when you come back, " **+** username **+** "!")

greet\_user()

输出：

> **python verify\_user.py**

What is your name? **eric**

We'll remember you when you come back, eric!

> **python verify\_user.py**

Are you eric? (y/n) **y**

Welcome back, eric!

> **python verify\_user.py**

Are you eric? (y/n) **n**

What is your name? **ever**

We'll remember you when you come back, ever!

> **python verify\_user.py**

Are you ever? (y/n) **y**

Welcome back, ever!

你可能注意到了，在这个版本的greet\_user()中，有两个相同的else代码块。要整理这个函数，一种方法是使用空的return语句。空的return语句让Python离开当前函数，不执行后面的代码。

下面的greet\_user()版本更清晰的：

**def** **greet\_user**():

"""基于用户名问候用户。"""

username **=** get\_stored\_username()

**if** username:

correct **=** input("Are you " **+** username **+** "? (y/n) ")

**if** correct **==** 'y':

**print**("Welcome back, " **+** username **+** "!")

**return**

*# 获得了用户名，但不对。*

*# 因此提示用户输入其用户名。*

username **=** get\_new\_username()

**print**("We'll remember you when you come back, " **+** username **+** "!")

这条return语句意味着打印欢迎回来的消息后，不再执行后面的代码。如果用户名不存在或不对，就根本不会执行这条return语句。仅当if语句中的条件不满足时，才会执行这个函数的第二部分，因此不需要将它们放在else代码块中。现在，只要有一条if语句中的条件不满足，这个函数就会提示用户输入其用户名。

现在唯一的问题是嵌套了if语句。为解决这个问题，可将检查用户名是否正确的代码移到另一个函数中。如果你觉得这个练习很有意思，可再尝试编写一个名为check\_username()的函数，以免在greet\_user()中嵌套if语句。

### 第11章

##### 练习11-1 城市和国家

编写一个函数，它接受两个形参：一个城市名和一个国家名。这个函数返回一个格式为City, Country的字符串，如Santiago, Chile。将这个函数存储在一个名为city \_functions.py的模块中。

创建一个名为test\_cities.py的程序，对刚编写的函数进行测试（别忘了，你需要导入模块unittest以及要测试的函数）。编写一个名为test\_city\_country()的方法，核实使用类似于'santiago'和'chile'这样的值来调用前述函数时，得到的字符串是正确的。运行test\_cities.py，确认测试test\_city\_country()通过了。

*city\_functions.py:*

"""一系列处理城市的函数。"""

**def** **city\_country**(city, country):

"""返回一个形如'Santiago, Chile'的字符串。"""

**return**(city**.**title() **+** ", " **+** country**.**title())

注意：这个函数是在练习8-6中编写的。

*test\_cities.py:*

import unittest

from city\_functions import city\_country

**class** **CitiesTestCase**(unittest**.**TestCase):

"""测试city\_functions.py。"""

**def** **test\_city\_country**(self):

"""传入简单的城市和国家可行吗？"""

santiago\_chile **=** city\_country('santiago', 'chile')

self**.**assertEqual(santiago\_chile, 'Santiago, Chile')

unittest**.**main()

输出：

.

----------------------------------------------------------------------

Ran 1 test in 0.000s

OK

##### 练习11-2 人口数量

修改前面的函数，使其包含第三个必不可少的形参——population，并返回一个格式为City, Country – population xxx的字符串，如Santiago, Chile – population 5000000。运行test\_cities.py，确认测试test\_city\_country()未通过。

修改上述函数，将形参population设置为可选的。再次运行test\_cities.py，确认测试test\_city\_country()又通过了。

再编写一个名为test\_city\_country\_population()的测试，核实可以使用类似于'santiago'、'chile'和'population=5000000'这样的值来调用这个函数。再次运行test\_cities.py，确认测试test\_city\_country\_population()通过了。

修改后的city\_functions.py，包含必不可少的形参population：

"""一系列处理城市的函数。"""

**def** **city\_country**(city, country, population):

"""返回一个形如'Santiago, Chile - population 5000000'的字符串。"""

output\_string **=** city**.**title() **+** ", " **+** country**.**title()

output\_string **+=** ' - population ' **+** str(population)

**return** output\_string

运行test\_cities.py得到的输出：

E

======================================================================

ERROR: test\_city\_country (\_\_main\_\_.CitiesTestCase)

Does a simple city and country pair work?

----------------------------------------------------------------------

Traceback (most recent call last):

File "pcc\solutions\test\_cities.py", line 10, in test\_city\_country

santiago\_chile = city\_country('santiago', 'chile')

TypeError: city\_country() missing 1 required positional argument: 'population'

----------------------------------------------------------------------

Ran 1 test in 0.000s

FAILED (errors=1)

修改后的city\_functions.py，包含可选形参population：

"""一系列处理城市的函数。"""

**def** **city\_country**(city, country, population**=**0):

"""返回一个表示城市和国家的字符串。"""

output\_string **=** city**.**title() **+** ", " **+** country**.**title()

**if** population:

output\_string **+=** ' - population ' **+** str(population)

**return** output\_string

运行test\_cities.py得到的输出：

.

----------------------------------------------------------------------

Ran 1 test in 0.001s

OK

修改后的test\_cities.py：

import unittest

from city\_functions import city\_country

**class** **CitiesTestCase**(unittest**.**TestCase):

"""测试'city\_functions.py'。"""

**def** **test\_city\_country**(self):

"""传入简单的城市和国家可行吗？"""

santiago\_chile **=** city\_country('santiago', 'chile')

self**.**assertEqual(santiago\_chile, 'Santiago, Chile')

**def** **test\_city\_country\_population**(self):

"""可向形参population传递值吗？"""

santiago\_chile **=** city\_country('santiago', 'chile', population**=**5000000)

self**.**assertEqual(santiago\_chile, 'Santiago, Chile - population 5000000')

unittest**.**main()

输出：

..

----------------------------------------------------------------------

Ran 2 tests in 0.000s

OK

##### 练习11-3 雇员

编写一个名为Employee的类，其方法\_\_init\_\_()接受名、姓和年薪，并将它们都存储在属性中。编写一个名为give\_raise()的方法，它默认将年薪增加5000美元，但也能够接受其他的年薪增加量。

为Employee编写一个测试用例，其中包含两个测试方法：test\_give\_default\_raise()和test\_give\_custom\_raise()。使用方法setUp()，以免在每个测试方法中都创建新的雇员实例。运行这个测试用例，确认两个测试都通过了。

*employee.py:*

**class** **Employee**():

"""一个表示雇员的类。"""

**def** **\_\_init\_\_**(self, f\_name, l\_name, salary):

"""初始化雇员。"""

self**.**first **=** f\_name**.**title()

self**.**last **=** l\_name**.**title()

self**.**salary **=** salary

**def** **give\_raise**(self, amount**=**5000):

"""给雇员加薪。"""

self**.**salary **+=** amount

*test\_employee.py:*

import unittest

from employee import Employee

**class** **TestEmployee**(unittest**.**TestCase):

"""测试模块employee。"""

**def** **setUp**(self):

"""创建一个Employee实例，以便在测试中使用。"""

self**.**eric **=** Employee('eric', 'matthes', 65000)

**def** **test\_give\_default\_raise**(self):

"""测试使用默认的年薪增加量是否可行。"""

self**.**eric**.**give\_raise()

self**.**assertEqual(self**.**eric**.**salary, 70000)

**def** **test\_give\_custom\_raise**(self):

"""测试自定义年薪增加量是否可行。"""

self**.**eric**.**give\_raise(10000)

self**.**assertEqual(self**.**eric**.**salary, 75000)

unittest**.**main()

输出：

..

----------------------------------------------------------------------

Ran 2 tests in 0.000s

OK

### 第15章

##### 练习15-1 立方

数字的三次方被称为其立方。请绘制一个图形，显示前5个整数的立方值；再绘制一个图形，显示前5000个整数的立方值。

from matplotlib import pyplot **as** plt

*# 定义数据。*

x\_values **=** [1, 2, 3, 4, 5]

cubes **=** [1, 8, 27, 64, 125]

*# 绘制图形。*

plt**.**scatter(x\_values, cubes, edgecolor**=**'none', s**=**40)

*# 定制图形。*

plt**.**title("Cubes", fontsize**=**24)

plt**.**xlabel('Value', fontsize**=**14)

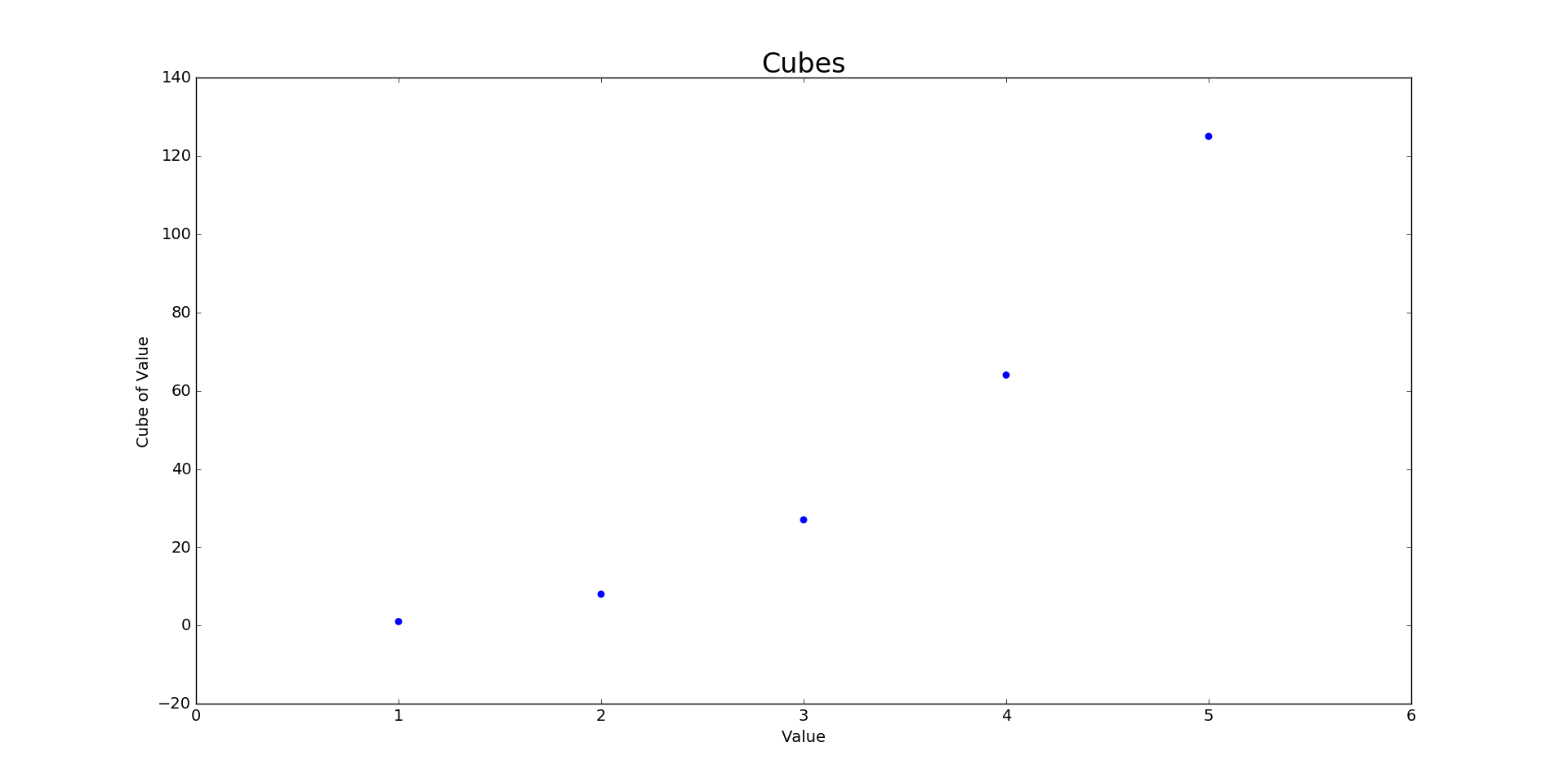
plt**.**ylabel('Cube of Value', fontsize**=**14)

plt**.**tick\_params(axis**=**'both', labelsize**=**14)

*# 显示图形。*

plt**.**show()

输出：



绘制一个图形，显示前5000个整数的立方值：

from matplotlib import pyplot **as** plt

*# 定义数据。*

x\_values **=** list(range(5001))

cubes **=** [x**\*\***3 **for** x **in** x\_values]

*# 绘制图形。*

plt**.**scatter(x\_values, cubes, edgecolor**=**'none', s**=**40)

*# 定制图形。*

plt**.**title("Cubes", fontsize**=**24)

plt**.**xlabel('Value', fontsize**=**14)

plt**.**ylabel('Cube of Value', fontsize**=**14)

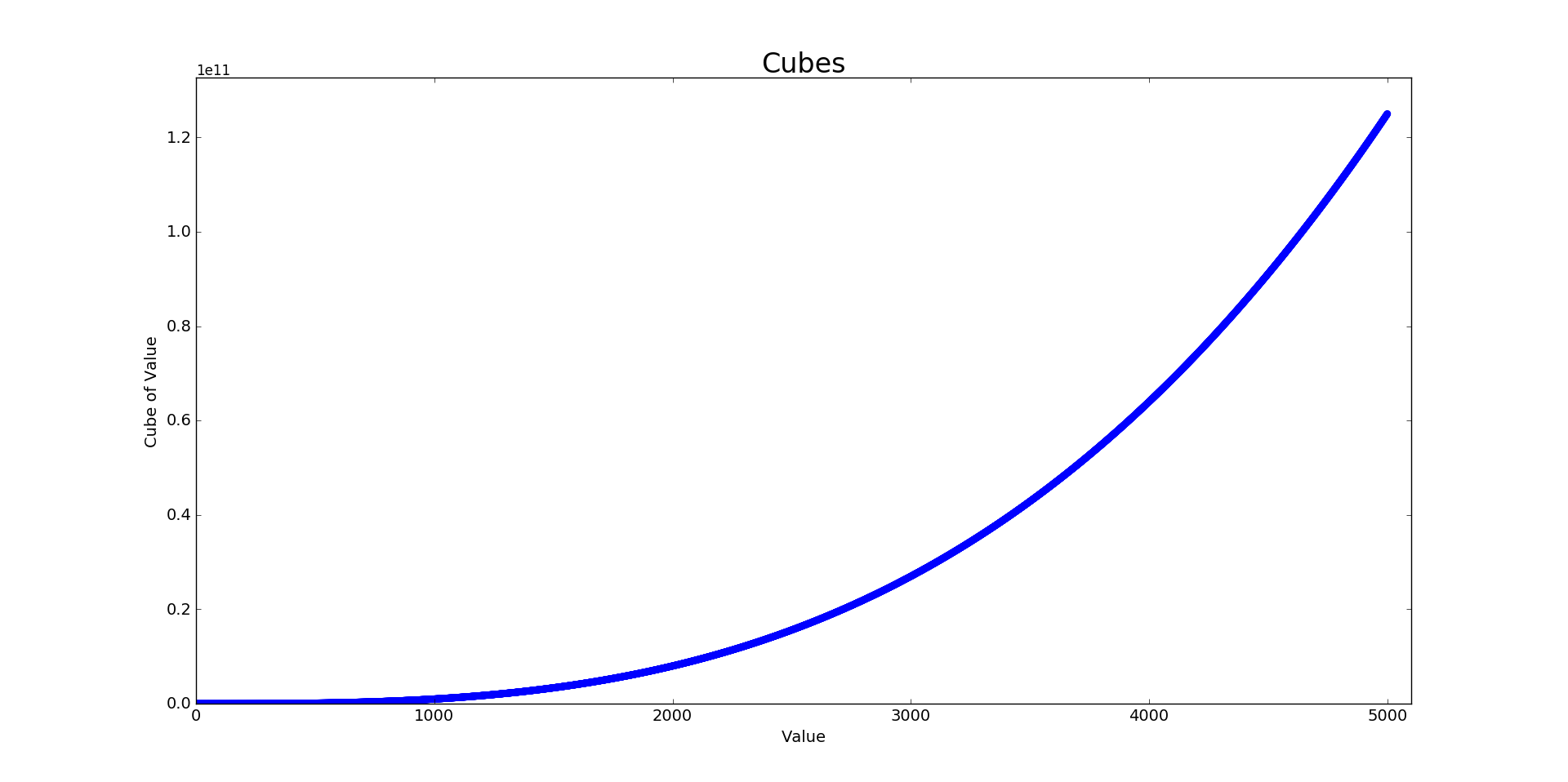
plt**.**tick\_params(axis**=**'both', labelsize**=**14)

plt**.**axis([0, 5100, 0, 5100**\*\***3])

*# 显示图形。*

plt**.**show()

输出：



##### 练习15-2 彩色立方

给前面绘制的立方图指定颜色映射。

from matplotlib import pyplot **as** plt

*# 定义数据。*

x\_values **=** list(range(5001))

cubes **=** [x**\*\***3 **for** x **in** x\_values]

*# 绘制图形。*

plt**.**scatter(x\_values, cubes, edgecolor**=**'none', c**=**cubes, cmap**=**plt**.**cm**.**BuGn, s**=**40)

*# 定制图形。*

plt**.**title("Cubes", fontsize**=**24)

plt**.**xlabel('Value', fontsize**=**14)

plt**.**ylabel('Cube of Value', fontsize**=**14)

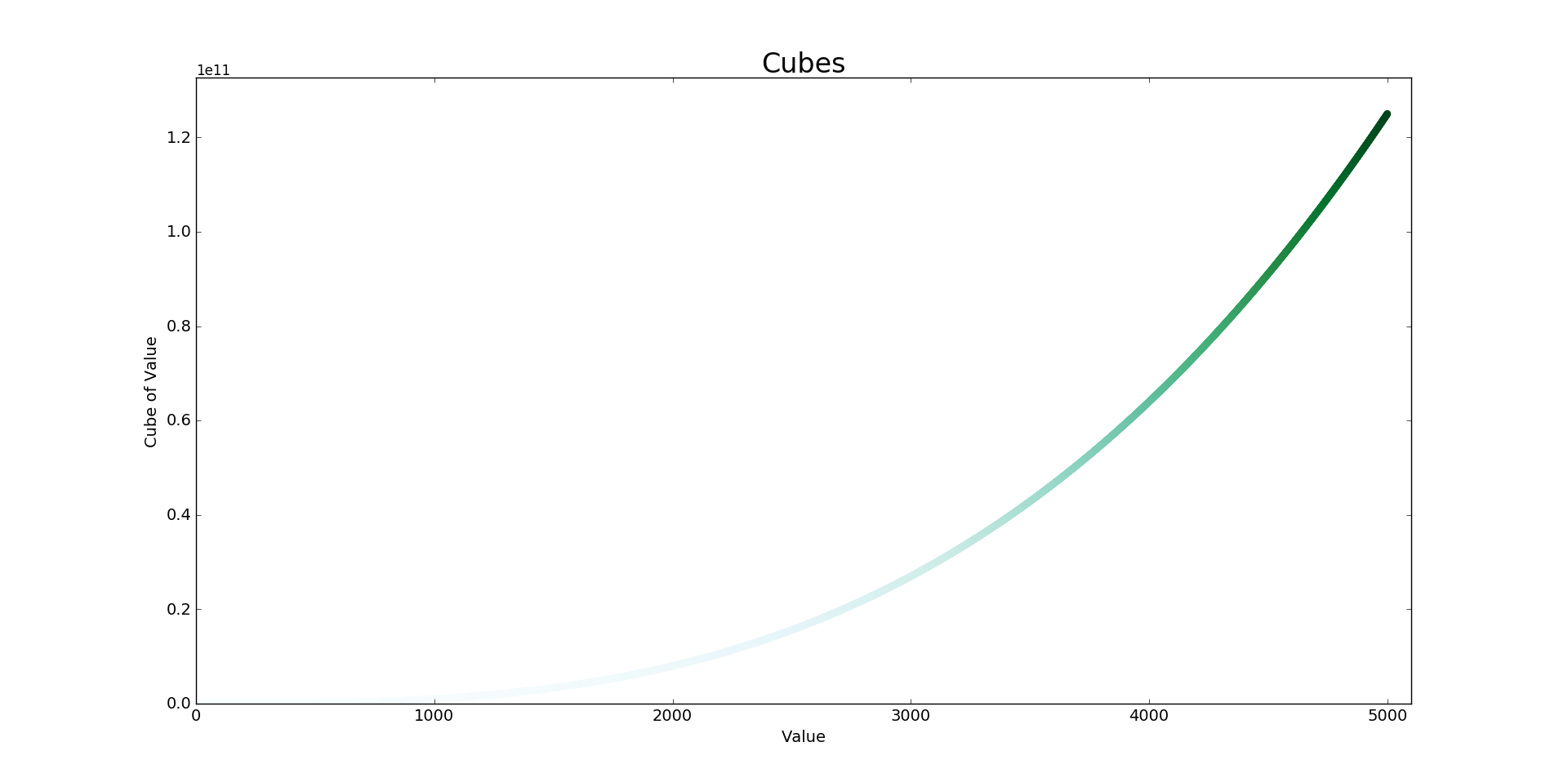
plt**.**tick\_params(axis**=**'both', labelsize**=**14)

plt**.**axis([0, 5100, 0, 5100**\*\***3])

*# 显示图形。*

plt**.**show()

输出：



##### 练习15-3 分子运动

修改rw\_visual.py，将其中的plt.scatter()替换为plt.plot()。为模拟花粉在水滴表面的运动路径，向plt.plot()传递rw.x\_values和rw.y\_values，并指定实参值linewidth。使用5000个点而不是50 000个点。

import matplotlib.pyplot **as** plt

from random\_walk import RandomWalk

*# 只要程序处于活动状态，就不断地模拟随机漫步。*

**while** True:

*# 创建一个RandomWalk实例，并将其包含的点都绘制出来。*

rw **=** RandomWalk(5000)

rw**.**fill\_walk()

*# 设置绘图窗口的尺寸。*

plt**.**figure(dpi**=**128, figsize**=**(10, 6))

point\_numbers **=** list(range(rw**.**num\_points))

plt**.**plot(rw**.**x\_values, rw**.**y\_values, linewidth**=**1)

*# 突出起点和终点。*

plt**.**scatter(0, 0, c**=**'green', edgecolors**=**'none', s**=**75)

plt**.**scatter(rw**.**x\_values[**-**1], rw**.**y\_values[**-**1], c**=**'red', edgecolors**=**'none',

s**=**75)

*# 隐藏坐标轴。*

plt**.**axes()**.**get\_xaxis()**.**set\_visible(False)

plt**.**axes()**.**get\_yaxis()**.**set\_visible(False)

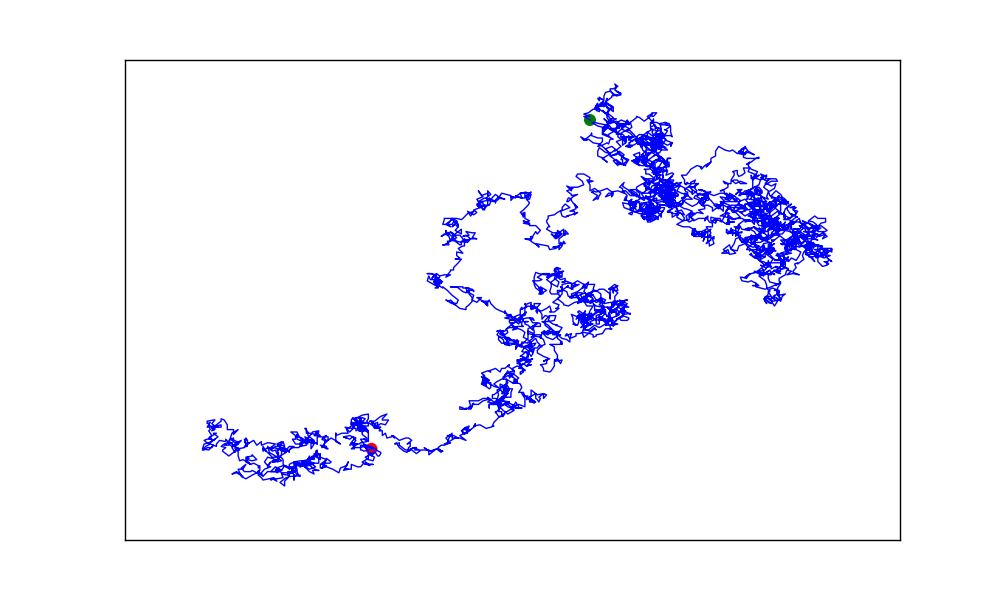
plt**.**show()

keep\_running **=** input("Make another walk? (y/n): ")

**if** keep\_running **==** 'n':

**break**

输出：



起点和终点看起来位于折线的后面。要将它们放在折线的前面，可使用参数zorder。zorder值大的图表元素位于zorder值小的图表元素前面。

import matplotlib.pyplot **as** plt

from random\_walk import RandomWalk

*# 只要程序处于活动状态，就不断地模拟随机漫步。*

**while** True:

*# 创建一个RandomWalk实例，并将其包含的点都绘制出来。*

rw **=** RandomWalk(5000)

rw**.**fill\_walk()

*# 设置绘图窗口的尺寸。*

plt**.**figure(dpi**=**128, figsize**=**(10, 6))

point\_numbers **=** list(range(rw**.**num\_points))

plt**.**plot(rw**.**x\_values, rw**.**y\_values, linewidth**=**1, zorder**=**1)

*# 突出起点和终点。*

plt**.**scatter(0, 0, c**=**'green', edgecolors**=**'none', s**=**75, zorder**=**2)

plt**.**scatter(rw**.**x\_values[**-**1], rw**.**y\_values[**-**1], c**=**'red', edgecolors**=**'none',

s**=**75, zorder**=**2)

*# 隐藏坐标轴。*

plt**.**axes()**.**get\_xaxis()**.**set\_visible(False)

plt**.**axes()**.**get\_yaxis()**.**set\_visible(False)

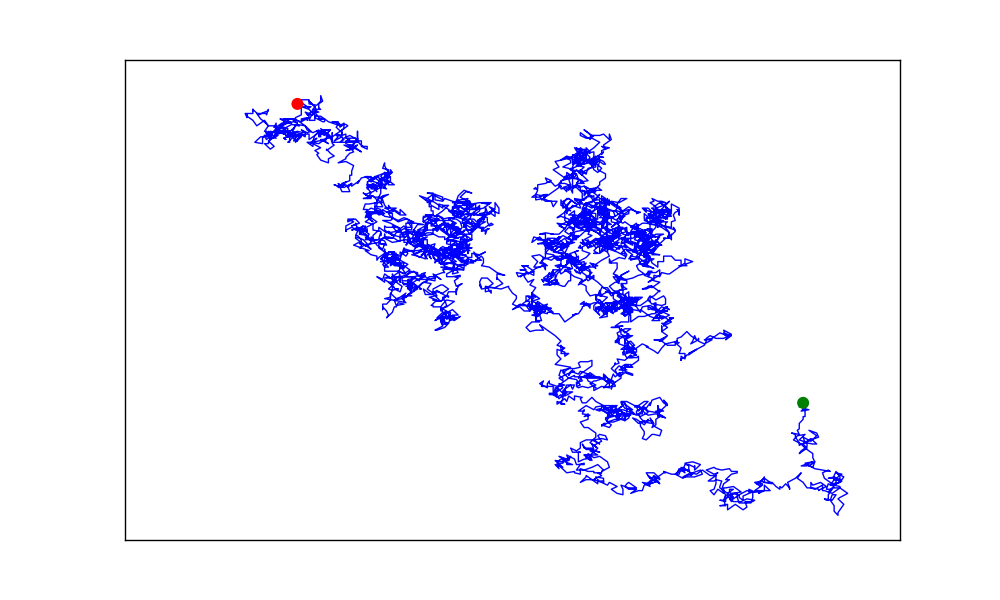
plt**.**show()

keep\_running **=** input("Make another walk? (y/n): ")

**if** keep\_running **==** 'n':

**break**

输出：



##### 练习15-5 重构

方法fill\_walk()很长。请新建一个名为get\_step()的方法，用于确定每次漫步的距离和方向，并计算这次漫步将如何移动。然后，在fill\_walk()中调用get\_step()两次：

x\_step **=** get\_step()

y\_step **=** get\_step()

通过这样的重构，可缩小fill\_walk()的规模，让这个方法阅读和理解起来更容易。

random\_walk.py:

from random import choice

**class** **RandomWalk**():

"""一个生成随机漫步数据的类。"""

**def** **\_\_init\_\_**(self, num\_points**=**5000):

"""初始化随机漫步的属性。"""

self**.**num\_points **=** num\_points

*# 所有随机漫步都始于(0, 0)。*

self**.**x\_values **=** [0]

self**.**y\_values **=** [0]

**def** **get\_step**(self):

"""决定漫步的方向和距离。"""

direction **=** choice([1, **-**1])

distance **=** choice([0, 1, 2, 3, 4])

step **=** direction **\*** distance

**return** step

**def** **fill\_walk**(self):

"""计算随机漫步包含的所有点。"""

*# 不断漫步，直到列表达到指定的长度。*

**while** len(self**.**x\_values) **<** self**.**num\_points:

*# 决定前进方向以及沿这个方向前进的距离。*

x\_step **=** self**.**get\_step()

y\_step **=** self**.**get\_step()

*# 拒绝原地踏步。*

**if** x\_step **==** 0 **and** y\_step **==** 0:

**continue**

*# 计算下一个点的x和y值。*

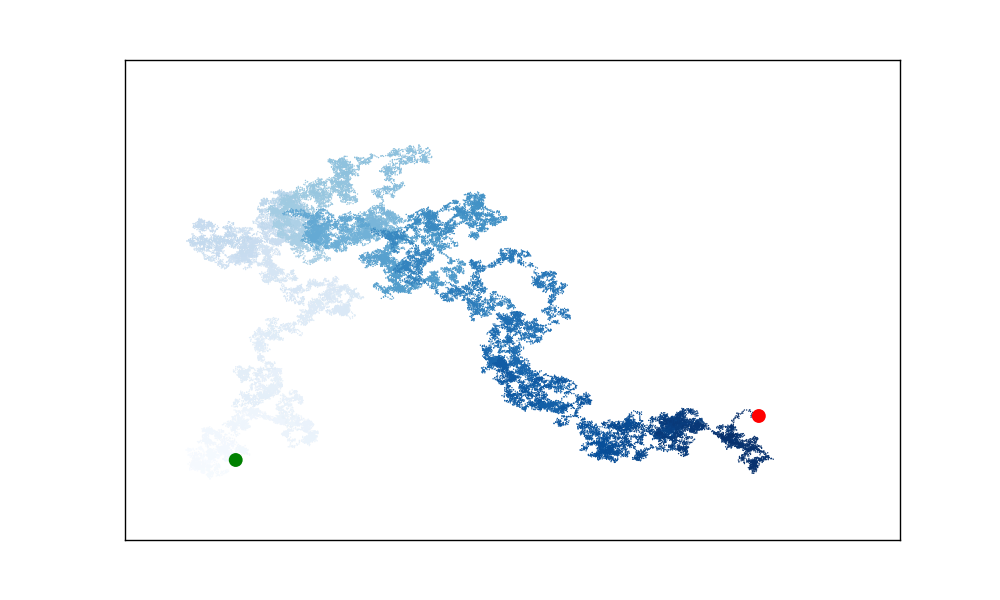
next\_x **=** self**.**x\_values[**-**1] **+** x\_step

next\_y **=** self**.**y\_values[**-**1] **+** y\_step

self**.**x\_values**.**append(next\_x)

self**.**y\_values**.**append(next\_y)

输出：



##### 练习15-6 自动生成标签

请修改die\_visual.py和dice\_visual.py，将用来设置hist.x\_labels值的列表替换为一个自动生成这种列表的循环。如果你熟悉列表解析，可尝试将die\_visual.py和dice\_visual.py中的其他for循环也替换为列表解析。

die\_visual.py:

import pygal

from die import Die

*# 创建一个D6骰子。*

die **=** Die()

*# 掷骰子多次，并将结果存储在一个列表中。*

results **=** [die**.**roll() **for** roll\_num **in** range(1000)]

*# 分析结果。*

frequencies **=** [results**.**count(value) **for** value **in** range(1, die**.**num\_sides**+**1)]

*#可视化结果。*

hist **=** pygal**.**Bar()

hist**.**title **=** "Results of rolling one D6 1000 times."

hist**.**x\_labels **=** [str(x) **for** x **in** range(1, die**.**num\_sides**+**1)]

hist**.**x\_title **=** "Result"

hist**.**y\_title **=** "Frequency of Result"

hist**.**add('D6', frequencies)

hist**.**render\_to\_file('die\_visual.svg')

dice\_visual.py:

import pygal

from die import Die

*#创建两个D6骰子。*

die\_1 **=** Die()

die\_2 **=** Die()

*#掷骰子多次，并将结果存储在一个列表中。*

results **=** [die\_1**.**roll() **+** die\_2**.**roll() **for** roll\_num **in** range(1000)]

*#分析结果。*

max\_result **=** die\_1**.**num\_sides **+** die\_2**.**num\_sides

frequencies **=** [results**.**count(value) **for** value **in** range(2, max\_result**+**1)]

*#可视化结果。*

hist **=** pygal**.**Bar()

hist**.**title **=** "Results of rolling two D6 dice 1000 times."

hist**.**x\_labels **=** [str(x) **for** x **in** range(2, max\_result**+**1)]

hist**.**x\_title **=** "Result"

hist**.**y\_title **=** "Frequency of Result"

hist**.**add('D6 + D6', frequencies)

hist**.**render\_to\_file('dice\_visual.svg')

##### 练习15-7 两个D8骰子

请模拟同时掷两个8面骰子1000次的结果。逐渐增加掷骰子的次数，直到系统不堪重负为止。

import pygal

from die import Die

*#创建两个D8骰子。*

die\_1 **=** Die(8)

die\_2 **=** Die(8)

*#掷骰子多次，并将结果存储在一个列表中。*

results **=** []

**for** roll\_num **in** range(1000000):

result **=** die\_1**.**roll() **+** die\_2**.**roll()

results**.**append(result)

*#分析结果。*

frequencies **=** []

max\_result **=** die\_1**.**num\_sides **+** die\_2**.**num\_sides

**for** value **in** range(2, max\_result**+**1):

frequency **=** results**.**count(value)

frequencies**.**append(frequency)

*#可视化结果。*

hist **=** pygal**.**Bar()

hist**.**title **=** "Results of rolling two D8 dice 1,000,000 times."

hist**.**x\_labels **=** [str(x) **for** x **in** range(2, max\_result**+**1)]

hist**.**x\_title **=** "Result"

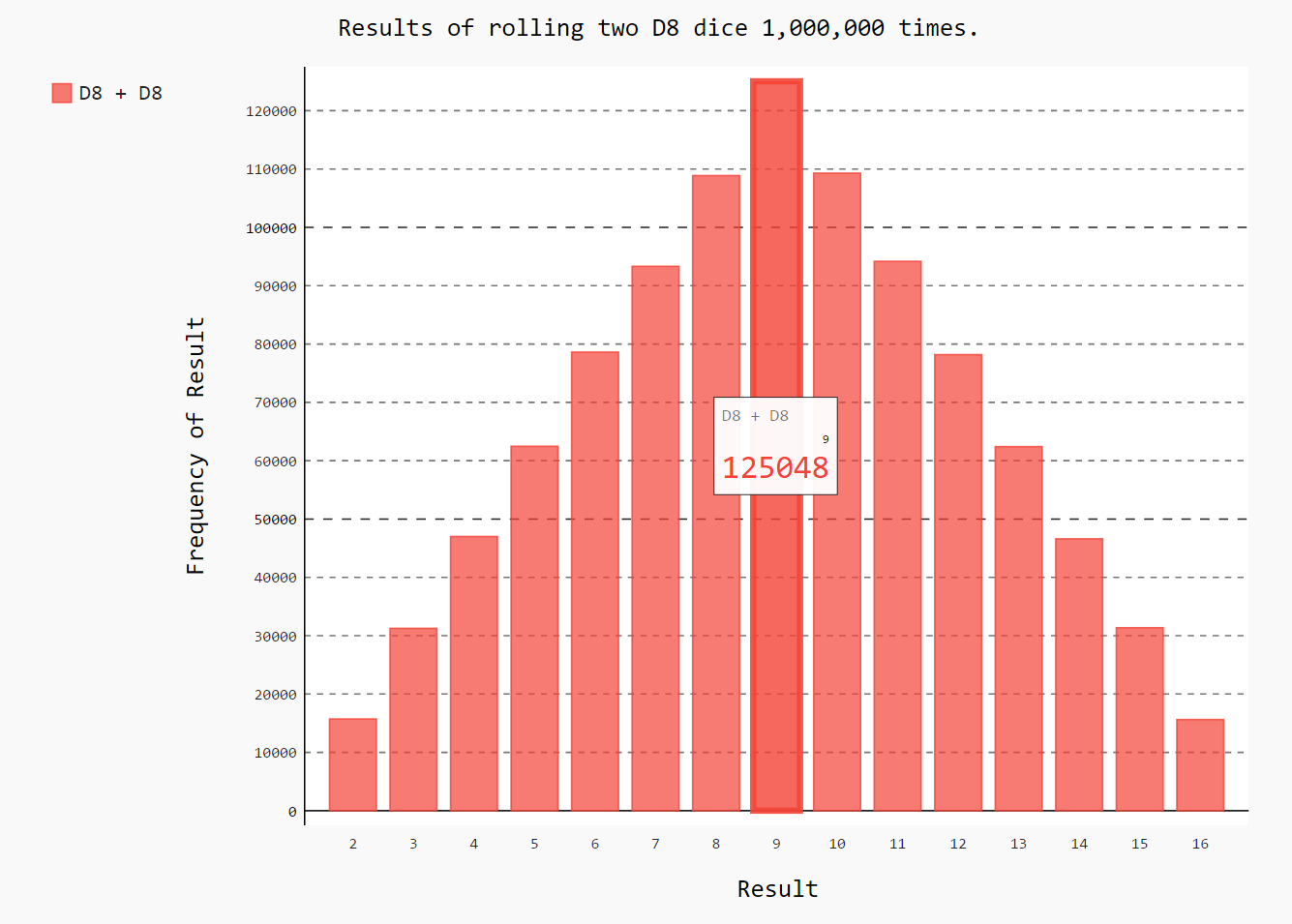
hist**.**y\_title **=** "Frequency of Result"

hist**.**add('D8 + D8', frequencies)

hist**.**render\_to\_file('dice\_visual.svg')

注意：在这个解决方案中，只使用了列表解析来设置参数hist.x\_labels。你可能想尝试将其他循环也替换为列表解析。

输出：



##### 练习15-8 同时掷三个骰子

如果你同时掷三个D6骰子，可能得到的最小点数为3，而最大点数为18。请通过可视化展示同时掷三个D6骰子的结果。

import pygal

from die import Die

*# 创建三个D6骰子。*

die\_1 **=** Die()

die\_2 **=** Die()

die\_3 **=** Die()

*# 掷骰子多次，并将结果存储在一个列表中。*

results **=** []

**for** roll\_num **in** range(1000000):

result **=** die\_1**.**roll() **+** die\_2**.**roll() **+** die\_3**.**roll()

results**.**append(result)

*# 分析结果。*

frequencies **=** []

max\_result **=** die\_1**.**num\_sides **+** die\_2**.**num\_sides **+** die\_3**.**num\_sides

**for** value **in** range(3, max\_result**+**1):

frequency **=** results**.**count(value)

frequencies**.**append(frequency)

*# 可视化结果。*

hist **=** pygal**.**Bar()

hist**.**title **=** "Results of rolling three D6 dice 1,000,000 times."

hist**.**x\_labels **=** [str(x) **for** x **in** range(3, max\_result**+**1)]

hist**.**x\_title **=** "Result"

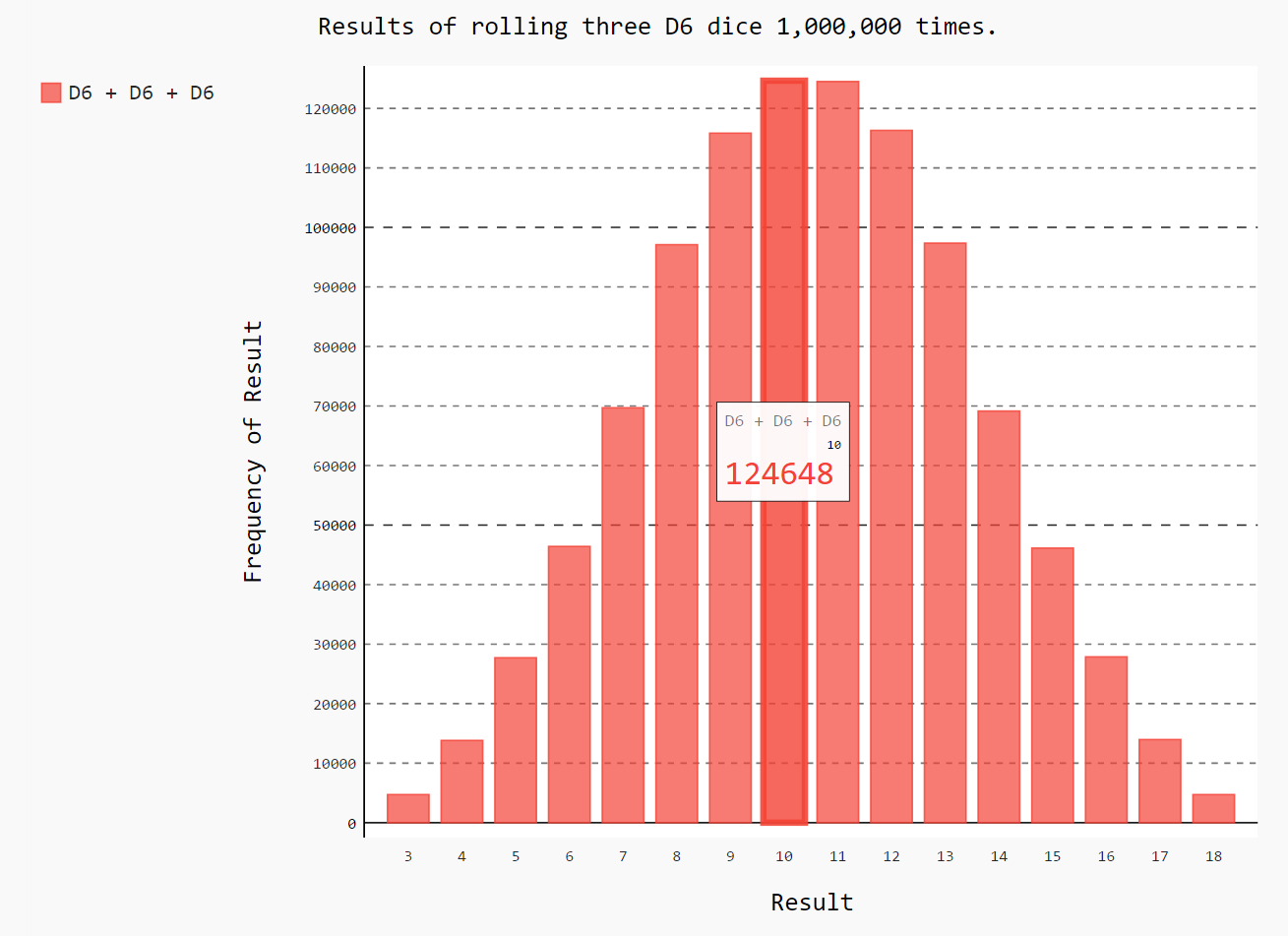
hist**.**y\_title **=** "Frequency of Result"

hist**.**add('D6 + D6 + D6', frequencies)

hist**.**render\_to\_file('dice\_visual.svg')

注意：在这个解决方案中，只使用了列表解析来设置参数hist.x\_labels。你可能想尝试将其他循环也替换为列表解析。

输出：



##### 练习15-9 将点数相乘

同时掷两个骰子时，通常将它们的点数相加。请通过可视化展示将两个骰子的点数相乘的结果。

import pygal

from die import Die

*#创建两个D6骰子。*

die\_1 **=** Die()

die\_2 **=** Die()

*#掷骰子多次，并将结果存储在一个列表中。*

results **=** []

**for** roll\_num **in** range(1000000):

result **=** die\_1**.**roll() **\*** die\_2**.**roll()

results**.**append(result)

*# 分析结果。*

frequencies **=** []

max\_result **=** die\_1**.**num\_sides **\*** die\_2**.**num\_sides

**for** value **in** range(1, max\_result**+**1):

frequency **=** results**.**count(value)

frequencies**.**append(frequency)

*# 可视化结果。*

hist **=** pygal**.**Bar()

hist**.**title **=** "Results of multiplying two D6 dice. (1,000,000 rolls)"

hist**.**x\_labels **=** [str(x) **for** x **in** range(1, max\_result**+**1)]

hist**.**x\_title **=** "Result"

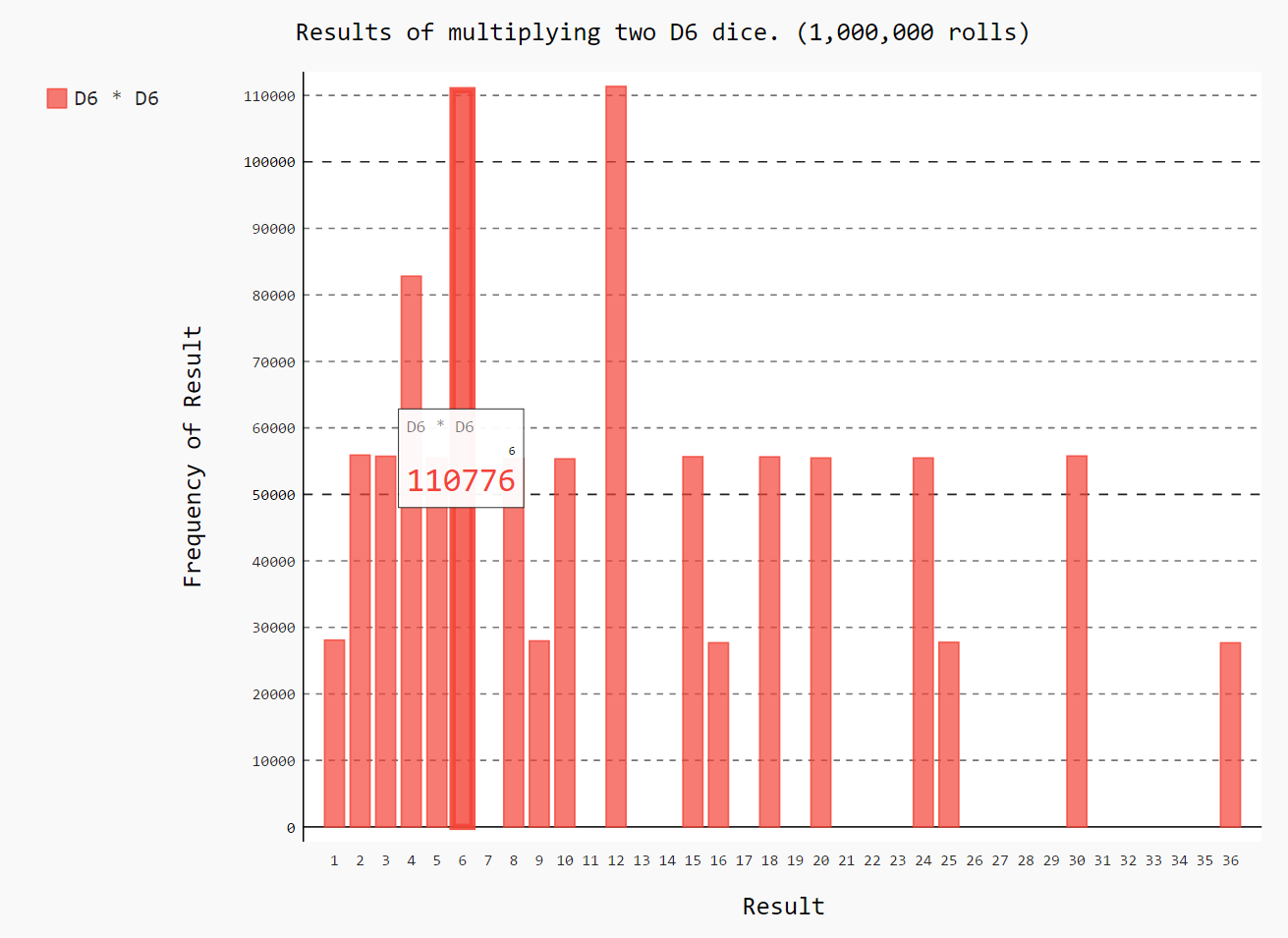
hist**.**y\_title **=** "Frequency of Result"

hist**.**add('D6 \* D6', frequencies)

hist**.**render\_to\_file('dice\_visual.svg')

注意：在这个解决方案中，只使用了列表解析来设置参数hist.x\_labels。你可能想尝试将其他循环也替换为列表解析。

输出：



### 第16章

##### 练习16-2 比较锡特卡和死亡谷的气温

在有关锡特卡和死亡谷的图表中，气温刻度反映了数据范围的不同。为准确地比较锡特卡和死亡谷的气温范围，需要在y轴上使用相同的刻度。为此，请修改图16.5和16.6所示图表的y轴设置，对锡特卡和死亡谷的气温范围进行直接比较（你也可以对任何两个地方的气温范围进行比较）。你还可以尝试在一个图表中呈现这两个数据集。

要设置y轴的取值范围，可使用pyplot函数ylim()；要设置x轴的取值范围，可使用函数xlim()。

import csv

from datetime import datetime

from matplotlib import pyplot **as** plt

*# 从文件中获取日期、最高气温和最低气温。*

filename **=** 'sitka\_weather\_2014.csv'

**with** open(filename) **as** f:

reader **=** csv**.**reader(f)

header\_row **=** next(reader)

dates, highs, lows **=** [], [], []

**for** row **in** reader:

**try**:

current\_date **=** datetime**.**strptime(row[0], "%Y-%m-%d")

high **=** int(row[1])

low **=** int(row[3])

**except** ValueError:

**print**(current\_date, 'missing data')

**else**:

dates**.**append(current\_date)

highs**.**append(high)

lows**.**append(low)

*# 根据数据绘制图形。*

fig **=** plt**.**figure(dpi**=**128, figsize**=**(10, 6))

plt**.**plot(dates, highs, c**=**'red', alpha**=**0.5)

plt**.**plot(dates, lows, c**=**'blue', alpha**=**0.5)

plt**.**fill\_between(dates, highs, lows, facecolor**=**'blue', alpha**=**0.1)

*# 设置图形的格式。*

title **=** "Daily high and low temperatures - 2014\nSitka, AK"

plt**.**title(title, fontsize**=**20)

plt**.**xlabel('', fontsize**=**16)

fig**.**autofmt\_xdate()

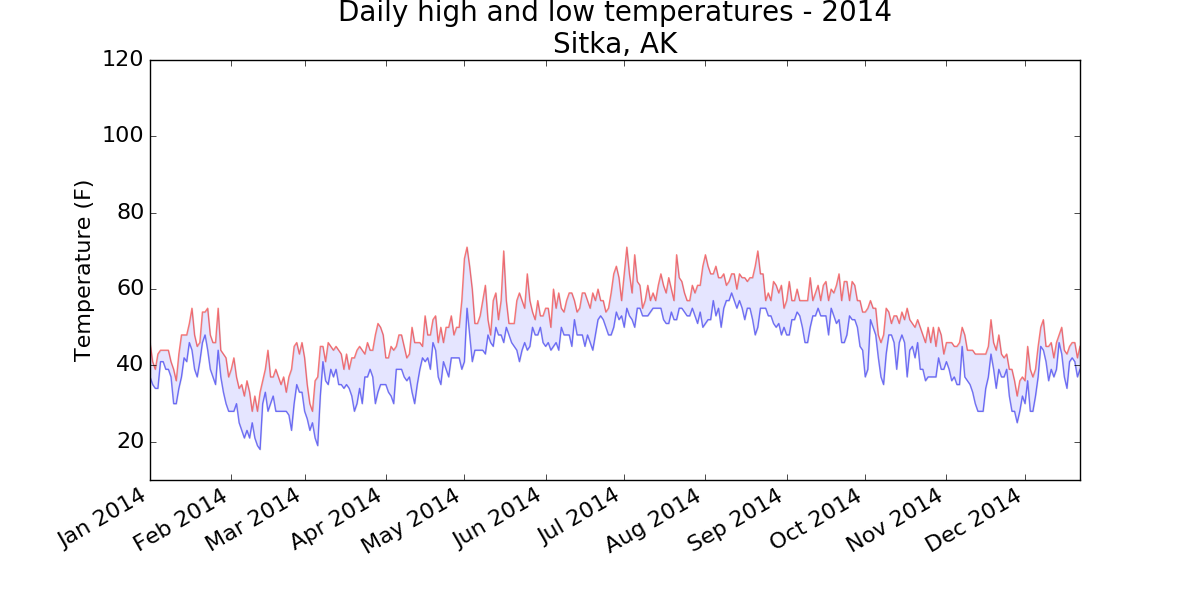
plt**.**ylabel("Temperature (F)", fontsize**=**16)

plt**.**tick\_params(axis**=**'both', which**=**'major', labelsize**=**16)

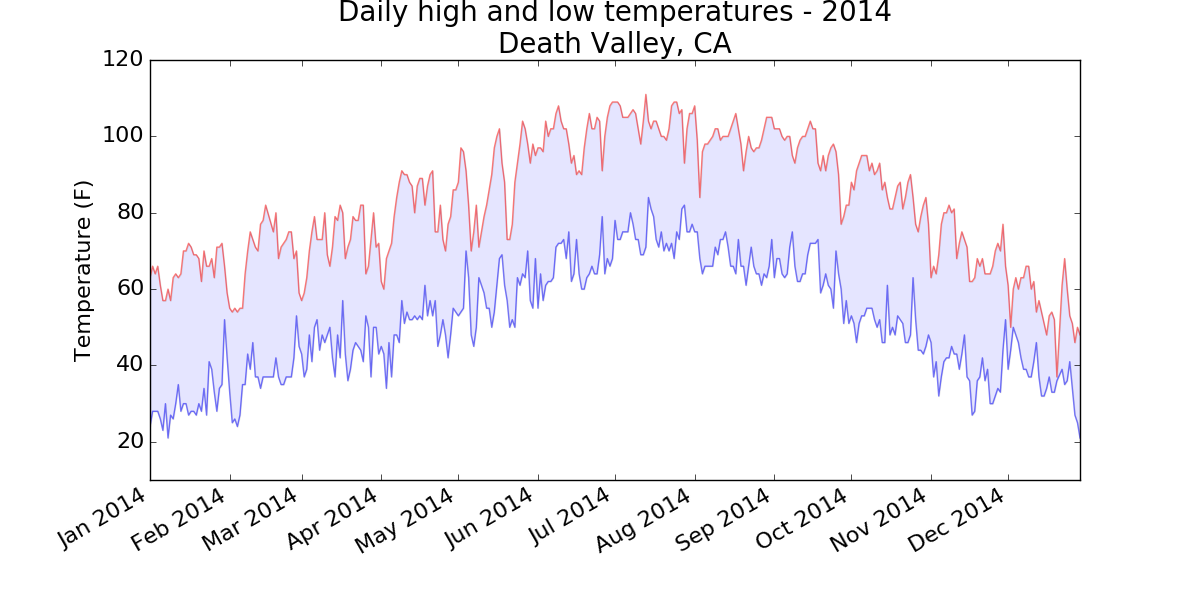
plt**.**ylim(10, 120)

plt**.**show()

输出：



根据死亡谷的气温数据绘制图形时，通过使用函数ylim()指定同样的取值范围，可让两个图形的y轴刻度相同：



在同一个图表中呈现两个数据集的方法很多。在下面的解决方案中，我将读取csv文件的代码放在一个函数中。然后调用这个函数来获取锡特卡的最高气温和最低气温，并绘制图表；接下来，我再次调用这个函数获取死亡谷的数据，并将它们添加到既有的图表中。我稍微调整了颜色，以便将这两个地方的数据区分开来。

import csv

from datetime import datetime

from matplotlib import pyplot **as** plt

**def** **get\_weather\_data**(filename, dates, highs, lows):

"""从数据文件中获取最高气温和最低气温。"""

**with** open(filename) **as** f:

reader **=** csv**.**reader(f)

header\_row **=** next(reader)

*# dates, highs, lows = [], [], []*

**for** row **in** reader:

**try**:

current\_date **=** datetime**.**strptime(row[0], "%Y-%m-%d")

high **=** int(row[1])

low **=** int(row[3])

**except** ValueError:

**print**(current\_date, 'missing data')

**else**:

dates**.**append(current\_date)

highs**.**append(high)

lows**.**append(low)

*# 获取锡特卡的气温数据。*

dates, highs, lows **=** [], [], []

get\_weather\_data('sitka\_weather\_2014.csv', dates, highs, lows)

*# 根据锡特卡的数据绘制图表。*

fig **=** plt**.**figure(dpi**=**128, figsize**=**(10, 6))

plt**.**plot(dates, highs, c**=**'red', alpha**=**0.6)

plt**.**plot(dates, lows, c**=**'blue', alpha**=**0.6)

plt**.**fill\_between(dates, highs, lows, facecolor**=**'blue', alpha**=**0.15)

*# 获取死亡谷的数据。*

dates, highs, lows **=** [], [], []

get\_weather\_data('death\_valley\_2014.csv', dates, highs, lows)

*# 将死亡谷的数据添加到当前图表中。*

plt**.**plot(dates, highs, c**=**'red', alpha**=**0.3)

plt**.**plot(dates, lows, c**=**'blue', alpha**=**0.3)

plt**.**fill\_between(dates, highs, lows, facecolor**=**'blue', alpha**=**0.05)

*# 设置图表的格式。*

title **=** "Daily high and low temperatures - 2014"

title **+=** "\nSitka, AK and Death Valley, CA"

plt**.**title(title, fontsize**=**20)

plt**.**xlabel('', fontsize**=**16)

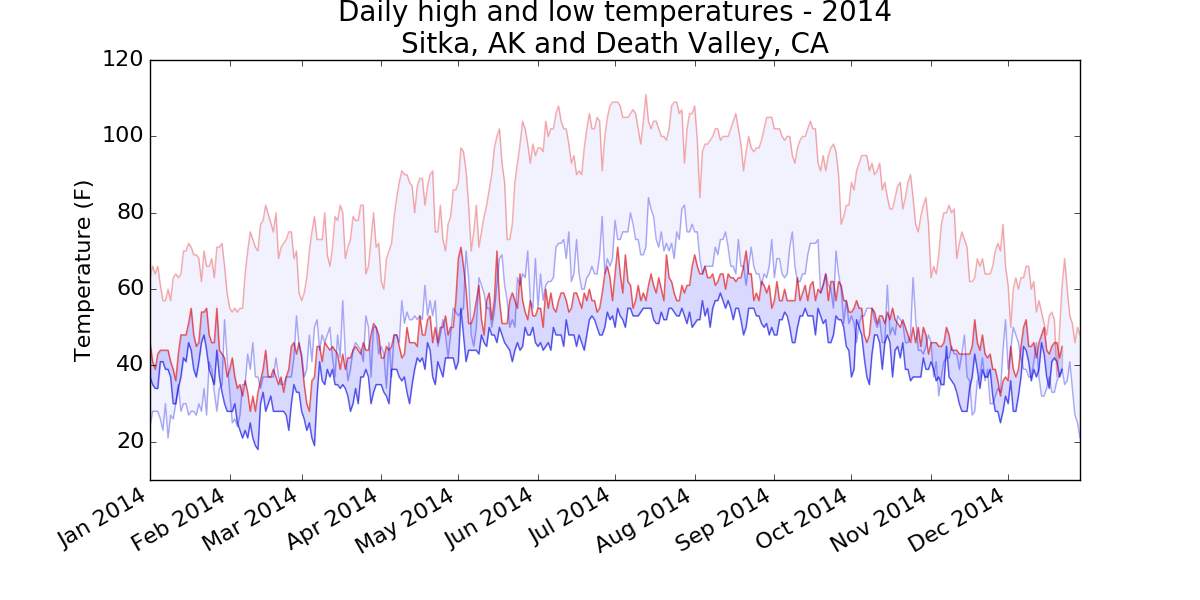
fig**.**autofmt\_xdate()

plt**.**ylabel("Temperature (F)", fontsize**=**16)

plt**.**tick\_params(axis**=**'both', which**=**'major', labelsize**=**16)

plt**.**ylim(10, 120)

plt**.**show()



##### 练习16-3 降雨量

选择你感兴趣的任何地方，通过可视化将其降雨量呈现出来。为此，可先只涵盖一个月的数据，确定代码正确无误后，再使用一整年的数据。

注意：这个示例使用的数据可在这里找到。

import csv

from datetime import datetime

from matplotlib import pyplot **as** plt

*# 从数据文件中获取日期和降雨量数据。*

*# 降雨量数据在第19列。*

filename **=** 'sitka\_rainfall\_2015.csv'

**with** open(filename) **as** f:

reader **=** csv**.**reader(f)

header\_row **=** next(reader)

dates, rainfalls **=** [], []

**for** row **in** reader:

**try**:

current\_date **=** datetime**.**strptime(row[0], "%Y-%m-%d")

rainfall **=** float(row[19])

**except** ValueError:

**print**(current\_date, 'missing data')

**else**:

dates**.**append(current\_date)

rainfalls**.**append(rainfall)

*# 根据数据绘制图表。*

fig **=** plt**.**figure(dpi**=**128, figsize**=**(10, 6))

plt**.**plot(dates, rainfalls, c**=**'blue', alpha**=**0.5)

plt**.**fill\_between(dates, rainfalls, facecolor**=**'blue', alpha**=**0.2)

*# 设置图表的格式。*

title **=** "Daily rainfall amounts - 2015\nSitka, AK"

plt**.**title(title, fontsize**=**20)

plt**.**xlabel('', fontsize**=**16)

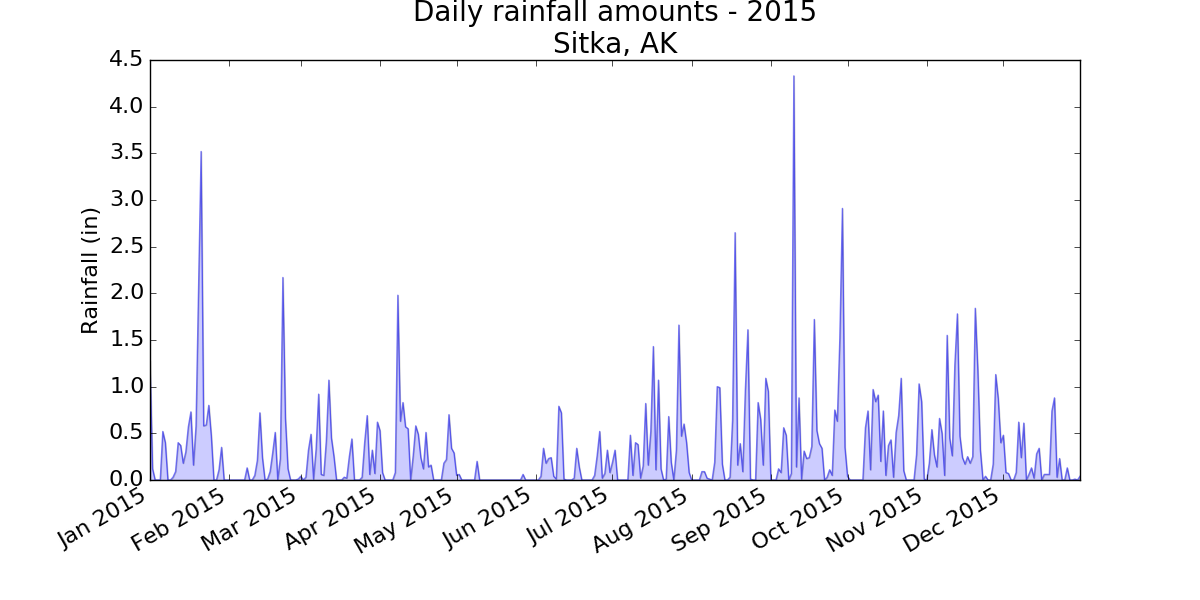
fig**.**autofmt\_xdate()

plt**.**ylabel("Rainfall (in)", fontsize**=**16)

plt**.**tick\_params(axis**=**'both', which**=**'major', labelsize**=**16)

plt**.**show()

输出：



##### 练习16-4 探索

生成一些图表，对你好奇的任何地方的其他天气数据进行研究。

我生活在热带雨林，因此对研究降雨量数据感兴趣。我计算一年期间的累积降雨量，并根据累积降雨量和每天降雨量绘制出相应的图形。即便常年生活在这样的地方，看到降雨量如此之大我还是有些吃惊。

import csv

from datetime import datetime

from matplotlib import pyplot **as** plt

*# 从数据文件中获取日期和降雨量数据。*

*# 降雨量数据在第19列。*

filename **=** 'sitka\_rainfall\_2015.csv'

**with** open(filename) **as** f:

reader **=** csv**.**reader(f)

header\_row **=** next(reader)

dates, rainfalls, totals **=** [], [], []

**for** row **in** reader:

**try**:

current\_date **=** datetime**.**strptime(row[0], "%Y-%m-%d")

rainfall **=** float(row[19])

**except** ValueError:

**print**(current\_date, 'missing data')

**else**:

dates**.**append(current\_date)

rainfalls**.**append(rainfall)

**if** totals:

totals**.**append(totals[**-**1] **+** rainfall)

**else**:

totals**.**append(rainfall)

*# 根据数据绘制图表。*

fig **=** plt**.**figure(dpi**=**128, figsize**=**(10, 6))

plt**.**plot(dates, rainfalls, c**=**'blue', alpha**=**0.5)

plt**.**fill\_between(dates, rainfalls, facecolor**=**'blue', alpha**=**0.2)

plt**.**plot(dates, totals, c**=**'blue', alpha**=**0.75)

plt**.**fill\_between(dates, totals, facecolor**=**'blue', alpha**=**0.05)

*# 设置图表的格式。*

title **=** "Daily rainfall amounts and cumulative rainfall - 2015\nSitka, AK"

plt**.**title(title, fontsize**=**20)

plt**.**xlabel('', fontsize**=**16)

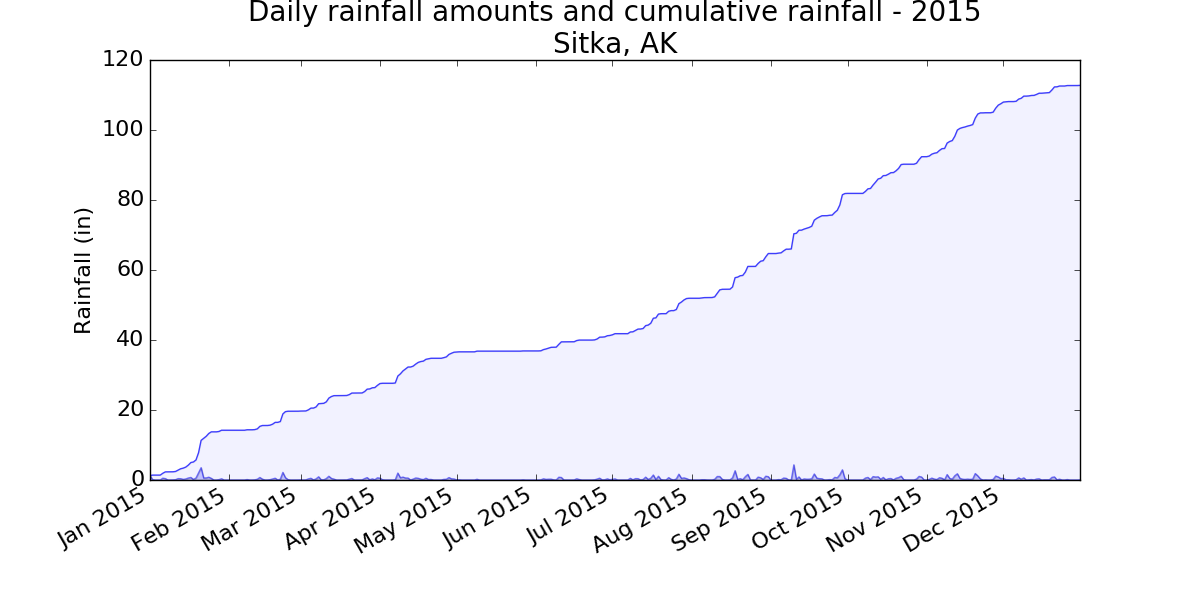
fig**.**autofmt\_xdate()

plt**.**ylabel("Rainfall (in)", fontsize**=**16)

plt**.**tick\_params(axis**=**'both', which**=**'major', labelsize**=**16)

plt**.**show()

输出：



### 第17章

##### 练习17-1 其他语言

修改python\_repos.py中的API调用，使其在生成的图表中显示使用其他语言编写的最受欢迎的项目。请尝试语言JavaScript、Ruby、C、Java、Perl、Haskell和Go。

注意：在这个练习中，对python\_repos.py的代码进行了修订，以防项目没有描述时导致程序崩溃；另外，将字号设置放到了my\_style中。

import requests

import pygal

from pygal.style import LightColorizedStyle **as** LCS, LightenStyle **as** LS

*# 执行API调用并存储响应。*

url **=** 'https://api.github.com/search/repositories?q=language:javascript&sort=stars'

r **=** requests**.**get(url)

**print**("Status code:", r**.**status\_code)

*# 将API响应存储在一个变量中。*

response\_dict **=** r**.**json()

**print**("Total repositories:", response\_dict['total\_count'])

*# 探索有关仓库的信息。*

repo\_dicts **=** response\_dict['items']

names, plot\_dicts **=** [], []

**for** repo\_dict **in** repo\_dicts:

names**.**append(repo\_dict['name'])

*# 项目被删除后，它依然会被列出并有星级，但没有描述。*

*# 将描述用作标签时，如果它为None，将引发异常。*

**if** repo\_dict['description']:

desc **=** repo\_dict['description']

**else**:

desc **=** 'No description provided.'

plot\_dict **=** {

'value': repo\_dict['stargazers\_count'],

'label': desc,

'xlink': repo\_dict['html\_url'],

}

plot\_dicts**.**append(plot\_dict)

*# 可视化。*

my\_style **=** LS('#333366', base\_style**=**LCS)

my\_style**.**title\_font\_size **=** 24

my\_style**.**label\_font\_size **=** 14

my\_style**.**major\_label\_font\_size **=** 18

my\_config **=** pygal**.**Config()

my\_config**.**x\_label\_rotation **=** 45

my\_config**.**show\_legend **=** False

my\_config**.**truncate\_label **=** 15

my\_config**.**show\_y\_guides **=** False

my\_config**.**width **=** 1000

chart **=** pygal**.**Bar(my\_config, style**=**my\_style)

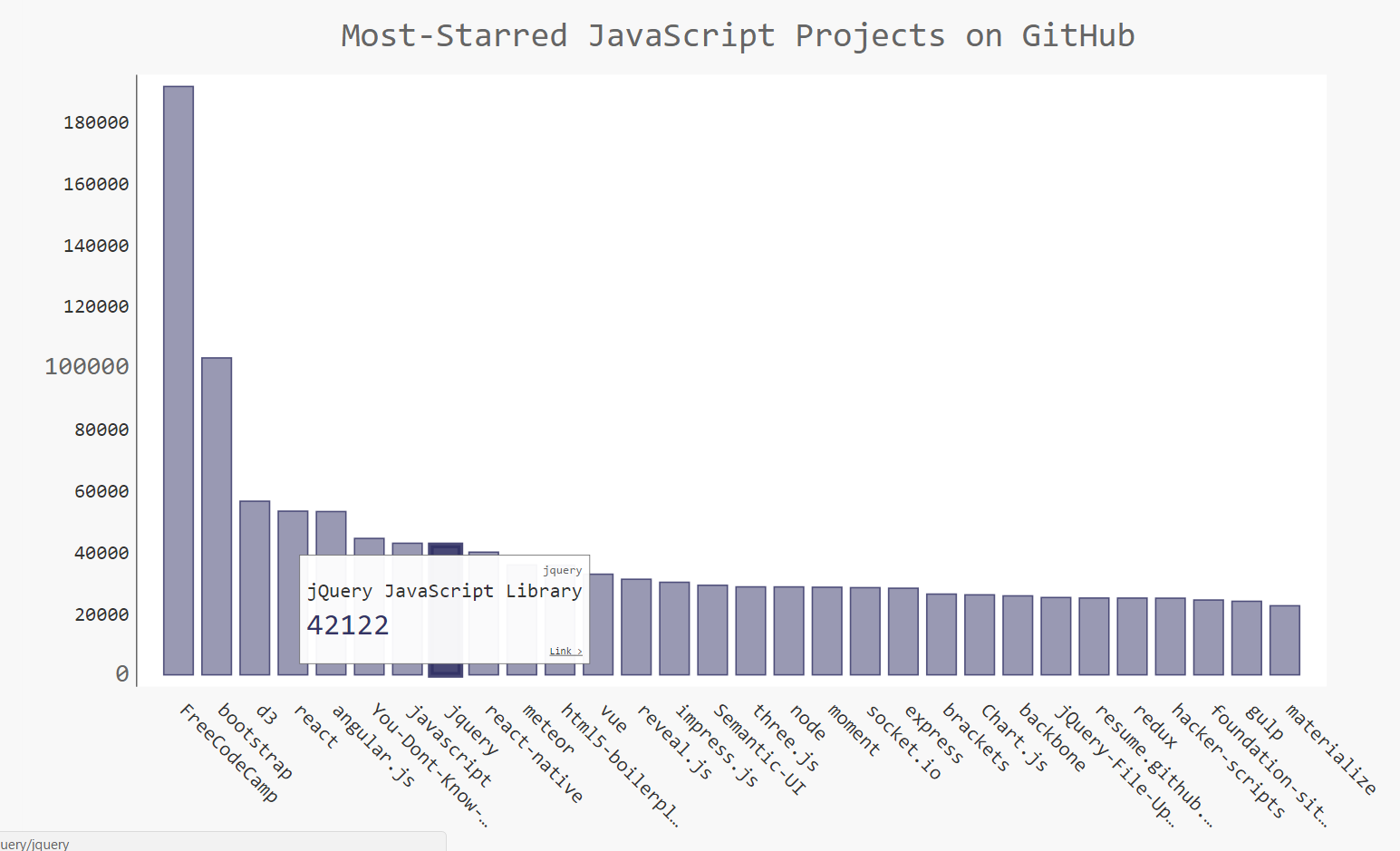
chart**.**title **=** 'Most-Starred JavaScript Projects on GitHub'

chart**.**x\_labels **=** names

chart**.**add('', plot\_dicts)

chart**.**render\_to\_file('js\_repos.svg')

输出：



##### 17-2 最活跃的讨论

使用hn\_submissions.py中的数据，创建一个条形图，显示Hacker News上当前最活跃的讨论。条形的高度应对应于文章得到的评论数量；条形的标签应包含文章的标题；而每个条形应是到文章讨论页面的链接。

import requests

import pygal

from pygal.style import LightColorizedStyle **as** LCS, LightenStyle **as** LS

from operator import itemgetter

*# 执行API调用并存储响应。*

url **=** 'https://hacker-news.firebaseio.com/v0/topstories.json'

r **=** requests**.**get(url)

**print**("Status code:", r**.**status\_code)

*# 处理有关每篇文章的信息。*

submission\_ids **=** r**.**json()

submission\_dicts **=** []

**for** submission\_id **in** submission\_ids[:30]:

*# 对于每篇文章，都执行一个API调用。*

url **=** ('https://hacker-news.firebaseio.com/v0/item/' **+**

str(submission\_id) **+** '.json')

submission\_r **=** requests**.**get(url)

**print**(submission\_r**.**status\_code)

response\_dict **=** submission\_r**.**json()

submission\_dict **=** {

'title': response\_dict['title'],

'link': 'http://news.ycombinator.com/item?id=' **+** str(submission\_id),

'comments': response\_dict**.**get('descendants', 0)

}

submission\_dicts**.**append(submission\_dict)

submission\_dicts **=** sorted(submission\_dicts, key**=**itemgetter('comments'),

reverse**=**True)

**for** submission\_dict **in** submission\_dicts:

**print**("\nTitle:", submission\_dict['title'])

**print**("Discussion link:", submission\_dict['link'])

**print**("Comments:", submission\_dict['comments'])

titles, plot\_dicts **=** [], []

**for** submission\_dict **in** submission\_dicts:

titles**.**append(submission\_dict['title'])

plot\_dict **=** {

'value': submission\_dict['comments'],

'label': submission\_dict['title'],

'xlink': submission\_dict['link'],

}

plot\_dicts**.**append(plot\_dict)

*# 可视化。*

my\_style **=** LS('#333366', base\_style**=**LCS)

my\_style**.**title\_font\_size **=** 24

my\_style**.**label\_font\_size **=** 14

my\_style**.**major\_label\_font\_size **=** 18

my\_config **=** pygal**.**Config()

my\_config**.**x\_label\_rotation **=** 45

my\_config**.**show\_legend **=** False

my\_config**.**truncate\_label **=** 15

my\_config**.**show\_y\_guides **=** False

my\_config**.**width **=** 1000

my\_config**.**y\_title **=** 'Number of Comments'

chart **=** pygal**.**Bar(my\_config, style**=**my\_style)

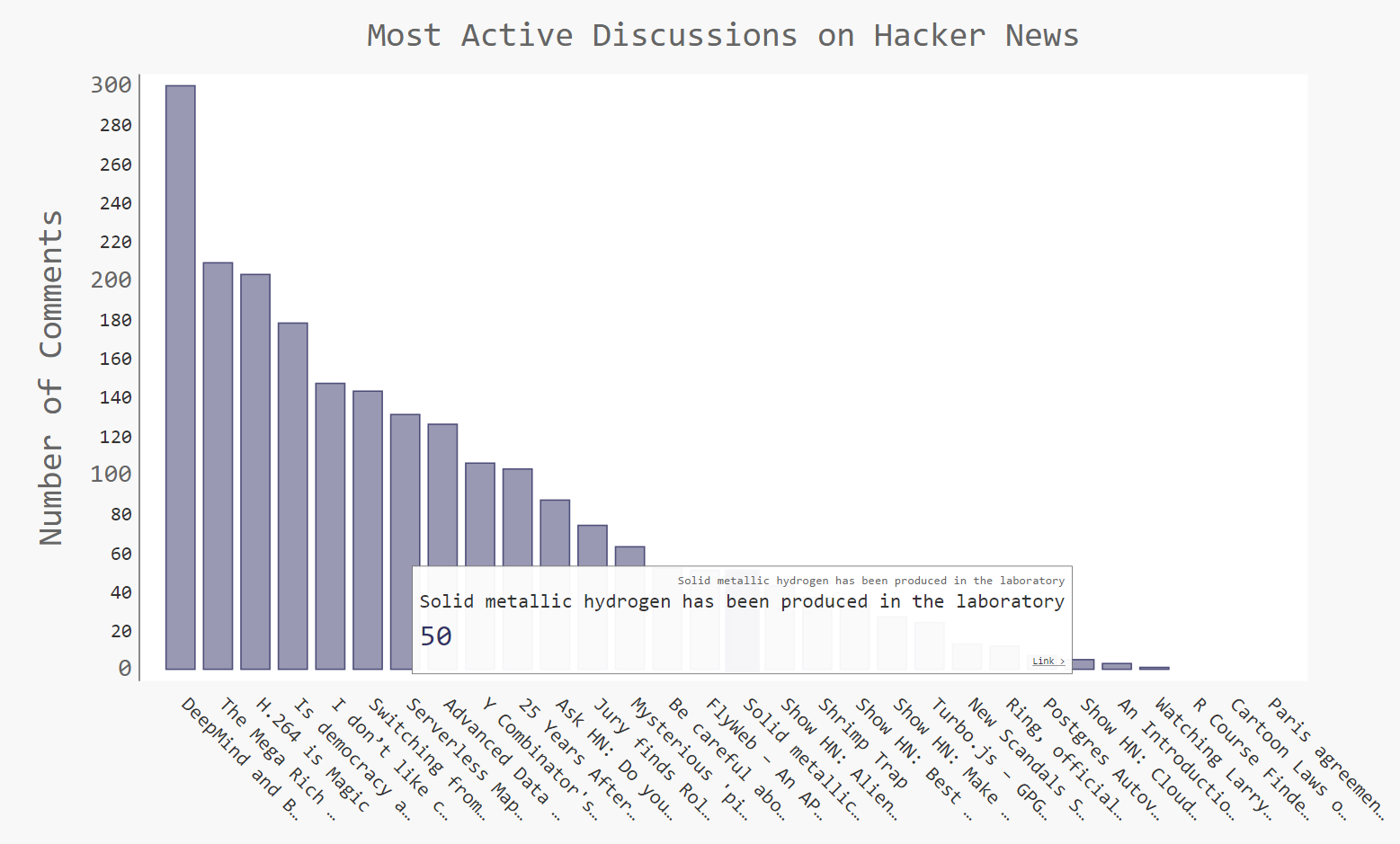
chart**.**title **=** 'Most Active Discussions on Hacker News'

chart**.**x\_labels **=** titles

chart**.**add('', plot\_dicts)

chart**.**render\_to\_file('hn\_discussions.svg')

输出：



##### 练习17-3测试python\_repos.py

在python\_repos.py中，我们打印status\_code的值，以核实API调用是否成功了。请编写一个名为test\_python\_repos.py的程序，它使用unittest来断言status\_code的值为200。想想你还可做出哪些断言，如返回的条目（item）数符合预期；仓库总数超过特定的值。

注意：在这个练习中，对python\_repos.py的代码进行了修订，以防项目没有描述时导致程序崩溃；另外，将字号设置放到了my\_style中。

编写测试可促使你以合理的方式组织代码，确保可对其进行测试。下面是修订后的python\_repos.py，其执行的所有操作都是在4个函数中完成的：

import requests

import pygal

from pygal.style import LightColorizedStyle **as** LCS, LightenStyle **as** LS

**def** **get\_response**():

"""执行API调用，并返回响应。"""

url **=** 'https://api.github.com/search/repositories?q=language:python&sort=stars'

r **=** requests**.**get(url)

**return** r

**def** **get\_repo\_dicts**(response):

"""返回一系列表示最受欢迎仓库的字典。"""

response\_dict **=** r**.**json()

repo\_dicts **=** response\_dict['items']

**return** repo\_dicts

**def** **get\_names\_plot\_dicts**(repo\_dicts):

""" 处理这些表示仓库的字典，从中提取用于绘制图形的数据。"""

names, plot\_dicts **=** [], []

**for** repo\_dict **in** repo\_dicts:

names**.**append(repo\_dict['name'])

*# 有些项目没有描述，这将导致给相应的条形添加标签时出错。*

*# 因此，如果项目没有描述，就指定一个标签。*

description **=** repo\_dict['description']

**if** **not** description:

description **=** "No description provided."

plot\_dict **=** {

'value': repo\_dict['stargazers\_count'],

'label': description,

'xlink': repo\_dict['html\_url'],

}

plot\_dicts**.**append(plot\_dict)

**return** names, plot\_dicts

**def** **make\_visualization**(names, plot\_dicts):

"""可视化最受欢迎的项目。"""

my\_style **=** LS('#333366', base\_style**=**LCS)

my\_style**.**title\_font\_size **=** 24

my\_style**.**label\_font\_size **=** 14

my\_style**.**major\_label\_font\_size **=** 18

my\_config **=** pygal**.**Config()

my\_config**.**x\_label\_rotation **=** 45

my\_config**.**show\_legend **=** False

my\_config**.**truncate\_label **=** 15

my\_config**.**show\_y\_guides **=** False

my\_config**.**width **=** 1000

chart **=** pygal**.**Bar(my\_config, style**=**my\_style)

chart**.**title **=** 'Most-Starred Python Projects on GitHub'

chart**.**x\_labels **=** names

chart**.**add('', plot\_dicts)

chart**.**render\_to\_file('python\_repos.svg')

r **=** get\_response()

repo\_dicts **=** get\_repo\_dicts(r)

names, plot\_dicts **=** get\_names\_plot\_dicts(repo\_dicts)

make\_visualization(names, plot\_dicts)

现在可以给这些函数编写测试了。在这里，我们做如下测试：获得的响应的状态码为200；在表示第一个仓库的字典中，包含我们期望在每个表示仓库的字典中都有的键。

import unittest

import python\_repos\_for\_testing **as** pr

**class** **PythonReposTestCase**(unittest**.**TestCase):

"""测试python\_repos.py。"""

**def** **setUp**(self):

"""调用所有的函数，并分别测试各个方面。"""

self**.**r **=** pr**.**get\_response()

self**.**repo\_dicts **=** pr**.**get\_repo\_dicts(self**.**r)

self**.**repo\_dict **=** self**.**repo\_dicts[0]

self**.**names, self**.**plot\_dicts **=** pr**.**get\_names\_plot\_dicts(self**.**repo\_dicts)

**def** **test\_get\_response**(self):

"""测试获得了有效的响应。"""

self**.**assertEqual(self**.**r**.**status\_code, 200)

**def** **test\_repo\_dicts**(self):

"""测试获得了期望的数据。"""

*# 应获得30个描述仓库的字典。*

self**.**assertEqual(len(self**.**repo\_dicts), 30)

*# 描述仓库的字典应包含必要的键。*

required\_keys **=** ['name', 'owner', 'stargazers\_count', 'html\_url']

**for** key **in** required\_keys:

self**.**assertTrue(key **in** self**.**repo\_dict**.**keys())

unittest**.**main()

输出：

..

----------------------------------------------------------------------

Ran 2 tests in 1.969s

OK