EFFECTIVE 系列丛书

HZ BOOKS 华章IT

PEARSON

Google高级软件工程师Brett Slatkin融合自己多年Python开发实战经验，深入探讨编写高质量Python代码的技巧、禁忌和最佳实践

涵盖Python 3.x和Python 2.x主要应用领域，汇聚59条优秀实践原则、开发技巧和便捷方案，包含大量实用范例代码

Effective Python

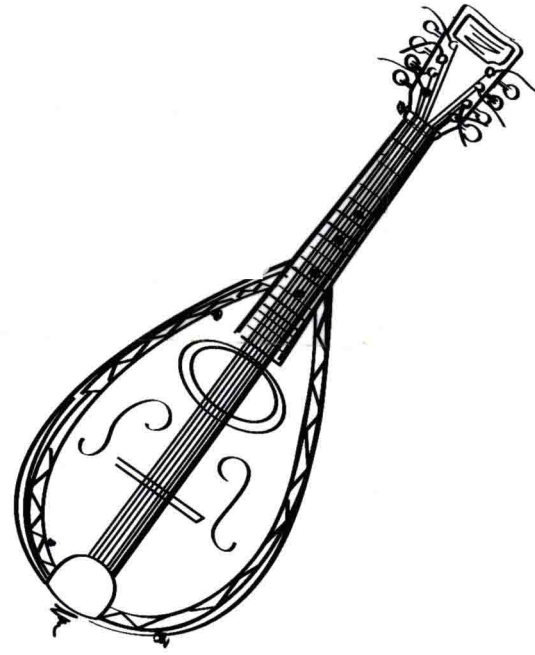
59 Specific Ways to Write Better Python

Effective Python

编写高质量Python代码的59个有效方法

[美] 布雷特·斯拉特金（Brett Slatkin）著

爱飞翔 译



机械工业出版社
China Machine Press

EFFECTIVE
系列丛书

Effective Python

59 Specific Ways to Write Better Python

Effective Python

编写高质量Python代码的59个有效方法

[美] 布雷特·斯拉特金（Brett Slatkin）著

爱飞翔 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Effective Python：编写高质量 Python 代码的 59 个有效方法 /（美）斯拉特金（Slatkin，B.）著；爱飞翔译 . —北京：机械工业出版社，2016.1
（Effective 系列丛书）
书名原文：Effective Python：59 Specific Ways to Write Better Python

ISBN 978-7-111-52355-0

I. E… II. ①斯… ②爱… III. 软件工具－程序设计 IV. TP311.56

中国版本图书馆 CIP 数据核字（2015）第 305873 号

本书版权登记号：图字：01-2015-2812

Effective Python
编写高质量 Python 代码的 59 个有效方法

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：关　敏　　责任校对：董纪丽
印　　刷：三河市宏图印务有限公司　　版　　次：2016 年 1 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：14
书　　号：ISBN 978-7-111-52355-0　　定　　价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

Praise 本书赞誉

“Slatkin 所写的这本书，其每个条目（item）都是一项独立的教程，并包含它自己的源代码。这种编排方式使我们可以随意跳读：大家可以按照学习的需要来浏览这些条目。本书涉及的话题十分广泛，作者针对这些话题，给出了相当精练而又符合主流观点的建议，我把这本书推荐给中级 Python 程序员。”

——Brandon Rhodes，Dropbox 的软件工程师、2016 ~ 2017 年 PyCon 会议的主席

“我用 Python 写了很多年程序，自认为对 python 已经了解得很透彻了。但在看过这本讲解诀窍和技巧的好书之后我才发现，其实我还能把 Python 代码写得更高效（例如，使用内置的数据结构）、更易读（例如，设定只允许通过关键字形式来指定的参数），以令其更加符合 Python 风格（例如，用 zip 函数来同时迭代两个列表）。”

——Pamela Fox，Khan 学院的教师

“当初我刚从 Java 转向 Python 时，要是能先看到这本书的话，那就能节省好几个月的时间。这本书使我意识到：以前反复编写的那些代码，都不是很符合 Python 的编程风格。这本书包含了 Python 语言的绝大部分必备知识，使我们无需通过数月乃至数年的艰难探索，即可逐个了解它们。本书的内容非常丰富，从 PEP8 的重要性和 Python 语言的主要编程习惯开始，然后谈到如何设计函数、方法和类，如何高效地使用标准库，以及如何设计高质量的 API，最后，又讲了测试及性能问题。新手和老手都可以通过这本优秀教程来领略 Python 编程的真谛。”

——Mike Bayer，SQLAlchemy 的创立者

“这本书会清楚地告诉你如何改善 Python 代码的风格及函数的质量，它会令你的 Python 技能更上一层楼。”

——Leah Culver，Dropbox 的开发者代言人（developer advocate）

“这是一本极好的书，对其他编程语言较有经验的开发者，可以通过本书迅速学习 Python，并了解更符合 Python 风格的基础语言结构。本书内容清晰、简明，而且易于理解，只需阅读某个条目或某一章，即可单独研究某个话题。书中讲解了大量纯 Python 的语言结构，使读者不会把它们与 Python 生态圈中的其他复杂事物相混淆。经验更多的开发者可以通过书中提供的一些深度范例来了解自己尚未遇到的语言特性，以及原来不常使用的语言功能。作者肯定是一位非常熟悉 Python 的人，他用自己丰富的经验来给读者指出各种经常出现的 bug 以及经常出错的写法。另外，本书也恰当地说明了 Python 2.X 与 Python 3.X 之间的微妙区别，大家在各种版本的 Python 之间迁移时，可以把本书用作参考资料。”

——Katherine Scott，Tempo Automation 的软件主管

“这是一本对初级开发者和熟练开发者都适用的好书。代码范例及其讲解都写得非常细致、非常简洁、非常透彻。”

——C. Titus Brown，加州大学戴维斯分校副教授

“这本参考书非常有用，它提供了很多高级的 Python 用法，并讲解了如何构建更清晰、更易维护的软件。把书中的建议付诸实践，就可以令自己的 Python 技能得到提升。”

——Wes McKinney，pandas 程序库的创立者《Python for Data Analysis》[⊖]的作者、Cloudera 的软件工程师

⊖ 中文版为《利用 Python 进行数据分析》，已由机械工业出版社引进出版。——编辑注

The Translator's Words 译 者 序

自 Scott Meyers 撰写的《Effective C++》问世以来，出现了很多以 Effective 命名的技术书籍。Effective 一词，并不单单局限于执行速度层面的高效率，同时有着令代码易于阅读、易于测试且易于维护等意思，此外，它还蕴涵着易于扩展、易于修改和易于多人协作等更为高阶的理念。

因此，从上述宏观层面来看，Effective 式的心得手册，无论是对初学者还是熟练者，都有较大意义。本书自然也不例外。对于初学者来说，书中展示了该语言的大体轮廓，使我们能够知道 Python 的强项和弱项。在知道了这些特性之后，开发者就可以结合自己的兴趣与需求，有选择、有顺序地学习。

而对于熟练者来说，则可以把书中的心得与自己的经验相比对，看看自己还有哪些区域尚未深入研究，同时思考一下书中的方案与自己常用的方案各有什么优点与缺点。

从本书各条技巧的具体编排方式来看，本书既可以像字典那样查阅，也可以像普通图书那样通读。很多条目都是用渐进的方式来编写的。作者不会在一开始就给出最佳方案，而是会先从简单的写法入手，逐步发现其缺点并加以完善，最后总结出一套便于使用且易于扩充的解决办法。

这样的演进方式既适用于本书所列的各个场景，也适用于日常的编程工作。如果能通过这些具体的条目来培养一套分析并解决问题的思路，那就可以更加深刻地体会 Python 语言的设计哲学及实践艺术。很多 Python 开发者都崇尚 Pythonic 编程方式，这种 Pythonic 方式不仅应该体现在代码风格和项目规范之中，而且更应该体现在思维模式和架构设计层面，这一点，我想应该是 Effective 系列的书籍值得反复品味的缘由吧。

虽说这 59 条技巧并不能涵盖所有的 Python 领域，但在经常接触的那几个主要领域中，它们却是相当有代表性和启发性的。我们可以把这些技巧以自己的方式实现出

来，并封装成模块及软件包，以便在后续的工作中使用。而对于本书没有专门涉及的领域，如游戏开发、图形绘制、网络通信等，大家不妨也沿用 Effective 书系的一贯做法，把自己的经验总结成条目，进而以博客或开源项目的形式互相交流。这可以说是对 Effective 理念的一种延伸和发展。

由于许多 Python 开发者都同时具备 C++ 及 Java 等其他语言的开发背景，所以 Python 中的很多概念都有好几种不同的称呼方式，而这些术语的中文翻译，自然也就呈现出了一词多译的现象。本书将尽量采用较为折中的办法来处理这些问题。

本书的翻译过程中，得到了机械工业出版社华章公司诸位编辑和工作人员的帮助，在此深表谢意。

由于译者水平有限，不足与疏漏之处，请大家发邮件至 eastarstormlee@gmail.com，或访问 github.com/jeffreybaoshenlee/zh-translation-errata-effective-python/issues 留言，给我以批评和指教。该网页还有《中英文词汇对照表》，以供参考。

Preface 前　言

Python 编程语言很强大、很有魅力，但同时也很独特，所以掌握起来比较困难。许多程序员从他们所熟悉的语言转入 Python 之后，没能把思路打开，以致写出的代码无法完全发挥出 Python 的特性，而另外一些程序员则相反，他们滥用 Python 的特性，导致程序可能在将来出现严重问题。

本书会深入讲解如何以符合 Python 风格的（Pythonic）方式来编写程序，这种方式就是运用 Python 语言的最佳方式。笔者假定你对这门语言已经有了初步了解。编程新手可以通过本书学到各种 Python 功能的最佳用法，而编程老手则能够学会如何自信地运用一种功能强大的新工具。

笔者的目标是令大家学会用 Python 来开发优秀的软件。

本书涵盖的内容

本书每一章都包含许多互相关联的条目，大家可以按照自己的需要，随意阅读这些条目。每个条目都包含简明而具体的教程，告诉你应该如何更高效地编写 Python 程序。笔者在每个条目里面都给出了建议，告诉大家应该怎样做、应该避免哪些用法，以及如何在各种做法之间求得平衡，并解释了笔者所选的做法好在哪里。

本书中的各项条目，适用于 Python 3 和 Python 2（请参阅本书第 1 条）。对于 Jython、IronPython 或 PyPy 等其他运行时环境，大部分条目应该同样适用。

第 1 章：用 Pythonic 方式来思考

Python 开发者用 Pythonic 这个形容词来描述具有特定风格的代码。这种风格是大家在使用 Python 语言进行编程并相互协作的过程中逐渐形成的习惯。本章讲解如何以

该风格来完成常见的 Python 编程工作。

第 2 章：函数

Python 中的函数具备多种特性，这可以简化编程工作。Python 函数的某些性质与其他编程语言中的函数相似，但也有些性质是 Python 独有的。本章介绍如何用函数来表达意图、提升可复用程度，并减少 bug。

第 3 章：类与继承

Python 是面向对象的语言。用 Python 编程时，通常需要编写新类，并定义这些类应该如何通过其接口及继承体系与外界相交互。本章讲解如何使用类和继承来表达对象所应具备的行为。

第 4 章：元类及属性

元类（metaclass）及动态属性（dynamic attribute）都是很强大的 Python 特性，然而它们也可能导致极其古怪、极其突然的行为。本章讲解这些机制的常见用法，以确保读者写出来的代码符合最小惊讶原则（rule of least surprise）。

第 5 章：并发及并行

用 Python 很容易就能写出并发程序，这种程序可以在同一时间做许多件不同的事情。我们也可以通过系统调用、子进程（subprocess）及 C 语言扩展来实现并行处理。本章讲解如何在不同情况下充分利用这些 Python 特性。

第 6 章：内置模块

Python 预装了许多写程序时会用到的重要模块。这些标准软件包与通常意义上的 Python 语言联系得非常紧密，我们可以将其当成语言规范的一部分。本章将会讲解基本的内置模块。

第 7 章：协作开发

如果许多人要开发同一个 Python 程序，那就得仔细商量代码的写法了。即便你是一个人开发，也需要理解其他人所写的模块。本章讲解多人协作开发 Python 程序时所

用的标准工具及最佳做法。

第 8 章：部署

Python 提供了一些工具，使我们可以把软件部署到不同的环境中。它也提供了一些模块，令开发者可以把程序编写得更加健壮。本章讲解如何使用 Python 调试、优化并测试程序，以提升其质量与性能。

本书使用的约定

本书在 Python 代码风格指南（Python style guide）的基础上做了一些修改，使范例代码便于印刷，也便于凸显其中的重要内容。一行代码比较长时，会以➥字符来表示折行。代码中的某些部分，与当前要讲的问题联系不大，笔者会将这部分代码略去，并在注释中以省略号来表示（# ...）。为了缩减范例代码的篇幅，笔者也把内嵌的文档删去了。读者在开发自己的项目时不应该这么做，而是应该遵循 Python 风格指南（参见本书第 2 条），并为源代码撰写开发文档（参见本书第 49 条）。

书中大部分代码，运行之后都会产生输出（output）。笔者所谓的输出，意思是说：在互动式解释器（interactive interpreter）中运行这些 Python 程序时，控制台或终端机里面会打印出一些信息。这些打印出来的信息，以等宽字体印刷，它们上方的那一行会标有 >>> 符号（这个 >>> 符号是 Python 解释器的提示符）。笔者使用这个符号是想告诉大家：把 >>> 上方的那些范例代码输入 Python shell 之后，会产生与 >>> 下方文字相符的输出信息。

除此之外，还有一些上方虽无 >>> 符号，但却以等宽字体印刷的代码段。这些内容用来表示产生于 Python 解释器之外的输出信息。它们的上方通常都会有 $ 字符，这表示笔者是在 Bash 之类的命令行 shell 里面先运行了程序，然后才产生这些输出的。

获取源代码及勘误表

大家可以抛开本书的讲解部分，把某些范例作为完整的程序运行一遍，这样是很有好处的。你可以用这些代码做实验，以了解整个程序的运行原理。全部源码都可以从本书网站（http://www.effectivepython.com/）下载[㊀]。书中的错误也会张贴到该网站[㊁]。

㊀ 范例代码的网址是：https://github.com/bslatkin/effectivepython。——译者注

㊁ 这里针对的是英文原书，勘误表的网址是：https://github.com/bslatkin/effectivepython/issues。——译者注

致　谢 *Acknowledgements*

在生活中，有很多人给了我指导、支持及鼓励，没有他们，本书就不会面世。

感谢《Effective Software Development》系列的顾问 Scott Meyers。笔者 15 岁那年初次阅读了 Scott 所写的《Effective C++》，当时我就迷上了这门语言。我后来的教育经历，以及在 Google 的第一份工作，无疑都得益于 Scott 的那本书。这次有机会写作本书，本人深感荣幸。

感谢核心技术评审者 Brett Cannon、Tavis Rudd 和 Mike Taylor，他们为本书提供了深刻而透彻的反馈意见。感谢 Leah Culver 和 Adrian Holovaty，他们两位认为写作这样一本书很有意义。感谢友人 Michael Levine、Marzia Niccolai、Ade Oshineye 和 Katrina Sostek，他们耐心阅读了本书的初稿。也感谢 Google 诸位同事审读本书。若没有以上诸君的帮助，本书读起来可能就会比较费解。

感谢制作本书的每一位工作人员。感谢编辑 Trina MacDonald 启动本书制作流程，并提供大力支持。感谢诸位团队成员帮助制作本书，他们是：策划编辑 Tom Cirtin 和 Chris Zahn、助理编辑 Olivia Basegio、营销经理 Stephane Nakib、文字编辑 Stephanie Geels，以及生产编辑 Julie Nahil。

感谢与我共事的诸位优秀 Python 程序员：Anthony Baxter、Brett Cannon、Wesley Chun、Jeremy Hylton、Alex Martelli、Neal Norwitz、Guido van Rossum、Andy Smith、Greg Stein 和 Ka-Ping Yee。很高兴你们能督促并指引我学习 Python。Python 开发社团构建得非常优秀，成为一名 Python 开发者，令我感到特别荣幸。

感谢诸位同事这些年来对我的关照。感谢 Kevin Gibbs 帮助我应对风险。感谢 Ken Ashcraft、Ryan Barrett 和 Jon McAlister 教会我如何工作。感谢 Brad Fitzpatrick 帮助我提升工作能力。感谢 Paul McDonald 陪我一起创建我们的搞怪项目。感谢 Jeremy Ginsberg

和 Jack Hebert 令其成为现实。

感谢激发我编程兴趣的诸位老师：Ben Chelf、Vince Hugo、Russ Lewin、Jon Stemmle、Derek Thomson 和 Daniel Wang。正因为有了你们的指引，我才会努力磨练编程技术，进而使自己有能力去教导他人。

感谢母亲使我找到了人生的目标并鼓励我做程序员。感谢兄弟、祖父母、众亲戚以及儿时的玩伴，从小你们就是我的榜样，也使我找到了成长的快乐。

最后要感谢我的妻子 Colleen，感谢她的关爱和支持，感谢她带来的欢笑。

目　录 *Contents*

第 1 章 Chapter 1

用 Pythonic 方式来思考

一门语言的编程习惯是由用户来确立的。这些年来，Python 开发者用 Pythonic 这个形容词来描述那种符合特定风格的代码。这种 Pythonic 风格，既不是非常严密的规范，也不是由编译器强加给开发者的规则，而是大家在使用 Python 语言协同工作的过程中逐渐形成的习惯。Python 开发者不喜欢复杂的事物，他们崇尚直观、简洁而又易读的代码（请在 Python 解释器中输入 import this）。

对 C++ 或 Java 等其他语言比较熟悉的人，可能还在按自己喜欢的风格来使用 Python；而刚刚接触 Python 的程序员，则需要逐渐熟悉许多可以用 Python 代码来表达的概念。但无论哪一种开发者，都必须知道如何以最佳方式完成常见的 Python 编程工作，这种最佳方式，就是 Pythonic 方式。该方式将会影响你所写的每个程序。

第 1 条：确认自己所用的 Python 版本

本书绝大部分范例代码都遵循 Python 3.4（发布于 2014 年 3 月 17 日）的语法。某些范例还会同时给出 Python 2.7（发布于 2010 年 7 月 3 日）版本的代码，以强调两者的区别。笔者给出的建议适用于 CPython、Jython、IronPython 及 PyPy 等流行的 Python 运行时环境。

很多电脑都预装了多个版本的标准 CPython 运行时环境[⊖]。然而，在命令行中输入

⊖ 在不引起混淆的情况下，可以把 CPython 理解成默认的 Python 解释器。下同。——译者注

默认的 python 命令之后，究竟会执行哪个版本则无法肯定。python 通常是 python 2.7 的别名，但也有可能是 python 2.6 或 python 2.5 等旧版本的别名。请用 --version 标志来运行 python 命令，以了解所使用的具体 Python 版本。

```
$ python --version
Python 2.7.8
```

通常可以用 python3 命令来运行 Python 3。

```
$ python3 --version
Python 3.4.2
```

运行程序的时候，也可以在内置的 sys 模块里查询相关的值，以确定当前使用的 Python 版本。

```
import sys
print(sys.version_info)
print(sys.version)
```

```
>>>
sys.version_info(major=3, minor=4, micro=2,
➥releaselevel='final', serial=0)
3.4.2 (default, Oct 19 2014, 17:52:17)
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.51)]
```

Python 2 和 Python 3 都处在 Python 社区的积极维护之中。但是 Python 2 的功能开发已经冻结，只会进行 bug 修复、安全增强以及移植等工作，以便使开发者能顺利从 Python 2 迁移到 Python 3。2to3 与 six[⊖]等工具可以帮助大家把代码轻松地适配到 Python 3 及其后续版本上面。

Python 3 经常会添加新功能并提供改进，而这些功能与改进不会出现在 Python 2 中。笔者写作本书时，大部分 Python 开源代码库都已经兼容 Python 3 了，所以强烈建议大家使用 Python 3 来开发自己的下一个 Python 项目。

要点

- 有两个版本的 Python 处于活跃状态，它们是：Python 2 与 Python 3。
- 有很多种流行的 Python 运行时环境，例如，CPython、Jython、IronPython 以及 PyPy 等。
- 在操作系统的命令行中运行 Python 时，请确保该 Python 的版本与你想使用的 Python 版本相符。

⊖ 该工具的网址是：https://pythonhosted.org/six/。——译者注