

目录

第 1 章 入门准备

01 开篇词：你为什么要学 Python ？

02 我会怎样带你学 Python ？

03 让 Python 在你的电脑上安家落户

04 如何运行 Python 代码 ？

第 2 章 通用语言特性

05 数据的名字和种类—变量和类型

06 一串数据怎么存—列表和字符串

07 不只有一条路—分支和循环

08 将代码放进盒子—函数

09 知错能改—错误处理、异常机制

10 定制一个模子—类

11 更大的代码盒子—模块和包 [最近阅读](#)

12 练习—密码生成器

第 3 章 Python 进阶语言特性

13 这么多的数据结构（一）：列表、元祖、字符串

14 这么多的数据结构（二）：字典、集合

15 Python大法初体验：内置函数

16 深入理解下迭代器和生成器

17 生成器表达式和列表生成式

18 把盒子升级为豪宅：函数进阶

19 让你的模子更好用：类进阶

20 从小独栋升级为别墅区：函数式编程

11 更大的代码盒子—模块和包

更新时间：2019-09-06 09:45:46



“

勤学如春起之苗，不见其增，日有所长。

——陶潜

”

什么是模块

之前介绍过两种运行 Python 代码的方式，一种是解释器的交互模式，另一种是直接运行 Python 代码文件。

在 Python 中，每一个 Python 代码文件就是一个模块。写程序时，我们可以将代码分散在不同的模块（文件）中，然后在一个模块里引用另一个模块的内容。

模块的导入

在一个模块中引用（导入）另一个模块，可以使用 `import` 语句：

```
import 模块名
```

这里的模块名是除去 `.py` 后缀的文件名称。如，想要导入模块 `abc.py`，只需 `import abc`。

`import` 模块之后，就可以使用被导入模块中的名字（变量、函数类）。方式如下：

```
模块名.变量
模块名.函数
模块名.类
```

导入及使用模块示例

我们用个例子来试验下模块的导入和使用，在这个例子中，农民种下果树，然后等待果树结果收获。

<div>← 慕课专栏</div> <div>≡ 你的第一本Python基础入门书 / 11 更大的代码盒子—模块和包</div>	
目录	tree_farmer # 目录名 __tree.py # 文件名 __farmer.py # 文件名
第 1 章 入门准备	
01 开篇词：你为什么要学 Python？	第一个模块名为 <code>tree.py</code> ，内容如下：
02 我会怎样带你学 Python？	<pre>import random fruit_name = '' def harvest(): return [fruit_name] * random.randint(1, 9)</pre>
03 让 Python 在你的电脑上安家落户	代码中各个变量和函数的功能如下：
04 如何运行 Python 代码？	<ul style="list-style-type: none">• <code>fruit_name</code> 用来保存水果名称。将在函数 <code>harvest()</code> 中使用；• <code>random.randint(1, 9)</code>，随机生成 1 ~ 9 中的一个数；• <code>[fruit_name] * 数字</code>，该形式是将列表项重复若干遍。比如执行 <code>['X'] * 3</code> 将得到 <code>['X', 'X', 'X']</code>；• 总体而言，<code>harvest()</code> 函数返回一个包含 1 ~ 9 个列表项的列表，其中每个项都是 <code>fruit_name</code> 的值。
第 2 章 通用语言特性	第二个模块名为 <code>farmer.py</code> ，内容如下：
05 数据的名字和种类—变量和类型	<pre>import tree print('种下一棵果树。') tree.fruit_name = 'apple' print('等啊等，树长大了，可以收获啦！') fruits = tree.harvest() print(fruits)</pre>
06 一串数据怎么存—列表和字符串	代码中，
07 不只有一条路—分支和循环	<ul style="list-style-type: none">• 第一行用 <code>import tree</code> 将 <code>tree.py</code> 模块导入进来（使用 <code>import</code> 导入时不需要写 <code>.py</code> 后缀）；• 导入 <code>tree</code> 模块后，就可以使用其中的变量和函数了。将 <code>tree.fruit_name</code> 设置为 <code>apple</code>，调用 <code>tree.harvest()</code> 来收获 <code>apple</code>。
08 将代码放进盒子—函数	执行下模块 <code>farmer.py</code> 看看：
09 知错能改—错误处理、异常机制	<pre>→ ~ python3 farmer.py 种下一棵果树。 等啊等，树长大了，可以收获啦！ ['apple', 'apple', 'apple', 'apple']</pre>
10 定制一个模子—类	说明： <code>apple</code> 随机出现 1 ~ 9 个，所以你的结果可能和这里不一样。
11 更大的代码盒子—模块和包 最近阅读	
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元组、字符串	
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

<div>← 慕课专栏</div> <div>目录</div> <div>第 1 章 入门准备</div> <div>01 开篇词：你为什么要学 Python？</div> <div>02 我会怎样带你学 Python？</div> <div>03 让 Python 在你的电脑上安家落户</div> <div>04 如何运行 Python 代码？</div> <div>第 2 章 通用语言特性</div> <div>05 数据的名字和种类—变量和类型</div> <div>06 一串数据怎么存—列表和字符串</div> <div>07 不只有一条路—分支和循环</div> <div>08 将代码放进盒子—函数</div> <div>09 知错能改—错误处理、异常机制</div> <div>10 定制一个模子—类</div> <div>11 更大的代码盒子—模块和包 最近阅读</div> <div>12 练习—密码生成器</div> <div>第 3 章 Python 进阶语言特性</div> <div>13 这么多的数据结构（一）：列表、元祖、字符串</div> <div>14 这么多的数据结构（二）：字典、集合</div> <div>15 Python大法初体验：内置函数</div> <div>16 深入理解下迭代器和生成器</div> <div>17 生成器表达式和列表生成式</div> <div>18 把盒子升级为豪宅：函数进阶</div> <div>19 让你的模子更好用：类进阶</div> <div>20 从小独栋升级为别墅区：函数式编程</div>	<div>≡ 你的第一本Python基础入门书 / 11 更大的代码盒子—模块和包</div> <div>ee.py 的内容。</div> <div>标准库模块的导入</div> <div>上面的例子中，我们自己定义了模块，然后在其它模块中使用它。其中有个地方不知道你有没有注意到， <code>tree.py</code> 的第一行代码是 <code>import random</code>，<code>random</code> 并不是我们所定义的模块，那它是从哪里来的呢？</div> <div><code>random</code> 是标准库中的一个模块。标准库是由 Python 官方开发的代码库，和解释器一起打包分发，其中包含非常多实用的模块，我们在使用时直接 <code>import</code> 进来即可。</div> <div>执行模块时传入参数</div> <div>刚才我们用这种方式来执行模块：</div> <div>python3 模块文件名</div> <div>其实我们还可以进一步将参数传递到模块中去，像这样：</div> <div>python3 模块文件名 参数1 ...参数n</div> <div>参数传递到模块中以后，我们可以通过 <code>sys</code> 模块来取出这些参数，参数放在 <code>sys.argv</code> 列表中：</div> <div>import sys 模块文件名 = sys.argv[0] 参数1 = sys.argv[1] 参数N = sys.argv[N]</div> <div>首先需要导入 <code>sys</code> 模块，这是个标准库中的模块。<code>sys.argv</code> 是个列表，执行模块时被传递进来的参数保存在其中，它的列表项分别为：</div> <div><ul style="list-style-type: none"><code>sys.argv[0]</code> 保存当前被执行模块的文件名<code>sys.argv[1]</code> 保存第 1 个参数<code>sys.argv[2]</code> 保存第 2 个参数依次类推</div> <div>之前种果树那个例子中，<code>farmer.py</code> 固定种苹果树，我们可以改进一下，具体种什么树由传递的模块参数来决定。</div> <div>修改 <code>farmer.py</code> 的代码，内容如下：</div> <div>import sys # 新增 import tree print('种下一棵果树。') tree.fruit_name = sys.argv[1] # 将 'apple' 改为 参数 sys.argv[1] print('等啊等，树长大了，可以收获啦！') fruits = tree.harvest() print(fruits)</div>
---	--

<div>← 慕课专栏</div> <div>你的第一本Python基础入门书 / 11 更大的代码盒子—模块和包</div>	
目录	
第 1 章 入门准备	
01 开篇词：你为什么要学 Python ？	
02 我会怎样带你学 Python ？	
03 让 Python 在你的电脑上安家落户	
04 如何运行 Python 代码 ？	
第 2 章 通用语言特性	
05 数据的名字和种类—变量和类型	
06 一串数据怎么存—列表和字符串	
07 不只有一条路—分支和循环	
08 将代码放进盒子—函数	
09 知错能改—错误处理、异常机制	
10 定制一个模子—类	
11 更大的代码盒子—模块和包 最近阅读	<div>→ ~ python3 farmer.py banana</div> <div>种下一棵果树。</div> <div>等啊等，树长大了，可以收获了！</div> <div>[‘banana’， ‘banana’， ‘banana’]</div>
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元祖、字符串	
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	

在这个例子中 `sys.argv` 的值是：

- `sys.argv[0]`： farmer.py
- `sys.argv[1]`： banana

什么是包

之前我们将定义的两个模块放在同一目录下，然后通过 `import` 语句来相互引用，这是一种扁平的模块组织结构，当模块数量很大的时候就很不灵活了，也难以维护。

Python 中可以用文件树这样的树形结构来组织模块，这种组织形式下的模块集合称为包（Package）。

比如包的结构可以是这样的：

```
包/
├── __init__.py
├── 模块1.py
├── 模块2.py
├── 子包1/
│   ├── __init__.py
│   ├── 模块3.py
│   └── 模块4.py
└── 子包2/
    ├── __init__.py
    ├── 模块5.py
    └── 孙子包1/
        ├── __init__.py
        └── 模块6.py
```

这是个很明显的层级结构——包里面包含子包、子包包含孙子包…… 单独将子包或孙子包拿出来，它们也是包。

包的存在形式是目录，模块的存在形式是目录下的文件。所以我们可以很容易地构造出这样一个包，只要在文件系统中创建相应的目录和文件即可。

需要注意的是，每个层级的包下都需要有一个 `__init__.py` 模块。这是因为只有当目录中存在 `__init__.py` 时，Python 才会把这个目录当作包。

包的导入

导入包中模块的方法是：

```
import 包.子包.模块
```

←

慕课专栏

你的第一本Python基础入门书 / 11 更大的代码盒子—模块和包

目录

第 1 章 入门准备

01 开篇词：你为什么要学 Python ？

02 我会怎样带你学 Python ？

03 让 Python 在你的电脑上安家落户

04 如何运行 Python 代码 ？

第 2 章 通用语言特性

05 数据的名字和种类—变量和类型

06 一串数据怎么存—列表和字符串

07 不只有一条路—分支和循环

08 将代码放进盒子—函数

09 知错能改—错误处理、异常机制

10 定制一个模子—类

11 更大的代码盒子—模块和包 [最近阅读](#)

12 练习—密码生成器

第 3 章 Python 进阶语言特性

13 这么多的数据结构（一）：列表、元祖、字符串

14 这么多的数据结构（二）：字典、集合

15 Python大法初体验：内置函数

16 深入理解下迭代器和生成器

17 生成器表达式和列表生成式

18 把盒子升级为豪宅：函数进阶

19 让你的模子更好用：类进阶

20 从小独栋升级为别墅区：函数式编程

如，从上面示例的包结构中，

导入 `模块1.py`，使用：

```
import 包.模块1
```

导入 `模块3.py`，使用：

```
import 包.子包1.模块3
```

导入 `模块6.py`，使用：

```
import 包.子包2.孙子包1.模块6
```

为什么需要模块和包

模块的存在是为了更好的组织代码。将不同功能的代码分散在不同模块中，清晰地划分出各个模块的职责，有利于使用和维护代码，同时也可避免模块中的内容过长。

包的存在是为了更好的组织模块。与模块同理，包在更高的抽象层次上组织着代码。

总结

模块可以更好的组织代码，它的存在形式是文件。包的可以更好的组织模块，它的存在形式是目录。

导入模块使用 `import` 语句：

```
import 模块名
```

导入包下的模块：

```
import 包名.模块名
```

模块导入后，可以使用该模块中所定义的名字（变量、函数类）。方式如下：

```
模块名.变量
模块名.函数
模块名.类
```

←10 定制一个模子—类

12 练习—密码生成器→

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

<div>← 慕课专栏</div> <div>你的第一本Python基础入门书 / 11 更大的代码盒子—模块和包</div>	
目录	<div>!</div> <div>目前暂无任何讨论</div>
第 1 章 入门准备	
01 开篇词：你为什么要学 Python ？	
02 我会怎样带你学 Python ？	千学不如一看，千看不如一练
03 让 Python 在你的电脑上安家落户	
04 如何运行 Python 代码 ？	
第 2 章 通用语言特性	
05 数据的名字和种类—变量和类型	
06 一串数据怎么存—列表和字符串	
07 不只有一条路—分支和循环	
08 将代码放进盒子—函数	
09 知错能改—错误处理、异常机制	
10 定制一个模子—类	
11 更大的代码盒子—模块和包 最近阅读	
12 练习—密码生成器	
第 3 章 Python 进阶语言特性	
13 这么多的数据结构（一）：列表、元祖、字符串	
14 这么多的数据结构（二）：字典、集合	
15 Python大法初体验：内置函数	
16 深入理解下迭代器和生成器	
17 生成器表达式和列表生成式	
18 把盒子升级为豪宅：函数进阶	
19 让你的模子更好用：类进阶	
20 从小独栋升级为别墅区：函数式编程	