

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？ [最近阅读](#)

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

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

04 如何运行 Python 代码？

更新时间：2019-11-26 09:47:23



“人的影响短暂而微弱，书的影响则广泛而深远。”
——普希金

上节课我带着大家来让 Python 在我们的电脑上安家落户。既然安装成功了，那么使用起来也不是问题。运行 Python 代码有两种方式。一种是直接在 Python 解释器中的输入代码，然后就地执行它（也就是交互模式）。另一种是把 Python 代码保存到文件中，之后去执行这个文件。

交互模式执行 Python 代码

先来看下如何在交互模式下执行 Python 代码。首先进入 Python 解释器交互模式，进入方法是：

- 使用 Linux 和 MacOS 的读者在命令行（虚拟终端）中执行命令

```
python3
```

- 使用 Windows 的读者在命令行（CMD）中执行命令

```
python
```

之后你会看一些版本和帮助信息：

```
→ ~ python3 Python 3.7.3 (default, Mar 27 2019, 09:23:15)
[Clang 10.0.1 (clang-1001.0.46.3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

← 慕课专栏

你的第一本Python基础入门书 / 04 如何运行 Python 代码？

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？

最近阅读

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

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

等待你的输入。试着输入一些内容，然后按回车键去执行它：

12345 + 54321

>>> 12345 + 54321
66666

'apple' + 'pen'

>>> 'apple' + 'pen'
'applepen'

可以看到，按下回车键后解释器会立即执行刚才输入的代码，并将直接将执行结果输出出来。

执行 Python 文件

再来看下如何执行 Python 文件。所谓 Python 文件，其实就是保存 Python 代码的文件，通常将其文件后缀名约定为 `.py`。

扩展：其实用别的后缀名甚至不用后缀名都是可以的，但是既然是约定，那么大家就该遵守。`.py` 后缀能很清晰的表明这是 Python 文件。

我们来创建一个文件，如 `today.py`，然后把以下代码拷贝进去：

import datetime
today = datetime.date.today().strftime('%Y{}%m{}%d{}').format('*'年月日')
print('今天是：' + today)

然后通过 Python 命令来执行这个 `today.py` 文件：

• Windows 下执行：

python today.py

C:> python today.py
今天是：2019年07月03日

• Linux 和 MacOS 下执行：

python3 today.py

www.imooc.com/read/46/article/811

2/6

慕课专栏

你的第一本Python基础入门书 / 04 如何运行 Python 代码？

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？最近阅读

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

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

今天是：2019年07月03日

注意：上述文件使用的是省略了路径前缀的相对路径形式，执行命令前须先进入到 Python 文件所在目录（即确保工作目录和文件所在目录一致），否则解释器会找不到该文件。也可以使用绝对路径的形式，如 `python C:\Users\happy\today.py`，此时无需考虑工作目录。

交互模式执行和 Python 文件执行的差别

显而易见，Python 代码的交互模式执行和文件执行在使用形式上是不同的。除此之外，在输出结果的方式上这两种方法也存在一些不同。

比如代码：

12345 + 54321

在交互模式下执行时，计算结果会被直接输出出来：

>>> 12345 + 54321
66666

在这里我们没有使用任何输出操作，代码执行时计算出结果并自动输出。

而如果把这行代码写到文件里（如 `add.py`），然后使用文件执行的方式来执行它，这时命令行中没有任何内容被输出显示。如下：

→ ~ python3 add.py
→ ~

这是因为文件执行时，每个语句的执行结果并不会被自动输出，除非代码中有明确的输出操作。

这输出操作具体是什么呢？比如 Python 里有个叫 `print()` 函数的东西，它可以把结果输出显示到命令行中。使用时只需将想要输出的内容放在 `print()` 的括号中，像这样：

print(12345 + 54321)

好了，我们修改下文件中的代码，用 `print(12345 + 54321)` 完整替换原来的 `12345 + 54321`，再次执行下看看：

慕课专栏

你的第一本Python基础入门书 / 04 如何运行 Python 代码？

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？最近阅读

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

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

66666

这样结果就显示出来了。交互模式下当然也可以使用 `print()`，效果也是一样的。

注意：代码中的所有符号，包括括号「`()`」、逗号「`,`」、分号「`;`」等必须使用英文半角符号，否则程序会出错。

扩展：`print()` 是 Python 的内置函数（函数概念后续会提到），无需从包（程序库）中导入。准确来讲，它的行为是将内容写到标准输出中。

交互模式 or Python 文件？

在做 Python 开发时，大多数时候都是用执行 Python 文件的方式来运行代码。交互模式往往只是在试验小段代码，或者用 Python 做一些简单事情的时候才会去使用。原因是：

• 大多数场景下，我们的程序需要以自动化的方式来运行，而交互模式顾名思义是需要人的参与的，不便于自动化

• 把代码写到文件里方便反复执行

• 交互模式下的输入体验不佳，编写大段代码时不够便利

趁手的代码编辑器

执行 Python 文件是运行 Python 代码的主要方式，正因为如此，我们需要一个好用的编辑器来编写 Python 文件，毕竟系统自带的文本编辑器太简陋了。

用来写代码的编辑器实在很多，比如老牌的 VIM、Emacs，之后的 Atom、Sublime，以及新秀 VS code。除此之外，还可以用重量级的 IDE（Integrated Development Environment——集成开发环境）。

扩展：编辑器通常有代码编辑、高亮显示、基于历史记录的代码补全等功能。IDE 在编辑器的基础上还支持基于静态分析的代码补全，引用查看，代码跳转，重构，代码格式化，编辑时错误提示，风格提示等强大且常用的功能。

编辑器能通过插件来扩展得到这些功能，但需要经历一番折腾。想省些力气的读者可以直接用 IDE 来做开发。

这里分别推荐一款代码编辑和一款 IDE：

• VS code，微软开源的优秀代码编辑器。官方下载页面

← 慕课专栏

☰ 你的第一本Python基础入门书 / 04 如何运行 Python 代码？

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？

最近阅读

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

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

Redefined.

Free. Built on open source. Runs everywhere.

Download for Mac

Stable Build

Other platforms and Insiders Edition

By using VS Code, you agree to its license and privacy statement.

VS Code Extensions

CF for Visual Studio Code (b...
Python 3.8...
Debugger for Chrome
C/C++
Go
Rich Go language support for...
ESLint
PowerShell
Develop PowerShell scripts in...

const debug = debugModule('node-express-typescript-server');
// Get port from environment and store in Express.
const port = normalizePort(process.env.PORT || '3000');
app.set('port', port);
// Create Express app
const server = express();
server.set('trust proxy', 1);
server.set('view engine', 'pug');
server.set('views', path.resolve(__dirname, 'views'));
server.use(bodyParser.json());
server.use(bodyParser.urlencoded({ extended: true }));
server.use(cookieParser());
server.use(session({
 secret: 'keyboard cat',
 resave: false,
 saveUninitialized: true,
 cookie: { expires: 60 * 60 * 24 * 7 }
}));
// Define routes
server.get('/', function(req, res) {
 res.render('index');
});
server.get('/about', function(req, res) {
 res.render('about');
});
// Start the server
server.listen(port, function() {
 console.log(`Server is running on port \${port}`);
});
// Graceful shutdown
process.on('SIGINT', function() {
 console.log('Received SIGINT, shutting down gracefully...');
 server.close(function() {
 process.exit(0);
 });
});

PC

Version: 2019.1.3
Build: 191.7479.30
Released: May 30, 2019

System requirements
Installation Instructions
Previous versions

Download PyCharm

Windows macOS Linux

Professional
For both Scientific and Web Python development. With HTML, JS, and SQL support.
DOWNLOAD
Free trial

Community
For pure Python development
DOWNLOAD
Free, open-source

本专栏代码演示说明

值得说明的是，后续文章中的代码执行演示以交互模式为主。交互模式的典型标记是 `>>>` 提示符，如下示例就是在解释器交互模式下进行的：

>>> 33+725
758

另外在代码较长时，本专栏也会使用执行 Python 文件的方式来作代码执行演示。此时的典型标记是 `→ ~`，表明这是在命令行（即 Windows CMD 或 Linux / MacOS 虚拟终端）中执行命令。如下：

→ ~ python3 hello.py
hello

说明：上述使用的是 Linux / MacOS 下的 `python3` 命令，使用 Windows 的读者请自行替换为 `python`。

这是本专栏的代码演示形式。读者自己试验代码时，是选择交互模式还是 Python 文件的方式完全由个人喜好而定。如果使用 Python 文件的方式，请不要忘记想要输出执行结果需要在代码里

www.imooc.com/read/46/article/811

5/6

<div>← 慕课专栏</div>		<div>≡ 你的第一本Python基础入门书 / 04 如何运行 Python 代码?</div>	
<div>目录</div>		<div>← 03 让 Python 在你的电脑上安家落户</div>	<div>05 数据的名字和种类—变量和类型 →</div>
<div>第 1 章 入门准备</div>		<div>精选留言 2</div>	
<div>01 开篇词：你为什么要学 Python ?</div>		<div>欢迎在这里发表留言，作者筛选后可公开显示</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>最近阅读</div></div>		<div><div>weixin_慕斯卡9537365</div><div>今天加入学习，之前有学过基础知识；但对类的使用还不太熟悉，少有练习。同样期望能快速更新，特别想看对应章节和练习。谢谢老师。☺☺。也请推荐一些学习书籍或网友</div><div><div>👍 1</div><div>回复</div><div>2019-08-23</div></div></div>	
<div>第 2 章 通用语言特性</div>		<div><div>黄浮云 回复 weixin_慕斯卡9537365</div><div>第2章节中会有类的介绍，第3章节中会更深入地来探讨类，不过专栏还需要按照预定进度来更新。如果你着急想学习类方面的内容的话，不妨先看看 Python 官方文档的类章节，中译版：https://docs.python.org/zh-cn/3/tutorial/classes.html；英文原版：https://docs.python.org/3/tutorial/classes.html。至于网友，不如你加我微信好了 flocloud 😊</div><div><div>回复</div><div>2019-08-23 22:26:11</div></div><div><div>weixin_慕斯卡9537365 回复 weixin_慕斯卡9537365</div><div>谢谢黄老师😊</div><div><div>回复</div><div>2019-08-24 09:39:08</div></div></div></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</div>			
<div>12 练习—密码生成器</div>		<div><div>橘彩星光</div><div>老师能不能一天多更新几节</div><div><div>👍 1</div><div>回复</div><div>2019-08-21</div></div></div>	
<div>第 3 章 Python 进阶语言特性</div>		<div><div>黄浮云 回复 橘彩星光</div><div>不好意思哈，这个专栏是按照每周三篇的进度来更新的，不过从第2章开始每篇文章的知识密度会渐渐大起来。</div><div><div>回复</div><div>2019-08-23 22:11:30</div></div></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>千学不如一看，千看不如一练</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>			