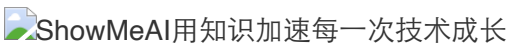


# 图解 Python 编程(21) | 模块

🕒 2021-11-09    👁 1362    💬 0    📁 工具教程 python 编程语言




作者：韩信子@**ShowMeAI**  
教程地址：<https://www.showmeai.tech/tutorials/56>  
本文地址：<https://www.showmeai.tech/article-detail/84>  
声明：版权所有，转载请联系平台与作者并注明出处  
收藏**ShowMeAI**查看更多精彩内容

## Python模块



在程序开发过程中，文件代码越来越长，维护越来越不容易。我们把很多不同的功能编写成函数，放到不同的文件里，方便管理和调用。在Python中，一个.py文件就称之为一个模块（Module）。

使用模块可以大大提高了代码的可维护性，而且当一个模块编写完毕，就可以被其他地方引用。我们在使用python完成很多复杂工作时，也经常引用其他第3方模块，受益于强大的python社区，几乎我们完成任何一项任务，都可以有对应的方便快捷可引用的库和模块来协助。




### Python模块 (Module)

<b>ourmodule.py</b> def square(x) print(x*x)	<b>yourmodule.py</b> def hello(x) print("Hello",x)
<b>code</b> import ourmodule ourmodule.square(5)	<b>code</b> import yourmodule yourmodule.hello("ShowMeAI")
<b>output</b> 25	<b>output</b> Hello ShowMeAI



🔍 搜索 | 微信 **ShowMeAI** 研究中心

 <http://www.showmeai.tech/>

模块是一个包含所有你定义的函数和变量的文件，以.py后缀结尾。模块可以被别的程序引入和使用其中的函数功能。

下面是一个使用 python 标准库中模块的例子。

```
1. import sys
2.
3. print('命令行参数如下:')
4. for i in sys.argv:
5.     print(i)
6.
7. print('\n
8. \
9. Python 路径为: ', sys.path, '\n
10. ')
```

执行结果如下所示：

```
1. $ python using_sys.py 参数1 参数2
2. 命令行参数如下:
3. using_sys.py
4. 参数1
5. 参数2
6.
7. Python 路径为: ['/root', '/usr/lib/python3.10', '/usr/lib/python3.10/plat-x86_64-linux-gnu',
'/usr/lib/python3.10/lib-dynload', '/usr/local/lib/python3.10/dist-packages', '/usr/lib/python3/dist-packages']
```

解释如下：

- **import sys** 引入 python 标准库中的 sys.py 模块；这是引入某一模块的方法。
- **sys.argv** 是一个包含命令行参数的列表。
- **sys.path** 包含了一个 Python 解释器自动查找所需模块的路径的列表。

## import语句

想使用Python模块，只需在另一个源文件里执行import语句，语法如下：

```
1. import module1[, module2[,... moduleN]
```

当解释器遇到 import 语句，如果在当前的搜索路径中能搜索到模块，就会直接导入。

搜索路径是一个解释器会先进行搜索的所有目录的列表。如想要导入模块 showmeai，需要把命令放在脚本的顶端：

**showmeai.py** 文件代码

```
1. def print_welcome( par ):
2.     print ("Welcome :", par)
3.     return
```

**test.py** 文件代码

```
1. # 导入模块
2. import showmeai
3.
4. # 现在可以调用模块里包含的函数了
5. showmeai.print_welcome("ShowMeAI")
```

以上代码输出结果：

```
1. $ python3 test.py
2. Welcome : ShowMeAI
```

当我们使用import语句的时候，Python解释器会在搜索路径中寻找对应模块，搜索路径是由一系列目录名组成的，它是在Python编译或安装的时候确定的，安装新的库应该也会修改。搜索路径被存储在sys模块中的path变量，做一个简单的实验，在交互式解释器中，输入以下代码：

```
1. >>> import sys
2. >>> sys.path
3. ['/', '/usr/lib/python3.10', '/usr/lib/python3.10/plat-x86_64-linux-gnu', '/usr/lib/python3.10/lib-
  dynload', '/usr/local/lib/python3.10/dist-packages', '/usr/lib/python3/dist-packages']
4. >>>
```

sys.path 输出是一个路径列表，其中第一项是空串”，执行python解释器的当前目录。

我们创建一个代码如下的fibo.py文件，把它放在sys.path中的任何一个目录里面：

```
1. def fib(n):  # 定义到 n 的斐波那契数列
2.     a, b = 0, 1
3.     while b < n:
4.         print(b, end=' ')
5.         a, b = b, a+b
6.     print()
7.
8. def fib_new(n): # 返回到 n 的斐波那契数列
9.     result = []
10.    a, b = 0, 1
11.    while b < n:
12.        result.append(b)
13.        a, b = b, a+b
14.    return result
```

然后进入Python解释器，使用下面的命令导入这个模块：

```
1. >>> import fibo
```

就可以使用模块名称来访问函数：

## 实例

```
1. >>>fibo.fib(1000)
2. 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
3. >>> fibo.fib_new(100)
4. [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
5. >>> fibo.__name__
6. 'fibo'
```

对于经常使用的函数，可以把它赋给一个本地的名称：

```
1. >>> my_fib = fibo.fib
2. >>> my_fib(500)
3. 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

## from ... import 语句

Python 的 from 语句让你从模块中导入一个指定的部分到当前命名空间中，语法如下：

```
1. from modname import name1[, name2[, ... nameN]]
```

例如，要导入模块 fibo 的 fib 函数，使用如下语句：

```
1. >>> from fibo import fib, fib_new
2. >>> fib(500)
3. 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

这个声明不会把整个fibo模块导入到当前的命名空间中，它只会将fibo里的fib函数引入进来。

## from ... import \* 语句

把一个模块的所有内容全都导入到当前的命名空间也是可行的，只需使用如下声明：

```
1. from modname import *
```

例如，要导入模块 fibo 的 所有 函数，使用如下语句：

```
1. >>> from fibo import *
2. >>> fib(500)
3. 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

## 标准模块

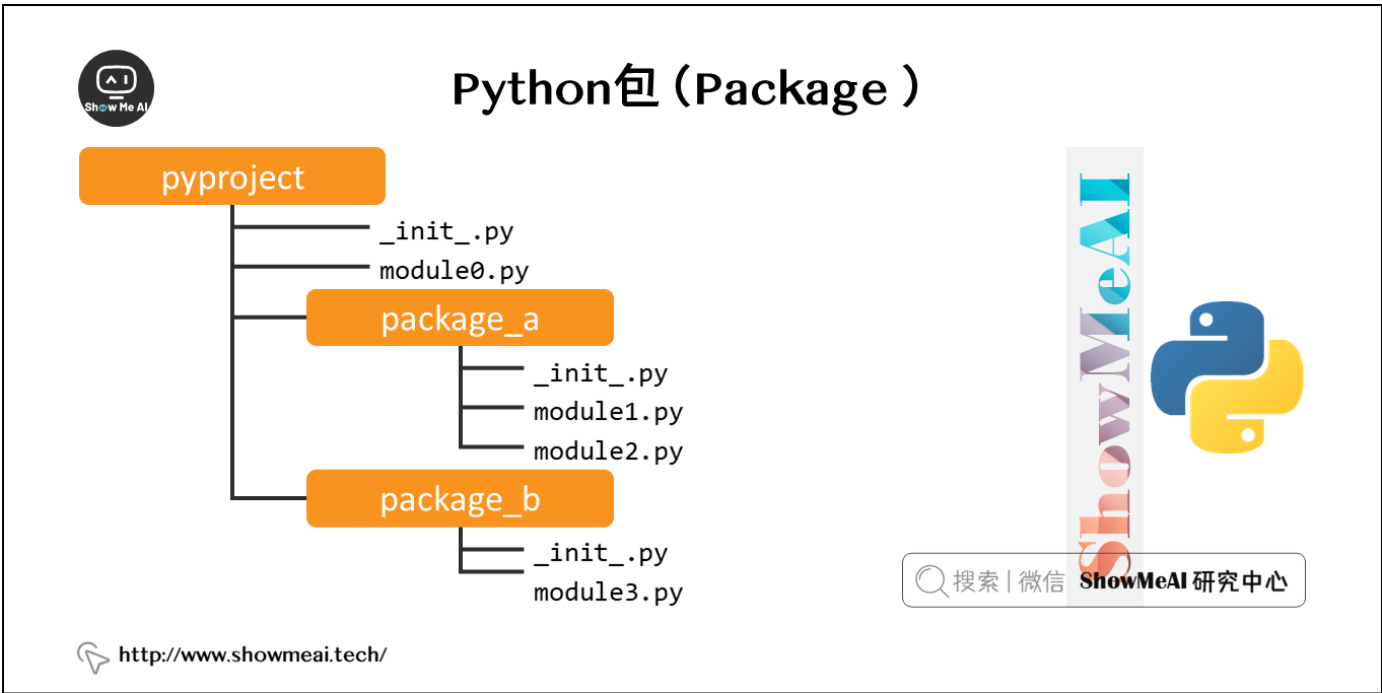
Python本身带着一些标准的模块库，有些模块直接被构建在解析器里，能很高效的使用。

比如模块sys，它内置在每一个Python解析器中。变量 sys.ps1 和 sys.ps2 定义了主提示符和副提示符所对应的字符串：

```
1. >>> import sys
2. >>> sys.ps1
3. '>>> '
4. >>> sys.ps2
5. '... '
6. >>> sys.ps1 = 'C> '
7. C> print('ShowMeAI!')
8. ShowMeAI!
9. C>
```

## 包

包是一种管理Python模块命名空间的形式，我们经常会以「包.模块」的形式来导入模块，例如一个模块的名称是C.D， 那么他表示一个包C中的子模块D。使用这种形式不用担心不同库之间的模块重名的情况。



假设你想设计一套统一处理视频文件 and 数据的模块（或者称之为一个”包”）。

现存很多种不同的音频文件格式（基本上都是通过后缀名区分的，例如：.mp4，.wmv，.avi，.mkv），所以需要有一组不断增加的模块，用来在不同的格式之间转换。

并且针对这些视频数据，还有很多不同的操作，所以你还需要一组庞大的模块来处理这些操作。

这里给出了一种可能的包结构（在分层的文件系统中）：

1. video/	顶层包
2.   __init__.py	初始化 video 包
3.   formats/	文件格式转换子包
4.       __init__.py	
5.       mp4read.py	
6.       mp4write.py	
7.       aviread.py	
8.       aviwrite.py	
9.       mkvread.py	
10.      mkvwrite.py	
11.      wmvread.py	
12.      wmvwrite.py	
13.      ...	
14.   audio/	声音效果子包
15.       __init__.py	
16.       io.py	
17.       fx.py	
18.       tools.py	
19.       ...	
20.   editor/	filters 子包
21.       __init__.py	
22.       period.py	
23.       faster.py	
24.       slower.py	
25.       ...	

在导入一个包的时候，Python 会根据 sys.path 中的目录来寻找这个包中包含的子目录。

目录只有包含一个叫做\_\_init\_\_.py 的文件才会被认作是一个包。最简单的处理是放一个空的\_\_init\_\_.py文件。

用户可以每次只导入一个包里面的特定模块，比如：

```
1. import video.audio.io
```

这将会导入子模块:video.audio.io。他必须使用全名去访问：

```
1. video.audio.io.readfile(input)
```

还有一种导入子模块的方法是：

```
1. from video.audio import io
```

这同样会导入子模块: io，并且他不需要那些冗长的前缀，所以他可以这样使用：

```
1. io.readfile(input)
```

还有一种变化就是直接导入一个函数或者变量：

```
1 from video.audio import readfile
```

```
from package import item
```

同样的，这种方法会导入子模块: io，并且可以直接使用他的 readfile() 函数:

1. readfile(input)

当使用 **from package import item** 这种形式的时候，对应的 item 既可以是包里面的子模块（子包），或者包里面定义的其他名称，比如函数，类或者变量。

import 语法会首先把 item 当作一个包定义的名称，如果没找到，再试图按照一个模块去导入。如果还没找到，抛出一个 **exc:ImportError** 异常。

如果我们使用形如 **import item.subitem.subsubitem** 这种导入形式，除了最后一项，都必须是包，而最后一项则可以是模块或者是包，但是不可以是类，函数或者变量的名字。

## 视频教程

也可以点击 [这里](#) 到B站查看有【中英字幕】的版本



## 一键运行所有代码

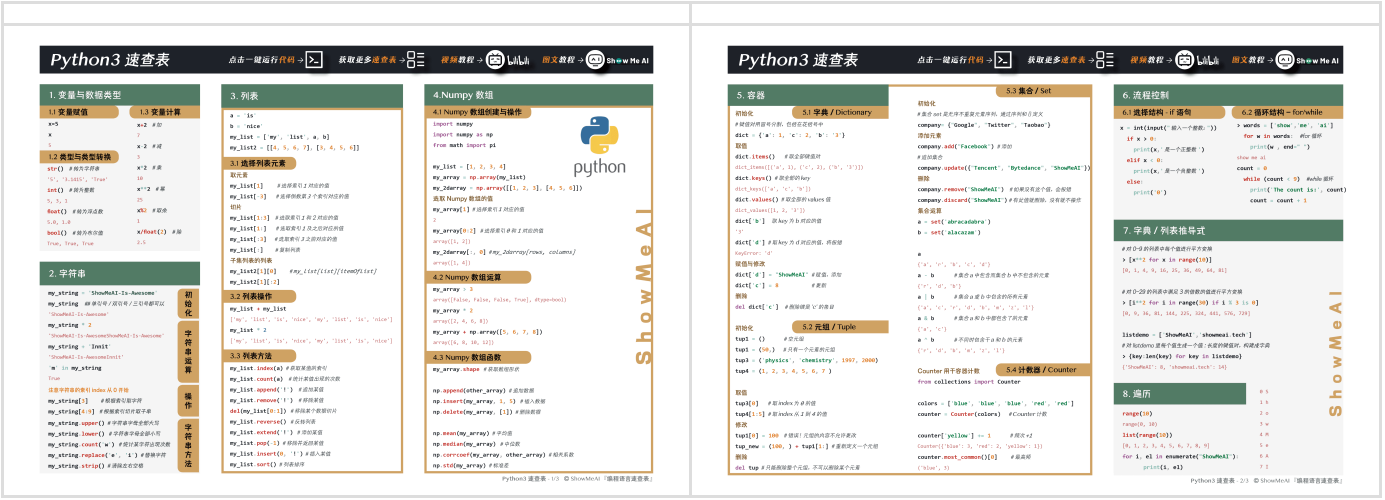
图解Python编程系列 配套的所有代码，可前往ShowMeAI 官方 **GitHub**，下载后即可在本地 Python 环境中运行。能访问 Google 的宝宝也可以直接借助 Google Colab一键运行与交互学习！

## 下载Python要点速查表

Awesome cheatsheets | **ShowMeAI速查表大全** 系列包含『编程语言』『AI技能知识』『数据科学工具库』『AI垂直领域工具库』四个板块，追平到工具库当前最新版本，并跑通了所有代码。点击 [官网](#) 或 [GitHub](#) 获取~

ShowMeAI速查表大全

### Python 速查表（部分）



## 拓展参考资料

- Python教程 - Python3文档
- Python教程 - 廖雪峰的官方网站

## ShowMeAI图解Python编程系列推荐（要点速查版）

- ShowMeAI 图解 Python 编程(1) | 介绍
- ShowMeAI 图解 Python 编程(2) | 安装与环境配置
- ShowMeAI 图解 Python 编程(3) | 基础语法
- ShowMeAI 图解 Python 编程(4) | 基础数据类型
- ShowMeAI 图解 Python 编程(5) | 运算符
- ShowMeAI 图解 Python 编程(6) | 条件控制与if语句
- ShowMeAI 图解 Python 编程(7) | 循环语句
- ShowMeAI 图解 Python 编程(8) | while循环
- ShowMeAI 图解 Python 编程(9) | for循环
- ShowMeAI 图解 Python 编程(10) | break语句
- ShowMeAI 图解 Python 编程(11) | continue语句
- ShowMeAI 图解 Python 编程(12) | pass语句
- ShowMeAI 图解 Python 编程(13) | 字符串及操作
- ShowMeAI 图解 Python 编程(14) | 列表
- ShowMeAI 图解 Python 编程(15) | 元组
- ShowMeAI 图解 Python 编程(16) | 字典
- ShowMeAI 图解 Python 编程(17) | 集合
- ShowMeAI 图解 Python 编程(18) | 函数



- [ShowMeAI 图解 Python 编程\(19\) | 迭代器与生成器](#)
- [ShowMeAI 图解 Python 编程\(20\) | 数据结构](#)
- [ShowMeAI 图解 Python 编程\(21\) | 模块](#)
- [ShowMeAI 图解 Python 编程\(22\) | 文件读写](#)
- [ShowMeAI 图解 Python 编程\(23\) | 文件与目录操作](#)
- [ShowMeAI 图解 Python 编程\(24\) | 错误与异常处理](#)
- [ShowMeAI 图解 Python 编程\(25\) | 面向对象编程](#)
- [ShowMeAI 图解 Python 编程\(26\) | 命名空间与作用域](#)
- [ShowMeAI 图解 Python 编程\(27\) | 时间和日期](#)

## ShowMeAI系列教程精选推荐

- [大厂技术实现：推荐与广告计算解决方案](#)
- [大厂技术实现：计算机视觉解决方案](#)
- [大厂技术实现：自然语言处理行业解决方案](#)
- [图解Python编程：从入门到精通系列教程](#)
- [图解数据分析：从入门到精通系列教程](#)
- [图解AI数学基础：从入门到精通系列教程](#)
- [图解大数据技术：从入门到精通系列教程](#)
- [图解机器学习算法：从入门到精通系列教程](#)
- [机器学习实战：手把手教你玩转机器学习系列](#)
- [深度学习教程：吴恩达专项课程·全套笔记解读](#)
- [自然语言处理教程：斯坦福CS224n课程·课程带学与全套笔记解读](#)
- [深度学习与计算机视觉教程：斯坦福CS231n·全套笔记解读](#)

## 图解 Python 编程(21) | 模块

[< 上一篇](#)

[图解 Python 编程\(20\) | 数据结构](#)

[下一篇 >](#)

[图解 Python 编程\(22\) | 文件读写](#)