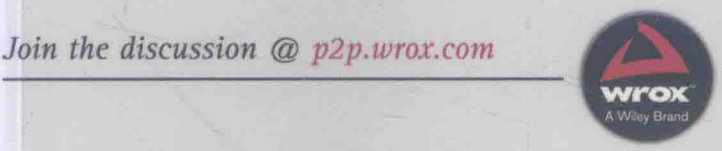

Professional Python

Python高级编程

[美] Luke Sneeringer 著
宋沄剑 刘磊 译

清华大学出版社

Python 高级编程

[美] Luke Sneeringer 著

宋沄剑 刘 磊 译

清华大学出版社

北 京

Luke Sneeringer

Professional Python

EISBN：978-1-119-07085-6

图书在版编目(CIP)数据

Python高级编程 / (美)卢克 • 斯内因格(Luke Sneeringer)著；宋沄剑，刘磊 译. —北京：清华大学出版社，2016

书名原文：Professional Python
ISBN 978-7-302-45285-0

Ⅰ. ①P…　Ⅱ. ①卢…　②宋…　③刘…　Ⅲ. ①软件工具—程序设计　Ⅳ. ①TP311.56

中国版本图书馆 CIP 数据核字(2016)第 246895 号

责任编辑：王　军　于　平
封面设计：牛艳敏
版式设计：思创景点
责任校对：成凤进
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：三河市君旺印务有限公司
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：16.5　　字　　数：412 千字
版　　次：2016 年 11 月第 1 版　　印　　次：2016 年 11 月第 1 次印刷
印　　数：1～3500
定　　价：49.80 元

产品编号：067812-01

译者序

Python是一门适用面极广的语言，被广泛应用于Web开发、网络开发、系统开发、科学计算、机器学习、数据分析、数据可视化等领域，而近些年数据科学的火爆更为Python的流行添砖加瓦。但让Python如此流行的更重要原因是：Python具有极好的可读性。

在国内，开发人员常常漠视可维护性、可测试性和质量。这种不幸局面的最终结果是项目协作困难、代码难以维护、修改风险极高。尽管Python语言的设计者和Python社区都非常重视编写干净、可维护的代码，但是仍然很容易出现相反的局面。我相信造成这种结果主要是由于Python开发人员并没有对Python的高级特性进行深入了解，没有对Python编写的最佳实践进行学习。缺乏这些技能，或许可以编写出完成功能的代码，但编写完成优雅、可维护的代码就力不从心，就像学会少量文字可以让我们写字，却不能让我们写诗。

本书不是为Python初学者准备的，文中很多语法和使用情景都要求读者有相关的技术基础。比如第Ⅰ部分关于函数的高级特性，如生成器、装饰符等；第Ⅱ部分关于类的高级特性，如抽象类、类工厂，第Ⅲ部分数据以及其他高级主题都需要你对Python基础有一定的理解。如果有其他语言的编程经验，也同样很有帮助。如果你有一定的编程经验，想要了解Python高级特性的缘起和内部工作机制，从而能够写出更优雅、更可读的代码，那么本书一定是你的不二之选。

在这里要感谢清华大学出版社的编辑，他们为本书的出版投入了巨大的热情并付出了很多心血。没有他们的帮助和鼓励，本书不可能顺利付梓。

在本书翻译过程中，译者本着忠于原文的态度，在翻译过程中力求真实再现原文风貌，但是鉴于译者水平有限，错误和失误在所难免，如有任何意见和建议，请不吝指正。本书全部章节由宋沄剑、刘磊翻译，参与本书翻译的还有郝奇、张毅、刘艳波、田冠华、李新、李宝安、毛瑞娟、刘轶宽。

译　者

作 者 简 介

Luke Sneeringer 曾为多家公司设计、架构、构建和贡献了大量 Python 应用程序，这些公司包括 FeedMagnet、May Designs 与 Ansible。此外，他经常作为讲师出席各种 Python 会议。他与妻子 Meagan 以及他们喜欢的猫和鱼共同住在美国德克萨斯州的奥斯汀。

技术编辑简介

Alan Gauld 是一位通过认证的开放组体系结构框架(The Open Group Architecture Framework，TOGAF)企业架构师，在电信与客户服务行业工作。他从 1974 年开始编程，从 1998 年开始使用 Python。他是两本有关 Python 书籍的作者。在业余时，他爱好徒步、摄影、旅行与音乐。

Elias Bachaalany 是一名计算机程序员、软件逆向工程师及技术作者。Elias 还与人合著了 *Practical Reverse Engineering* (Wiley, 2014)与 *The Antivirus Hacker's Handbook* (Wiley, 2015)。在供职于 Hex-Rays S.A 期间，他改善了 IDA Pro 的脚本设施并为 IDAPython 项目贡献代码。

致　谢

如果没有编辑 Kevin Shafer 以及技术编辑 Alan Gauld 与 Elias Bachaalany，本书不会顺利出版。他们的努力使本书更加完善(并减少了其中的大量错误)。Wiley 出版社整个团队的杰出工作使得我的开始相对没有吸引力的手稿成为现在的图书。

尤其感谢 Jason Ford——我亲密的朋友和杰出的企业家，他给了我一份能够源源不断找到乐趣的工作。他给了我专业性编写 Python 书籍的第一个机会，并使 Python 成为每天有趣的问题、有趣的辩论以及兴奋不已(嗯，兴奋的原因还包括薪水)的来源。

我还要对很多 Python 社区内外的朋友表示感激，他们在过去这些年陪我一起工作或娱乐。有很多名字需要列举，我的良心不允许我遗漏任何一位：Mickie Betz、Frank Burns、 David Cassidy、Jon Chappell、Diana Clarke、George Dupere、John Ferguson、Alex Gaynor、Jasmin Goedtel 、Chris Harbison、Boyd Hemphill、Rob Johnson、Daniel Lindsley、 Jeff McHale、Doug Napleone、Elli Pope、Tom Smith 与 Caleb Sneeringer。

感谢我的父母 Jim Sneeringer 与 Cheryl Sneeringer，他们教会我如何生活。

最后，如果没有一个完整的段落来感谢支持、奉献和爱我的妻子 Meagan，就不是完整的致谢。她说服我本书值得一写，并在写作过程的每一步都亲切地支持我。我非常幸福并且非常感激她陪伴在我生命中的每一天。

—Luke Sneeringer

前　言

最近，Python 已经成为越来越多开发人员的首选语言。Python 的使用者遍布全球，他们将该语言用于各种目的。随着 Python 的广泛应用，越来越多的开发人员都在花时间编写 Python。

Python 稳定增长的原因在于它是一门强大的语言，但很多经验丰富的 Python 开发人员也仅仅了解该语言所能完成工作的冰山一角。

本书读者对象

本书的目标读者是那些已经使用 Python 工作并熟悉它，而且希望进一步学习的开发人员。本书假定读者已经完成过 Python 开发过程中的大多数基本任务(例如使用过 Python 交互式终端)。

如果你是希望寻求由中级到高级的 Python 语言功能的读者，就应该从头至尾阅读本书。

你也可能曾经使用过更高级的语言功能，或可能需要维护使用这些功能的代码，那么可以考虑将本书作为参考手册；或者处理具体实现代码时可以将本书作为索引，从而充实对高级功能的理解。

本书内容

本书涵盖所有最新版本的 Python(包括 Python 2 与 Python 3)。在编写本书时，最新的可用版本是 Python 3.4，Python 3.5 还是 beta 版。本书主要涵盖了 Python 2.6/2.7/3.3/3.4。本书提供的大多数代码既能在 Python 2 上运行，也能在 Python 3 上运行，Python 2 代码会特别标注出来。

此外，本书还占用一整章的篇幅深入讲解 Python 2 与 Python 3 的区别，其中提供了编写跨版本运行代码的建议，以及如何将代码移植到 Python 3。

本书主要集中在两个领域。第一个领域是语言功能本身。例如，本书用几章阐述关于 Python 类与对象模型工作机制的多个方面。第二个领域是作为标准库的一部分提供的模块。例如，本书用 3 章阐述了 3 个模块：asyncio、unittest 与 argparse。

本书结构

本书可以分为 4 个部分：

本书的前 3 章介绍 Python 中的函数。前两章分别阐述装饰器与上下文管理器，它们是用于修改或为增加功能封装函数的可重用方式。第 3 章则是关于生成器的内容，生成器是一种设计函数的方法，它可以使函数一次返回一个值，而不是提前创建一整列值并将其一次性返回。

第 II 部分包含第 4～7 章，介绍 Python 类与语言对象模型。第 4 章阐述魔术方法，第 5 章和第 6 章分别阐述元类和类工厂，这是以强大方式构建类的两种方式。最后，第 7 章阐述抽象基类，解释 abc 模块，以及如何让类来声明实现的模式。

第III部分包含第 8 章和第 9 章，介绍字符串与数据。第 8 章阐述如何在 Python 中使用 Unicode 字符串(与使用字节字符串相比较)，详细介绍了在 Python 2 与 Python 3 中字符串的区别。第 9 章则阐述正则表达式，包含了 Python 的 re 模块，以及如何编写正则表达式。

最后，第Ⅳ部分包含无法融入前三部分的所有内容。第 10 章深入介绍 Python 2 与 Python 3 的区别，以及如何编写能够兼容这两个平台的代码。第 11 章介绍单元测试，主要关注 unittest 模块。关于命令行界面(CLI)工具的第 12 章介绍有关 optparse 与 argparse 的内容，这两个 Python 模块用于编写命令行工具。第 13 章介绍 asyncio，它是一个新的异步编程库，在 Python 3.4 中被引入到 Python 标准库。第 14 章介绍代码风格。

使用本书的前提条件

首先，需要一台运行 Python 的计算机。

虽然在大多数章节中没有区别，但本书在方法上稍微有点以 Linux 为中心(在第 12 章区别最大)。本书的示例在 Linux 环境中运行，如果在 Windows 环境中运行，输出结果可能会有略微不同。

勘误表

尽管我们已经尽了各种努力来保证书中或代码中不出现错误，但是人无完人，错误在所难免。如果发现本书中的错误，如拼写错误或代码错误，请告诉我们，我们将不胜感激。通过发送勘误表，可以让其他读者避免几个小时的困惑，同时可以帮助我们提供更高质量的信息。

要在网站上找到本书的勘误表，请访问 http://www.wrox.com 网页，点击勘误表链接。在这个网页上，您能看到本书提供的和 Wrox 编辑发布的所有勘误表。

如果在本书勘误页面上找不到提供的勘误表，请访问 www.wrox.com/contact/

techsupport.shtml 网页，填写那个网页上的表格并且发给我们您发现的错误。我们将核对反馈信息，如果正确，我们将在本书的勘误表页面张贴该错误消息，并在本书后续版本中修正这一问题。

P2P.wrox.com

要与作者和同行讨论，请加入 p2p.wrox.com 网站上的 P2P 论坛。这个论坛是一个基于 Web 的系统，可以张贴与 Wrox 图书相关的消息和相关技术，与其他读者和技术用户交流心得。该论坛提供电子邮件订阅功能，当论坛上有新贴发布时，可以给你传送你选择的感兴趣的论题。Wrox 作者、编辑和其他业界专家和读者都会到这个论坛上来探讨问题。

在 http://p2p.wrox.com 上，你将发现许多不同的论坛，它们不仅有助于你阅读本书，而且还有助于你开发自己的应用程序。要加入论坛，可以遵循下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读使用协议，并单击 Agree 按钮。

(3) 填写加入该论坛所需要的信息和自己希望提供的其他可选信息，单击 Submit 按钮。

(4) 你会收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

> 注意：不加入 P2P 就可以阅读论坛中的信息，但是必须加入论坛才能发布自己的信息。

加入论坛后，可以发布新消息和回复其他用户发布的信息。可以随时阅读这个网站上的信息。如果想收到特定论坛发来的新消息，单击论坛列表中该论坛名旁边的 Subscribe to this Forum 图标。

要想更多了解如何使用 Wrox P2P 的信息，一定要阅读 FAQ，了解有关论坛软件的工作情况，以及 P2P 和 Wrox 图书的许多常见问题的解答。要阅读 FAQ，只需要在任意 P2P 页面上单击 FAQ 链接即可。

目　录

第Ⅰ部分　函　数

第Ⅱ部分　类

第Ⅳ部分　其他高级主题

第Ⅰ部分

函　数

- 第 1 章　装饰器
- 第 2 章　上下文管理器
- 第 3 章　生成器

第 1 章

装 饰 器

装饰器是一个用于封装函数或类的代码的工具。它显式地将封装器应用到函数或类上，从而使它们选择加入到装饰器的功能中。对于在函数运行前处理常见前置条件(例如确认授权)，或在函数运行后确保清理(例如输出清除或异常处理)装饰器都非常有用。对于处理已经被装饰的函数或类本身，装饰器也很有用。例如，装饰器可以将函数注册到信号系统，或者注册到 Web 应用程序的 URI 注册表中。

本章将概要介绍什么是装饰器，以及装饰器如何与 Python 的函数和类交互。本章还列举了几个 Python 标准类库中常见的装饰器。最后，本章提供了编写装饰器并将其附加到函数和类上的指南。

1.1 理解装饰器

究其核心而言，装饰器就是一个可以接受调用也可以返回调用的调用。装饰器无非就是一个函数(或调用，如有_call_method_方法的对象)，该函数接受被装饰的函数作为其位置参数。装饰器通过使用该参数来执行某些操作，然后返回原始参数或一些其他的调用(大概以这种方式与装饰器交互)。

由于函数在 Python 中是一级对象，因此它们能够像其他对象一样被传递到另一个函数。装饰器就是接受另一个函数作为参数，并用其完成一些操作的函数。

实际上这很容易理解。考虑下面一个非常简单的装饰器。它仅仅为被装饰的调用点字符串附加了一个字符串。

```
def decorated_by(func):
    func.__doc__ += '\nDecorated by decorated_by.'
    return func
```

现在，考虑下面这个普通的函数：

```
def add(x, y):
    """Return the sum of x and y."""
    return x + y
```

函数的 docstring 是在第一行指定的字符串。假如在 Python 的 Shell 中对该函数运行 Help 命令，就能看到该字符串。下面是将装饰器应用到 add 函数的示例：

```
def add(x, y):
    """Return the sum of x and y."""
    return x + y
add = decorated_by(add)
```

以下是执行 help 命令得到的结果：

```
Help on function add in module __main__:

add(x, y)
    Return the sum of x and y.
    Decorated by decorated_by.
(END)
```

这里发生了什么？其实就是装饰器修改了 add 函数的_doc_属性，然后返回原来的函数对象。

1.2 装饰器语法

大多数时候开发人员使用装饰器来装饰函数，他们只对装饰过的最终函数感兴趣，而对于未装饰函数的引用最终就变得多余。

正因为如此(也是为了更整洁)，所以定义函数，给它赋一个特定的名称，然后立刻将装饰过的函数赋给相同的名称就不可取。

因此，Python 2.5 为装饰器引入了特殊的语法。装饰器的应用是通过在装饰器的名称前放置一个@字符，并在被装饰函数声明之上添加一行(不包含隐式装饰器的方法签名)来实现的。

下面来看一下如何优先将 decorated_by 装饰器应用到 add 方法：

```
@decorated_by
def add(x, y):
    """Return the sum of x and y."""
    return x + y
```

再次注意，这里没有给@decorated_by 提供方法签名。假设该装饰器有一个单独的位置参数，而这个参数是装饰过的方法(在某些情况下，会看到一个带有其他参数的方法签名，该内容将在本章后面讨论)。