TURING 图灵程序设计丛书

Python Tricks: A Buffet of Awesome Python Features

深入理解Python特性

[德] 达恩·巴德尔 著 孙波翔 译

- 影响全球1 000 000以上程序员的PythonistaCafe社区创始人Dan Bader手把手带你提升Python实践技能，快速写出更高效、更专业的Python代码

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Python Tricks: A Buffet of Awesome Python Features

深入理解Python特性

[德] 达恩·巴德尔 著 孙波翔 译

人民邮电出版社
北京

图书在版编目（CIP）数据

深入理解Python特性 / （德）达恩·巴德尔（Dan Bader）著 ; 孙波翔译. -- 北京 : 人民邮电出版社, 2019.6
（图灵程序设计丛书）
ISBN 978-7-115-51154-6

Ⅰ. ①深… Ⅱ. ①达… ②孙… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.561

中国版本图书馆CIP数据核字(2019)第079486号

内 容 提 要

本书致力于帮助 Python 开发人员挖掘这门语言及相关程序库的优秀特性，避免重复劳动，同时写出简洁、流畅、易读、易维护的代码。通过本书，你会明白用好 Python 需要了解的重要特性，从 Python 2 过渡到 Python 3 需要掌握的现代模式，以及有其他编程语言背景、想快速上手 Python 的程序员需要特别注意的问题，等等。

本书面向所有 Python 开发人员，以及所有对 Python 感兴趣的程序员。

◆ 著 [德] 达恩·巴德尔
译 孙波翔
责任编辑 杨 琳
责任印制 周昇亮

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：11
字数：267千字 2019年6月第1版
印数：1－3 500册 2019年6月北京第1次印刷
著作权合同登记号 图字：01-2018-3263号

定价：49.00元

读者服务热线：(010)51095183转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

版权声明

Python 高手对本书的评论

“我非常非常喜爱本书，它就像一位经验丰富的导师从旁解释各种小技巧一样。我在工作中学会了 PowerShell，现在在学习 Python，并且了解了很多很棒的新东西。在学习的过程中，每当遇到困难（通常是在 Flask 蓝图上遇到问题或觉得代码可以更加具有 Python 特色），我都会在公司内部的 Python 聊天室发布问题。

“同事们给出的答案常常让我惊讶，其中经常含有字典解析式、lambda、生成器这些技巧。在掌握并正确实现这些技巧后，我总是惊叹 Python 的强大。

“之前，我是一名迷茫的 PowerShell 脚本用户；而现在，这本书让我能正确、合理地使用这些常用且具有 Python 特色的技巧。

“我没有计算机科学学位，因此很高兴能有人用文字解释那些只有科班出身的人才懂的知识。我非常喜欢这本书，并且订阅了电子邮件，而本书也是我通过电子邮件了解到的。”

——**Daniel Meyer**，特斯拉 DA

“第一次听说本书，是因为一位同事想用书中的字典示例来考我。我当时几乎可以确定最终结果是一个更小、更简单的字典，但必须承认，结果还是出乎我的意料。:)

“他通过视频向我展示了本书，并翻页让我浏览了一下，我当时就求知欲爆棚，想要阅读更多的内容。

“当天下午我就购买了本书，并读完了其中对 Python 字典创建方式的解释。当天晚些时候，和另一位同事一起喝咖啡时，我也考了他这个问题。:)

“他基于相同的原理提出了另一个问题，但得益于书中条理分明的解释，我不用再猜结果了，而是给出了正确答案。这说明本书在讲解方面做得很好。:)

“我并不是 Python 新手，也熟悉书中介绍的一些概念，但我不得不说本书的每一章都让我受益良多。作者编写了一本好书，并非常出色地解释了这些技巧背后的概念。我一定会向朋友和同事推荐本书！”

——**Og Maciel**，Red Hat 工程师

“我非常喜欢读达恩的这本书。他用清晰的示例解释了 Python 的重要方面（例如使用双胞胎猫解释 `is` 和`==`）。

“书中除了给出代码示例，还解释了相关的实现细节。更重要的是，本书可以让你编写出更好的 Python 代码！

“实际上，本书让我最近养成了一些新的 Python 好习惯，例如使用自定义异常和抽象基类（我在搜索‘抽象类’时发现了达恩的博客）。这些新知识本身就让本书物有所值。”

——**Bob Belderbos**，Oracle 工程师、PyBites 联合创始人

序

从我第一次接触 Python 这门编程语言到现在已经有将近 10 年了。多年前第一次学习 Python 时，我还有点不情愿。在此之前，我使用另一门语言编程，但在工作中突然被分配到了另一个团队，其中每个人都在使用 Python。我的 Python 之旅就从那里开始了。

第一次接触 Python 时，我被告知 Python 很容易，能快速上手。当我向同事询问学习 Python 的资源时，他们只会给我 Python 官方文档的链接。刚上手就阅读文档会让人一头雾水，我就这样挣扎了一段时间，之后才慢慢适应。遇到问题时，我通常需要在 Stack Overflow 上寻找答案。

由于之前使用过另一门编程语言，我没有寻找介绍如何编程或者什么是类和对象这样的入门资源，而是一直在寻找能够介绍 Python 专有特性的资料，并尝试了解用 Python 编程与使用其他语言有何区别。

我花了好几年才充分理解了这门语言。当我阅读达恩的这本书时就在想，要是在当初开始学习 Python 时能有这样一本书该有多好。

举例来说，在众多独特的 Python 特性中，首先让我感到惊讶的是列表解析式。正如达恩在书中提到的那样，从编写 `for` 循环的方式就能看出一个人是否刚从其他语言转到 Python。我记得在刚用 Python 编程时，最初得到的代码审查评论中有一条就是：“为什么不在这里使用列表解析式？”达恩在第 6 章中清楚地解释了这个概念。他首先介绍了如何用具有 Python 特色的方式编写循环，之后介绍了迭代器和生成器。

在 2.5 节中，达恩介绍了几种在 Python 中格式化字符串的方法。字符串格式化无视了“Python 之禅”，即做一件事应该只有一种明确的方式。达恩介绍了几种不同的处理方式，其中包括我最喜欢的 Python 新增功能 f-string。除此之外，他还介绍了每种方法的优缺点。

第 8 章是本书的另一个亮点，其中介绍了 Python 编程语言之外的内容，包括如何调试程序和管理依赖关系，并且一窥了 Python 字节码的究竟。

我很荣幸也很乐意推荐我的朋友达恩·巴德尔编写的这本书。

通过以 CPython 核心开发人员的身份向 Python 做贡献，我与许多社区成员建立了联系。在我的 Python 之旅中，我遇到了不少导师、志同道合者，并结交了许多新朋友。这些经历提醒我，

Python 不仅仅是一门编程语言，更是一个社区。

掌握 Python 编程不仅要掌握该语言的理论方面，理解和采用社区使用的惯例和最佳实践同样重要。

达恩的书会帮助你完成这个旅程。我相信读完本书后，你在编写 Python 程序时会更有信心。

Mariatta Wijaya
Python 核心开发人员（mariatta.ca）

目　录

第 1 章 简 介

1.1 什么是 Python 技巧

> Python 技巧：一小段可以作为教学工具的代码。一个 Python 技巧要么简要介绍了 Python 的一个知识点，要么作为一个启发性的示例，让你自行深入挖掘，从而在大脑中形成直观的理解。

最初，这些 Python 技巧来自于我某一周在 Twitter 上分享的一组代码截图。出乎意料的是，大家的反响非常强烈，一连好几天不停地分享和转发我的 Python 技巧。

随后，许多开发人员问我有没有“完整合集”。其实我只是整理了一部分涵盖不同 Python 主题的技巧，并没有什么大的计划。这仅仅是一个有趣的 Twitter 小实验。

但从这些询问中，我意识到之前发布的示例代码完全可以用作教学工具。因此，我整理出更多的 Python 技巧，并用电子邮件分享给读者。让我吃惊的是，几天之内就有数百位 Python 开发人员注册订阅。

在随后的几周里，Python 开发人员读者已经形成了稳定的客户流。他们感谢我让曾经困扰过他们的 Python 知识点变得通俗易懂。收到这些反馈让我感觉棒极了。我以为这些 Python 技巧只不过是一些代码截图，但是许多开发人员因此受益匪浅。

因此，我决定在这个 Python 技巧的实验上倾注更多努力，将其扩展为包括约 30 封邮件的一个系列。每一封邮件只有一个标题和一幅代码截图，但我很快意识到这种格式有缺陷。那时有个视力存在障碍的 Python 开发人员很失望地通过邮件告诉我，他的屏幕阅读器无法读出这些以图片形式发送的 Python 技巧。

显然，我需要在这个项目上多花一点时间来吸引更多的人，同时让更多的读者受益。因此，我用纯文本加上适当的 HTML 语法高亮，重新创建了所有介绍 Python 技巧的邮件。新版的 Python 技巧稳定发布了一阵子。我从收到的反馈得知开发人员很开心，因为他们终于能够复制和粘贴代码来自己尝试了。

随着越来越多的开发人员订阅“Python技巧”这个系列的电子邮件，我从收到的回复和问题中发现了一个问题：有些技巧本身就足以作为启发性的示例，但一些比较复杂的示例则缺少一个讲述者来引导读者，也没有介绍一些有助于进一步理解的资料。

这是另一个可以大幅改进的地方。当初我创建 dbader.org 的目标就是**帮助 Python 开发人员提升自己**，显然现在正好有一个机会让我更加接近这个目标。

因此，我决定提取出之前电子邮件中最有价值的那些 Python 技巧，在此基础上编写一本书：

- 以短小且易于理解的示例介绍 Python 最酷的方面；
- 以“自助餐”的形式介绍一些优秀的 Python 特性，激励读者提升自己的能力；
- 手把手地引导读者更加深入地理解 Python。

我写本书完全是出于对 Python 的热爱，同时它也是一个巨大的实验。希望你能够喜欢，并在阅读过程中学到相关的 Python 知识。

1.2　本书作用

本书的目标是让你成为更加高效的 Python 开发人员，且知识和实践能力都获得提升。你可能会奇怪：**阅读本书为什么会获得这种能力上的提升？**

本书并不是循序渐进的 Python 教程，也不是入门级的 Python 课程。如果你在 Python 方面刚起步，靠本书并不会成为资深 Python 开发人员。虽然在这种情况下阅读本书依然有帮助，但你还是要靠其他材料来掌握 Python 的基本技能。

如果你对 Python 已经有了一定的了解，那么就能充分利用本书，并借此进入下一个阶段。如果你已经使用了一阵子 Python 并准备好更进一步，或是想对已掌握的知识进行归纳总结，或是想让代码更具 Python 特色，那么本书同样非常有用。

如果你已经掌握了其他编程语言并想快速掌握 Python，本书同样大有帮助。从本书中，你会发现许多实践技巧和设计模式，能让你成为更高效、更专业的 Python 程序员。

1.3　如何阅读本书

阅读本书最好的方法是将其看作含有各种强大 Python 特性的“自助餐”。书中的每个 Python 技巧都是独立的，所以你完全可以从一个技巧跳到另一个感兴趣的技巧。实际上，我也鼓励你这么做。

当然，你也可以按顺序通读本书，这样就不会错过书中的任何一个 Python 技巧。

有些技巧很简单、容易理解，读一遍就能应用到日常工作中。不过有些技巧需要花点时间钻研。

如果你在将某个技巧集成到自己的程序中时遇到了困难，可以先在 Python 解释器的会话中尝试。

如果这样还不行，欢迎随时与我联系。这样我不仅能帮到你，而且能改进本书的讲解方式，长远来看还能帮到所有阅读本书的 Python 爱好者。

第 2 章 Python 整洁之道

2.1 用断言加一层保险

有时，真正有用的语言特性得到的关注反而不多，比如 Python 内置的 `assert` 语句就没有受到重视。

本节将介绍如何在 Python 中使用断言。你将学习用断言来自动检测 Python 程序中的错误，让程序更可靠且更易于调试。

读到这里，你可能想知道什么是断言，以及它到底有什么好处。下面就来一一揭晓答案。

从根本上来说，Python 的断言语句是一种调试工具，用来测试某个断言条件。如果断言条件为真，则程序将继续正常执行；但如果条件为假，则会引发 `AssertionError` 异常并显示相关的错误消息。

2.1.1 示例：Python 中的断言

下面举一个断言能派上用场的简单例子。本书中的例子会尝试结合你可能在实际工作中遇到的问题。

假设你需要用 Python 构建在线商店。为了添加打折优惠券的功能，你编写了下面这个 `apply_discount` 函数：

```
def apply_discount(product, discount):
    price = int(product['price'] * (1.0 - discount))
    assert 0 <= price <= product['price']
    return price
```

注意到 `assert` 语句了吗？这条语句确保在任何情况下，通过该函数计算的折后价不低于 0，也不会高于产品原价。

来看看调用该函数能否正确计算折后价。在这个例子中，商店中的产品用普通的字典表示。

这样能够很好地演示断言的使用方法，当然实际的应用程序可能不会这么做。下面先创建一个示例产品，即一双价格为 149 美元的漂亮鞋子：

```
>>> shoes = {'name': 'Fancy Shoes', 'price': 14900}
```

顺便说一下，这里使用整数来表示以分为单位的价格，以此来避免货币的舍入问题。一般而言，这是个好办法……好吧，有点扯远了。现在如果为这双鞋打七五折，即优惠了 25%，则售价变为 111.75 美元：

```
>>> apply_discount(shoes, 0.25)
11175
```

嗯，还不错。接着再尝试使用一些无效的折扣，比如 200%的“折扣”会让商家向顾客付钱：

```
>>> apply_discount(shoes, 2.0)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
    apply_discount(prod, 2.0)
  File "<input>", line 4, in apply_discount
    assert 0 <= price <= product['price']
AssertionError
```

从上面可以看到，当尝试使用无效的折扣时，程序会停止并触发一个 `AssertionError`。发生这种情况是因为 200%的折扣违反了在 `apply_discount` 函数中设置的断言条件。

从异常栈跟踪信息中还能得知断言验证失败的具体位置。如果你（或者团队中的另一个开发人员）在测试在线商店时遇到这些错误，那么查看异常回溯就可以轻松地了解是哪里出了问题。

这极大地加快了调试工作的速度，并且长远看来，程序也更易于维护。朋友们，这就是断言的力量。

2.1.2 为什么不用普通的异常来处理

你可能很奇怪为什么不在前面的示例中使用 `if` 语句和异常。

要知道，断言是为了告诉开发人员程序中发生了**不可恢复**的错误。对于可以预料的错误（如未找到相关文件），用户可以予以纠正或重试，断言并不是为此而生的。

断言用于程序**内部自检**，如声明一些代码中**不可能**出现的条件。如果触发了某个条件，即意味着程序中存在相应的 bug。

如果程序没有 bug，那么这些断言条件永远不会触发。但如果违反了断言条件，程序就会崩溃并报告断言错误，告诉开发人员究竟违反了哪个“不可能”的情况。这样可以更轻松地追踪和修复程序中的 bug。我喜欢能让生活变轻松的东西，你也是吧？

现在请记住，Python 的断言语句是一种调试辅助功能，不是用来处理运行时错误的机制。使