

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

12. 错误、调试和测试

13. IO编程

14. 进程和线程

廖雪峰的官方网站 [Java教程](#) [Python教程](#) [JavaScript教程](#) [SQL教程](#) [手写Spring](#) [手写Tomcat](#) [区块链教程](#) [Git教程](#) [Makefile教程](#) [博客](#)

17. 常用第三方模块

18. 图形界面

19. 网络编程

20. 电子邮件

21. 访问数据库

22. Web开发

23. 异步IO

24. FAQ

25. 期末总结

下载PDF

模块



廖雪峰

资深软件开发工程师，业余马拉松选手。

在计算机程序的开发过程中，随着程序代码越写越多，在一个文件里代码就会越来越长，越来越不容易维护。

为了编写可维护的代码，我们把很多函数分组，分别放到不同的文件里，这样，每个文件包含的代码就相对较少，很多编程语言都采用这种组织代码的方式。在Python中，一个.py文件就称之为一个模块（Module）。

使用模块有什么好处？

最大的好处是大大提高了代码的可维护性。其次，编写代码不必从零开始。当一个模块编写完毕，就可以被其他地方引用。我们在编写程序的时候，也经常引用其他模块，包括Python内置的模块和来自第三方的模块。

使用模块还可以避免函数名和变量名冲突。相同名字的函数和变量完全可以分别存在不同的模块中，因此，我们自己在编写模块时，不必考虑名字会与其他模块冲突。但是也要注意，尽量不要与内置函数名字冲突。点[这里](#)查看Python的所有内置函数。

你也许还想到，如果不同的人编写的模块名相同怎么办？为了避免模块名冲突，Python又引入了按目录来组织模块的方法，称为包（Package）。

举个例子，一个 `abc.py` 的文件就是一个名字叫 `abc` 的模块，一个 `xyz.py` 的文件就是一个名字叫 `xyz` 的模块。

现在，假设我们的 `abc` 和 `xyz` 这两个模块名字与其他模块冲突了，于是我们可以通过包来组织模块，避免冲突。方法是选择一个顶层包名，比如 `mycompany`，按照如下目录存放：

```
mycompany
├── __init__.py
├── abc.py
└── xyz.py
```

引入了包以后，只要顶层的包名不与别人冲突，那所有模块都不会与别人冲突。现在，`abc.py` 模块的名字就变成了 `mycompany.abc`，类似的，`xyz.py` 的模块名变成了 `mycompany.xyz`。

请注意，每一个包目录下面都会有一个 `__init__.py` 的文件，这个文件是必须存在的，否则，Python就把这个目录当成普通目录，而不是一个包。`__init__.py` 可以是空文件，也可以有Python代码，因为 `__init__.py` 本身就是一个模块，而它的模块名就是 `mycompany`。

类似的，可以有多个目录，组成多级层次的包结构。比如如下的目录结构：

```
mycompany
├── web
│   ├── __init__.py
│   ├── utils.py
│   └── www.py
├── __init__.py
├── abc.py
└── utils.py
```

文件 `www.py` 的模块名就是 `mycompany.web.www`，两个文件 `utils.py` 的模块名分别是 `mycompany.utils` 和 `mycompany.web.utils`。

自己创建模块时要注意命名，不能和Python自带的模块名称冲突。例如，系统自带了 `sys` 模块，自己的模块就不可命名为 `sys.py`，否则将无法导入系统自带的 `sys` 模块。

`mycompany.web` 也是一个模块，请指出该模块对应的.py文件。

总结

模块是一组Python代码的集合，可以使用其他模块，也可以被其他模块使用。

创建自己的模块时，要注意：

- 模块名要遵循Python变量命名规范，不要使用中文、特殊字符；
- 模块名不要和系统模块名冲突，最好先查看系统是否已存在该模块，检查方法是在Python交互环境执行 `import abc`，若成功则说明系统存在此模块。

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

12. 错误、调试和测试

13. IO编程

14. 进程和线程

15. 正则表达式

16. 常用内建模块

17. 常用第三方模块

18. 图形界面

19. 网络编程

20. 电子邮件

21. 访问数据库

22. Web开发

23. 异步IO

24. FAQ

25. 期末总结

[下载PDF](#)


Comments

Comments loaded. To post a comment, please [Sign In](#)



@ 2025/7/22 11:30:43

```
mycompany
├ web
└ └ __init__.py
```



群青 @ 2025/7/13 23:47:14

mycompany.web也是一个模块，请指出该模块对应的.py文件。

```
mycompany
├ web
└ └ __init__.py
```

(怪不得之前同目录下导入函数，文件名带小数点不成功，原来是会把文件名判断成模块下的.py文件)



路 @ 2025/6/3 21:12:38

9. 模块 done



日向夏橘 @ 2025/5/9 06:55:17

打卡



张中二_invisible @ 2020/2/20 08:08:13

mycompany.web对应的为啥是 web下面的__init__.py，而不是web下的别的？



别叫我贵哥007 @ 2020/2/24 22:14:47

我也觉得，应该三个都是啊，有没有大佬回答一下



tb92288474 @ 2020/2/28 23:04:44

你可以把这看成文件夹，mycompany.web对应的是web文件夹下的默认文件(__init__.py)

mycompany.web.xxx对应的才是web文件夹下的其他文件

mycompany:

web----->web:

__init__.py __init__.py

xxxxxx xxxxxxxxx

xxxxxx xxxxxxxxx

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

12. 错误、调试和测试

13. IO编程

14. 进程和线程

15. 正则表达式

16. 常用内建模块

17. 常用第三方模块

18. 图形界面

19. 网络编程

20. 电子邮件

21. 访问数据库

22. Web开发

23. 异步IO

24. FAQ

25. 期末总结



用户6731664059 @ 2020/4/14 02:06:41

mycompany.web.a是web目录下的a.py

mycompany.web.b是web目录下的b.py

mycompany.web.c是web目录下的c.py

mycompany.web其实完整的称呼是mycompany.web.__init__.py

这个init文件是python包结构的指向的默认文件，没这个文件python就不承认这个web目录是包了
如果你喜欢，你如果改源码把mycompany.web包默认指向a.py。。。



陽22348 @ 2020/4/29 11:09:15

因为import包的时候会自动调用__init__.py进行初始化吧



用户6786697361_46373 @ 2020/5/8 21:24:34

未指定具体的模块，就使用默认的__init__.py

否则必须指定到具体的模块



用户5639235042 @ 2020/5/16 00:23:35

请问，一个文件夹就是一个包吗？把py文件拖进去就是有拖进包了吗？或者我怎么创建包啊？



o0stanley0o @ 2020/6/9 15:06:30

一个目录+__init__.py 它就是包；没有__init__.py的就是个普通目录。



不长到140斤不改名字 @ 2022/6/5 04:54:48

顶上去，解决了我的困惑



不长到140斤不改名字 @ 2022/6/5 05:01:09

提个问题，原文提到：因为__init__.py 本身就是一个模块，而它的模块名就是 mycompany

```
mycompany
├── __init__.py
├── abc.py
└── xyz.py
```

问：为何__init__.py 的模块名不是 mycompany.init ?还是说 mycompany`mycompany.init 两个都可以？



SunTM&R @ 2022/11/3 11:09:25

试下就知道了，引入模块时 两个都可以mycompany.__init__



Stan @ 2025/2/26 10:32:32

看这句话：

请注意，每一个包目录下面都会有一个__init__.py的文件，这个文件是必须存在的，否则，Python就把这个目录当成普通目录，而不是一个包。__init__.py可以是空文件，也可以有Python代码，因为__init__.py本身就是一个模块，而它的模块名就是mycompany

所以mycompany.web这个模块对应的py文件就是web模块下面的__init__.py

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

12. 错误、调试和测试

13. IO编程

14. 进程和线程

15. 正则表达式

16. 常用内建模块

17. 常用第三方模块

18. 图形界面

19. 网络编程

20. 电子邮件

21. 访问数据库

22. Web开发

23. 异步IO

24. FAQ

25. 期末总结

📄 下载PDF



@ 2024/10/7 05:57:21

好难懂啊，mycompany.web也是一个模块，请指出该模块对应的.py文件，这个到底该怎么理解



👋 你我山巅自相逢 @ 2024/10/17 05:30:40

我猜是mycompany.web.init.py吧，因为上面说：init.py可以是空文件，也可以有Python代码，因为__init__.py本身就是一个模块，而它的模块名就是mycompany。



Reborn @ 2024/12/29 20:17:41

我试了一下，import mycompany.web时，引入的是该包中的init.py文件。



zy、 @ 2025/2/25 04:41:11

他既是包，又是模块。作为包，他下面有自己的几个py文件，分别就是包里的几个模块；作为模块，他下面的__init__.py就是他的模块对应文件。



萧湘雨 @ 2025/2/7 20:43:17

在 Python 中，import 语句用于导入模块或包。传统的 Python 包（也称为“显式命名空间包”）确实需要传统包（需要 __init__.py 文件）

在 Python 3.3 之前，一个目录要被识别为一个包，必须包含一个 __init__.py 文件。这个文件可以为空，示例：

假设你有以下目录结构：

复制

```
mypackage/  
  __init__.py  
  module1.py  
  module2.py
```

你可以通过以下方式导入模块：

Python复制

```
import mypackage.module1  
import mypackage.module2
```

隐式命名空间包（不需要 __init__.py 文件）

从 Python 3.3 开始，引入了 PEP 420，支持隐式命名空间包。这种包不需要 __init__.py 文件。Python

[Read More ▼](#)



jx @ 2024/11/11 08:11:49

打卡第四天



森 @ 2024/2/5 10:54:46

继续前进!!!



倒吃甘蔗 @ 2024/1/30 05:05:37

```
__init__.py
```



天天睡大觉 @ 2023/5/21 08:16:30

引入了包以后，只要顶层的包名不与别人冲突，那所有模块都不会与别人冲突。现在，abc.py 模块的名字就变成了 mycompany.abc，类似的，xyz.py 的模块名变成了 mycompany.xyz。



tae @ 2023/5/13 10:04:21

每一个包目录下面都会有一个 __init__.py 的文件，这个文件是必须存在的，否则，Python就把这个目录当成普通目录，而不是一个包。

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

12. 错误、调试和测试

13. IO编程

14. 进程和线程

15. 正则表达式

16. 常用内建模块

17. 常用第三方模块

18. 图形界面

19. 网络编程

20. 电子邮件

21. 访问数据库

22. Web开发

23. 异步IO

24. FAQ

25. 期末总结

📄 下载PDF



飞云 @ 2022/12/20 23:16:01

讲得很清楚，感谢



lovepruy @ 2022/11/11 03:25:29

到此一游



拖鞋 @ 2022/6/30 05:58:48

__init__.py



°Destiny @ 2022/6/20 03:49:59

mycompany.web 也是一个模块名，该模块名对应的.py文件是

__init__.py



Zack_Chang @ 2022/5/7 08:06:54

mycompany.web模块对应的是mycompany.web.__init__.py文件



Simms_zh @ 2022/3/3 20:47:11

mycompany.web 也是一个模块，请指出该模块对应的.py文件。

__init__.py



恁麻了批 @ 2022/1/24 01:09:15

```
mycompany
├── __init__.py
└── abc.py
```

每一个包目录下面都会有一个 `__init__.py` 的文件，这个文件是必须存在的，否则，Python就把这个目录当成普通目录，而不是一个包。`__init__.py` 可以是空文件，也可以有Python代码，因为 `__init__.py` 本身就是一个模块，而它的模块名就是 `mycompany`。

```
mycompany
├── web
│   ├── __init__.py
│   ├── utils.py
│   └── www.py
├── __init__.py
├── abc.py
└── utils.py
```

所以mycompany.web的py文件是 `__init__.py`



做回自己ZXH @ 2021/11/30 21:05:42

mycompany.web 也是一个模块名，该模块名对应的.py文件是

__init__.py