8.6.2 导入特定的函数

还可以导入模块中的特定函数，这种导入方法的语法如下：

```
from module_name import function_name
```

通过用逗号分隔函数名，可根据需要从模块中导入任意数量的函数：

```
from module_name import function_0, function_1, function_2
```

对于前面的making_pizzas.py示例，如果只想导入要使用的函数，代码将类似于下面这样：

In [3]:
```
from pizza import make_pizza as mp
make_pizza(16, 'pepperoni')
make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

```
Making a 16-inch pizza with the following toppings:
- pepperoni

Making a 12-inch pizza with the following toppings:
- mushrooms
- green peppers
- extra cheese
```

使用这种语法时，调用函数时无须使用句点。 由于在import语句中显式地导入了函数make_pizza()，调用时只需指定其名称即可。

8.6.3 使用as给函数指定别名

如果要导入函数的名称可能与程序中现有的名称冲突，或者函数的名称太长，可指定简短而独一无二的别名： 函数的另一个名称，类似于外号。 要给函数取这种特殊外号，需要在导入它时指定。 下面给函数make_pizza()指定了别名mp()。 这是在import语句中使用make_pizza as mp 实现的，关键字as将函数重命名为指定的别名：

In [3]:
```
from pizza import make_pizza as mp
from pizza-hut import make_pizza as ph_mp
from pizza-mozzile import make_pizza as pm_mp

ph_mp(16, 'pepperoni')
pm_mp(12, 'mushrooms', 'green peppers', 'extra cheese')
```

```
Making a 16-inch pizza with the following toppings:
- pepperoni

Making a 12-inch pizza with the following toppings:
- mushrooms
- green peppers
- extra cheese
```

8.6.4 使用as给模块指定别名

还可以给模块指定别名。通过给模块指定简短的别名（如给模块pizza指定别名p），让你能够更轻松地调用模块中的函数。 相比于pizza.make_pizza()，p.make_pizza()更为简洁：

In [5]:
```
import pizza as p
```

```
p.make_pizza(16, 'pepperoni')
p.make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

```
Making a 16-inch pizza with the following toppings:
- pepperoni

Making a 12-inch pizza with the following toppings:
- mushrooms
- green peppers
- extra cheese
```

上述import语句给模块pizza指定了别名p，但该模块中所有函数的名称都没变。 要调用函数make_pizza()，可编写代码p.make_pizza()而非pizza.make_pizza()。 这样不仅代码更简洁，还让你不用再关注模块名，只专注于描述性的函数名。 这些函数名明确指出了函数的功能，对于理解代码而言，比模块名更重要。

8.6.5 导入模块中的所有函数

使用星号（*）运算符可让Python导入模块中的所有函数：

In [7]:
```
from pizza import *
make_pizza(16, 'pepperoni')
make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

```
Making a 16-inch pizza with the following toppings:
- pepperoni

Making a 12-inch pizza with the following toppings:
- mushrooms
- green peppers
- extra cheese
```

import语句中的星号让Python将模块pizza中的每个函数都复制到这个程序文件中。 由于导入了每个函数，可通过名称来调用每个函数，而无须使用句点表示法。 然而，使用并非自己编写的大型模块时，最好不要采用这种导入方法。这是因为如果模块中有函数的 名称与当前项目中使用的名称相同，可能导致意想不到的结果： Python可能遇到多个名称相同的函数或变量，进而覆盖函数，而不是分别导入所有的函数。 最佳的做法是，要么只导入需要使用的函数，要么导入整个模块并使用句点表示法。 这让代码更清晰，更容易阅读和理解。 这里之所以介绍这种导入方法，只是想让你在阅读别人编写的代码时，能够理解。

8.7 函数编写指南

编写函数时，需要牢记几个细节。 应给函数指定描述性名称，且只在其中使用小写字母和下划线。 描述性名称可帮助你和别人明白代码想要做什么。 给模块命名时也应遵循上述约定。 每个函数都应包含简要地阐述其功能的注释。 该注释应紧跟在函数定义后面，并采用文档字符串格式。 文档良好的函数让其他程序员只需阅读文档字符串中的描述就能够使用它。 他们完全可以相信代码如描述的那样运行，并且只要知道函数的名称、需要的实参以及返回值的类型，就能在自己的程序中使用它。

In [ ]:
```
PEP8建议代码行的长度不要超过79字符，这样只要编辑器窗口适中，就能看到整行代码。
如果形参很多，导致函数定义的长度超过了79字符，可在函数定义中输入左括号后按回车键，并在下一行按两次Tab键，

所有import语句都应放在文件开头。
唯一例外的情形是，在文件开头使用了注释来描述整个程序。
```

8.8 小结

在本章中，学习了：

- 如何编写函数，以及如何传递实参，让函数能够访问完成其工作所需的信息；
- 如何使用位置实参和关键字实参，以及如何接受任意数量的实参；
- 显示输出的函数和返回值的函数；
- 如何将函数同列表、字典、if语句和while循环结合起来使用；
- 如何将函数存储在称为模块的独立文件中，让程序文件更简单、更易于理解。
- 最后，学习了函数编写指南，遵循这些指南可让程序始终结构良好，并对你和其他人来说易于阅读。

程序员的目标之一是，编写简单的代码来完成任务，而函数有助于实现这样的目标。 它们让你编写好代码块并确定其能够正确运行后，就可置之不理。 确定函数能够正确地完成其工作后，就可以接着投身于下一个编码任务。

函数让你编写代码一次后，想重用它们多少次就重用多少次。 需要运行函数中的代码时，只需编写一行函数调用代码，就可让函数完成其工作。 需要修改函数的行为时，只需修改一个代码块，而所做的修改将影响调用这个函数的每个地方。

使用函数让程序更容易阅读，而良好的函数名概述了程序各个部分的作用。 相对于阅读一系列的代码块，阅读一系列函数调用让你能够更快地明白程序的作用。 函数还让代码更容易测试和调试。

如果程序使用一系列的函数来完成其任务，而其中的每个函数都完成一项具体的工作，测试和维护起来将容易得多：

- 可编写分别调用每个函数的程序，并测试每个函数是否在它可能遇到的各种情形下都能正确地运行。
- 经过这样的测试后你就能充满信心，深信每次调用这些函数时，它们都将正确地运行。

In [ ]:
```
# The end
```