

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

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

## 26 更加 Python 的 Python 代码风格

更新时间：2019-10-25 10:10:01



“

你若要喜爱你自己的价值，你就得给世界创造价值。

”

——歌德

### 代码风格是什么

代码编写好后交由计算机执行，计算机不关心代码文本的布局、各类对象的命名，只要合乎语法和语义，代码就可按预期被执行。

围绕着代码，人（开发者）的作用看起来似乎只是单纯地输出代码。其实不然，开发者不仅输出代码，也需要从代码中获取信息，需要经常性地阅读代码。

阅读代码的过程可能发生在开发调试、维护修改 bug 时，毕竟开发者的记忆是有限的，自己写过的代码可能在不久后便会有遗忘，日后需要反复回顾以确认细节。另一个阅读代码的典型情形是，开发过程中往往不是一个人在写代码，通常需要进行团队协作，这时我们可能需要阅读别人的代码以便熟悉对方的代码逻辑。

总而言之，代码不仅仅是被用来执行的，也会被开发者经常性地阅读，无论这些代码是自己所写还是别人所写。

良好的、统一的代码风格会给阅读带来非常大的便利，能尽可能地降低代码阅读障碍，提升阅读效率。同时对于编码者来说，良好的、统一的代码风格能让人有一种固定的模式可依循，以便写出高可读性的代码。

我们可以在公司或部门这个层面制定 Python 代码风格，将其用于各个项目，每个开发者遵从这些代码风格来编写代码。

当然我们也有更省事、更通用的方法，使用 Python 社区所制定的 [Python 代码风格](#)（PEP 8）。这其中非常详细地列举出了应该遵守的代码风格，在这里我们只选取其中重要的、常用的

<div>← 慕课专栏</div>	<div>☰ 你的第一本Python基础入门书 / 26 更加 Python 的 Python 代码风格</div>
目录	命名
第 1 章 入门准备	Python 中的命名主要有三种风格：
01 开篇词：你为什么要学 Python？	1. 全小写+下划线：所有字母均使用小写形式，多个单词时用下划线（ _ ）分隔，如 <code>response</code> 、 <code>message_body</code> 、 <code>seconds_per_hour</code>
02 我会怎样带你学 Python？	2. 全大写+下划线：所有字母均使用大写形式，多个单词时用下划线（ _ ）分隔，如 <code>DATE TIME</code> 、 <code>MAX_SEGMENT_LIFETIME</code>
03 让 Python 在你的电脑上安家落户	3. 驼峰写法：一个或多个单词，每个单词的首字母大写，如 <code>String</code> 、 <code>CurrentTime</code>
04 如何运行 Python 代码？	每种风格有不同的使用场景，不可随意使用，具体如下。
第 2 章 通用语言特性	变量和函数
05 数据的名字和种类—变量和类型	变量和函数使用「全小写+下划线」的命名方式：
06 一串数据怎么存—列表和字符串	<code>seconds_per_day</code>
07 不只有一条路—分支和循环	<code>print_hello()</code>
08 将代码放进盒子—函数	类名
09 知错能改—错误处理、异常机制	类的命名采用「驼峰写法」：
10 定制一个模子—类	<code>MyClass</code>
11 更大的代码盒子—模块和包	异常名
12 练习—密码生成器	异常也是类，所以也使用「驼峰写法」，并且以「Error」结尾：
第 3 章 Python 进阶语言特性	<code>FileParseError</code>
13 这么多的数据结构（一）：列表、元祖、字符串	常量
14 这么多的数据结构（二）：字典、集合	常量使用「全大写+下划线」命名：
15 Python大法初体验：内置函数	<code>MAX_RETRY_TIMES</code>
16 深入理解下迭代器和生成器	模块名和包名
17 生成器表达式和列表生成式	包名和模块名应该短小。
18 把盒子升级为豪宅：函数进阶	模块可使用「小写 + 下划线」的方式命名：
19 让你的模子更好用：类进阶	<code>open_api</code>
20 从小独栋升级为别墅区：函数式编程	包名仅使用小写字母命名，其中不建议使用下划线：
	<code>requests</code>

← 慕课专栏	:三 你的第一本Python基础入门书 / 26 更加 Python 的 Python 代码风格
目录	缩进
第 1 章 入门准备	Python 代码格外注重缩进，强制用缩进来区分代码块，以及子代码块。
01 开篇词：你为什么要学 Python ？	每级缩进应使用 4 个空格。
02 我会怎样带你学 Python ？	<pre>def func():     for i in range(x):         if condition :             pass</pre>
03 让 Python 在你的电脑上安家落户	有时我们习惯于使用 Tab 键来缩进代码，这时可在编辑器或 IDE 中将 Tab 键的效果设置为 4 个空格。
04 如何运行 Python 代码？	换行
第 2 章 通用语言特性	每行代码的最大字符数为 79。若某一行代码过长，可以将其换行书写。
05 数据的名字和种类—变量和类型	定义函数时，将参数换行后缩进两次，以和代码块相区分。
06 一串数据怎么存—列表和字符串	<pre># 建议： def long_function_name(     var_one, var_two, var_three,     var_four):     print(var_one)  # 不建议： def long_function_name(     var_one, var_two, var_three,     var_four):     print(var_one)</pre>
07 不只有一条路—分支和循环	函数调用时，若左括号后紧跟参数，后续行与左括号对齐。
08 将代码放进盒子—函数	<pre># 建议： foo = long_function_name(var_one, var_two,                            var_three, var_four)  # 不建议： foo = long_function_name(var_one, var_two,                            var_three, var_four)</pre>
09 知错能改—错误处理、异常机制	函数调用时，若左括号后没有参数，后续行可任意缩进，无需与左括号对齐。
10 定制一个模子—类	<pre># 建议： foo = long_function_name(     var_one, var_two,     var_three, var_four)</pre>
11 更大的代码盒子—模块和包	多行结构中（列表、元组、集合、函数参数列表），其中元素及右括号可换行书写。
12 练习—密码生成器	<pre># 右括号可与元素对齐 my_list = [     1, 2, 3,</pre>
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	

<div>← 慕课专栏</div> <div>≡ 你的第一本Python基础入门书 / 26 更加 Python 的 Python 代码风格</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 ？	<pre>result = some_function_that_takes_arguments(     'a', 'b', 'c',     'd', 'e', 'f', )</pre> <pre># 右括号也可顶格书写 my_list = [     1, 2, 3,     4, 5, 6, ]</pre> <pre>result = some_function_that_takes_arguments(     'a', 'b', 'c',     'd', 'e', 'f', )</pre>
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	<pre>with open('/path/to/some/file/you/want/to/read') as file_1, \n    open('/path/to/some/file/being/written', 'w') as file_2:     file_2.write(file_1.read())</pre>
08 将代码放进盒子—函数	<p>定义函数和类时，多个函数或类之间使用两个空行进行分隔：</p> <pre>def foo1():     pass  def foo2():     pass  class MyClass1:     pass  class MyClass2:     pass</pre>
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	

Python 中小括号、中括号、大括号中的对象可以直接被换行，而无需添加其它符号（这叫隐式换行），如上。但其它情况下，代码换行书写，需要在每行末尾添加反斜杠（`\`）。

类中的定义方法时，多个方法间用一个空行进行分隔。

## 导入

使用 `import` 时，`import` 语句应位于文件顶部，模块注释和文档字符串之后。

`import` 按下列类型和顺序使用：

1. 标准库导入
2. 第三方库导入
3. 本地库导入

以上分为三个分组，每个分组间用一个空行分隔。

## 注释

← 慕课专栏	☰ 你的第一本Python基础入门书 / 26 更加 Python 的 Python 代码风格
目录	3. 注释与代码矛盾比没有注释更糟，修改代码前首先修改注释
	4. 注释应该是完整的句子或短句，并且第一个单词首字母大写
第 1 章 入门准备	完整代码风格内容
01 开篇词：你为什么要学 Python ？	以上是常用的代码风格说明，关于 Python 完整的代码风格指导，可以阅读 <a href="#">PEP 8 – Style Guide for Python Code</a> （英文原版），也可以自行搜索「PEP 8 翻译」。建议大家仔细阅读。
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 ？	← 25 Python的影分身之术：虚拟环境 27 这么多条路，你想走那一条？ →
第 2 章 通用语言特性	精选留言 0
05 数据的名字和种类—变量和类型	欢迎在这里发表留言，作者筛选后可公开显示
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	