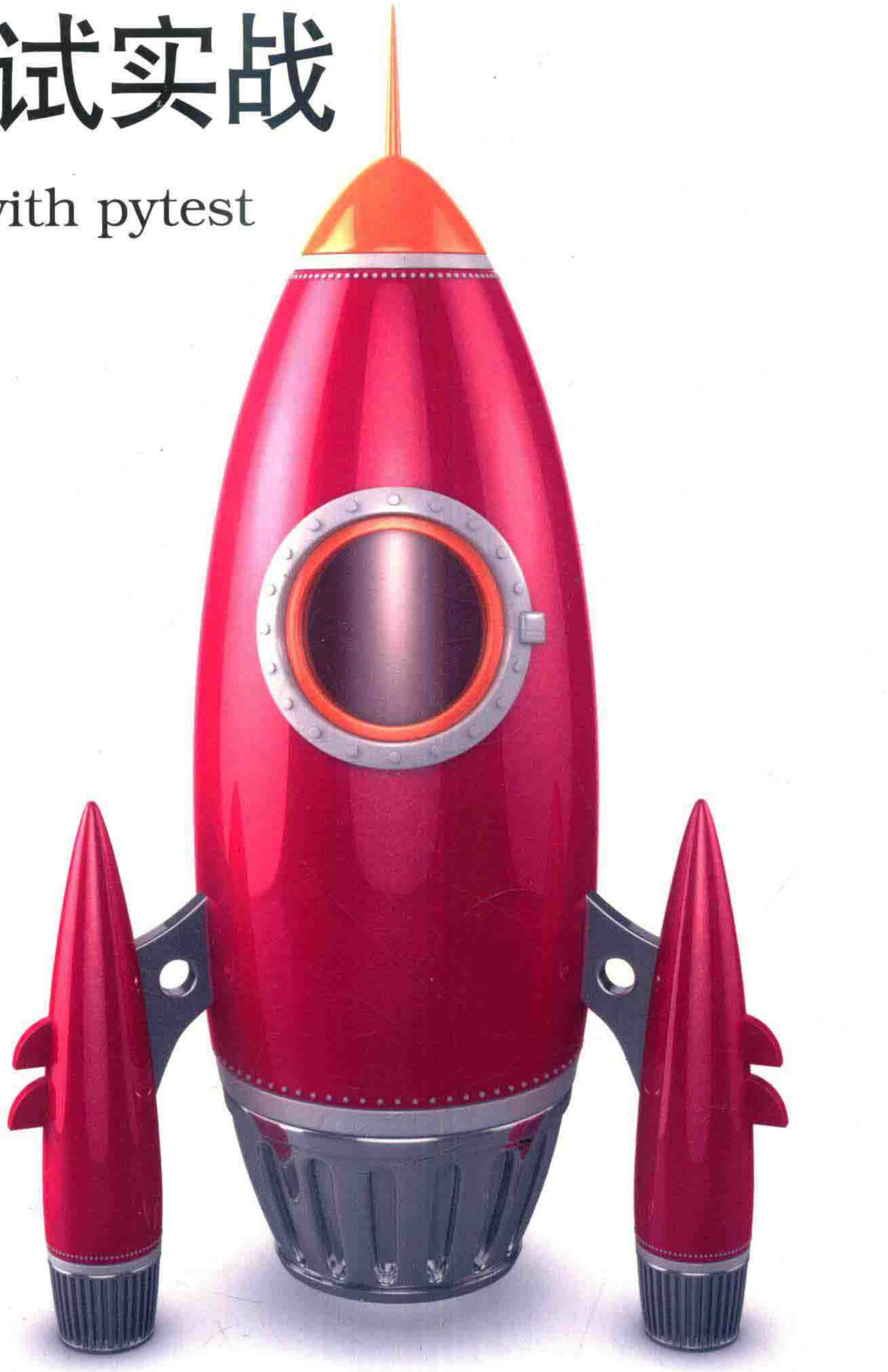
The Pragmatic Programmers

pytest测试实战

Python Testing with pytest



[美] Brian Okken 著

陆阳 饶勇 译

赏味不足 审校

華中科技大學出版社

http://www.hustp.com

pytest 测试实战

[美] Brian Okken 著

陆阳 饶勇 译

赏味不足 审校

华中科技大学出版社

中国·武汉

内 容 简 介

pytest 是动态编程语言 Python 专用的测试框架，它具有易于上手、功能强大、第三方插件丰富、效率高、可扩展性好、兼容性强等特点。本书深入浅出地讲解了 pytest 的使用方法，尤其是具有特色的 fixture 的用法。作者通过丰富的测试实例，手把手教读者编写简洁、易于维护的测试代码和插件，让你轻松掌握这个业界最受欢迎的 Python 测试工具。

湖北省版权局著作权合同登记　图字：17-2018-200 号

图书在版编目(CIP)数据

pytest 测试实战 /（美）布赖恩·奥肯(Brian Okken) 著；陆阳，饶勇译. —武汉：华中科技大学出版社，2018.8

ISBN 978-7-5680-4442-4

Ⅰ.①p…　Ⅱ.①布…　②陆…　③饶…　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(2018)第 164840 号

pytest 测试实战
pytest Ceshi Shizhan

[美]Brian Okken 著
陆阳 饶勇 译

策划编辑：徐定翔
责任编辑：徐定翔
责任监印：周治超
出版发行：华中科技大学出版社(中国·武汉)　电话：(027)81321913
武汉市东湖新技术开发区华工科技园　邮编：430223
录　　排：华中科技大学惠友文印中心
印　　刷：湖北新华印务有限公司
开　　本：787mm×960mm　1/16
印　　张：15
字　　数：361 千字
版　　次：2018 年 8 月第 1 版第 1 次印刷
定　　价：69.90 元

致谢

Acknowledgments

我首先要感谢妻子 Michelle，她也是我的挚友。我的书房是这样的：稿件都堆放在一张略有年代感的橡木方桌上，玻璃书柜上都是我这些年来收集的太空玩具、技术书籍、电路板、杂耍球。铝制储物箱内堆满了便签和装订线，以及宣传书籍用的火箭贴纸。房间一面墙被我和妻子几年前买的天鹅绒覆盖。墙面这样处理可以在录制音频时吸收回声。我喜欢在这里写作，不仅因为这里符合我的个性，更因为这里是 Michelle 与我共同打造的。她总是站在我这边，每当我在博客、播客上发表一些奇思异想时，她都会支持我，包括我计划编写本书。她帮助我确保了写作时间和空间，每当我感到疲惫不堪时，她便会提醒我休息，正如我们上大学时一起熬夜做功课一样。如果没有 Michelle，我的工作和生活将举步维艰。

我有两个聪慧可爱的女儿，Gabriella 和 Sophia，她们都是我的粉丝。Gabriella 和别人聊起编程时，都要提醒别人听我的播客节目；Sophia 上二年级时，书包上贴的都是 Test & Code 字样的贴纸。

我还要感谢编辑 Katharine Dvorak，她帮助我梳理了凌乱的想法，我才能将这些观点系统化地整理成书籍出版。起初我就像写博客一样，文章中夹杂了许多冗余的标题、符号。在 Katharine Dvorak 的悉心指导下，我成为了一名合格的作者。

感谢 Susannah Davidson Pfalzer、Andy Hunt，以及 The Pragmatic Bookshelf 出版公司的所有人。

技术审校在 pytest 规范和 Python 编码规范方面帮助我改正了许多错误，所以本书的示例代码才能遵循 PEP 8 规范。谢谢大家：Oliver Bestwalter、Florian Bruhin、Floris Bruynooghe、Mark Goody、Peter Hampton、Dave Hunt、Al Krinker、Lokesh Kumar Makani、Bruno Oliveira、Ronny Pfannschmidt、Raphael Pierzina、Luciano Ramalho、Frank Ruiz、Dmitry Zinoviev。这几位也都是 pytest 的核心开发者，或者热门插件的作者。

我还要感谢审校者 Luciano。我曾经把前四章书稿发给 Luciano 审阅，他的审校反馈意见毫无疑问是最苛刻的。我虽然没有完全采纳他的意见，但是几乎重写了前三章的内容，他的反馈意见也影响了我后续章节的写作思路。

感谢 pytest 团队创造出了 pytest 这样棒的工具。再次感谢 Oliver Bestwalter、Florian Bruhin、Floris Bruynooghe、Dave Hunt、Holger Krekel、Bruno Oliveira、Ronny Pfannschmidt、Raphael Pierzina，以及解答我疑惑的所有人。

最后我要感谢那些曾经感谢过我的人。曾有读者发邮件说我写的东西帮他们简化了工作，节约了时间。这对我来说是莫大的鼓励，感谢你们。

Brian Okken

2017 年 9 月

前言

Preface

Python 语言越来越流行，不单在软件开发领域，它还深入数据分析、科学研究、测量等领域。随着 Python 在这些重要领域的发展，高效可靠地进行软件测试变得尤为重要。此外，越来越多的软件开发项目开始使用持续集成，而自动化测试正是其中的关键一环。由于软件迭代周期不断缩短，所以手工测试已经行不通。项目团队必须确定测试结果是可靠的，才能放心地发布产品。

于是 pytest 派上用场。

pytest 是什么？
What Is pytest?

pytest 是一款强大的 Python 测试工具，可以胜任各种类型或级别的软件测试工作，既适合开发团队、QA 团队、独立的测试小组使用，又适合练习测试驱动开发的个人，以及开源团队使用。实际上，越来越多的互联网项目开始放弃 unittest 和 nose，转而使用 pytest，比如 Mozilla 和 Dropbox。因为 pytest 会提供更丰富的功能，包括 assert 重写、第三方插件，以及其他测试工具无法比拟的 fixture 模型。

pytest 是一个软件测试框架。它是一款命令行工具，可以自动找到测试用

例执行，并且汇报测试结果。它有丰富的基础库，可以大幅提高用户编写测试用例的效率。它具备可扩展性，用户可以自己编写插件，或者安装第三方提供的插件。pytest 可以直接测试各类 Python 程序，也可以很容易地与其他工具集成到一起使用，比如持续集成、Web 端自动化测试等。

下面列举了一些 pytest 优于其他测试框架的地方。

- 简单的测试可以很简单地编写；
- 复杂的测试也可以很简单地编写；
- 测试的可读性强；
- 易于上手；
- 断言测试失败仅使用原生 `assert` 关键字，而不是 `self.assertEqual()`，或者 `self.assertLessThan()`；
- pytest 可以运行由 unittest 和 nose 编写的测试用例。

pytest 项目是由一个正在快速壮大的社区开发和维护的。它灵活、扩展性好，可以很容易地融入已有的开发测试流程。它不依赖于 Python 版本，Python 2（2.6 及更高版本）和 Python 3（3.3 及更高版本）都可以安装最新版本的 pytest。

通过实用示例学习
Learn pytest While Testing an Example Application

没有人愿意学习实际工作中永远不会出现的例子，我也一样。本书将围绕一个实用的测试项目展开学习，其中的测试技巧很容易运用到你的实际工作中。

Tasks 项目

The Tasks Project

我们的测试对象是一个叫 Tasks 的应用程序。Tasks 是一个小巧的任务进度追踪程序，它有一个命令行交互界面，设计结构与很多应用程序相似。我希望读者在编写针对 Tasks 项目测试用例的同时，能够轻松地掌握相关技巧，并运用到自己的测试项目中去。

虽然 Tasks 程序通过 CLI（command-line interface）交互，但其底层编码是通过调用 API 实现的，因此我们的测试工作主要面向的是 API。API 与数据库控制层交互，数据库控制层与文档数据库（MongoDB 或 TinyDB）交互。数据库的类型是在数据库初始化时配置好的。

在进一步讨论 API 之前，我们先了解一下 Tasks 所使用的命令行交互工具。以下是一段示例代码：

```
$ tasks add 'do something' --owner Brian
$ tasks add 'do something else'
$ tasks list
  ID    owner  done  summary
  --    -----  ----  -------
   1    Brian  False  do something
   2           False  do something else
$ tasks update 2 --owner Brian
$ tasks list
  ID    owner  done  summary
  --    -----  ----  -------
   1    Brian  False  do something
   2    Brian  False  do something else
$ tasks update 1 --done True
$ tasks list
  ID    owner  done  summary
  --    -----  ----  -------
   1    Brian   True  do something
   2    Brian  False  do something else
$ tasks delete 1
$ tasks list
  ID    owner  done   summary
  --    -----  ----   -------
   2    Brian   True  do something
$
```

这个任务管理程序并不复杂，但作为我们的示例已经够用。

测试方法
Test Strategy

虽然 pytest 可以用于单元测试、集成测试、系统测试（端到端测试）、功能测试，但 Tasks 项目将主要采用皮下测试（subcutaneous test）。以下是一些有关测试方法的基本定义。

单元测试 检查一小块代码（如一个函数，或一个类）的测试。第 1 章中的测试就是针对 Tasks 数据结构的单元测试。

集成测试 检查大段的代码（如多个类，或一个子系统）的测试。集成测试的规模介于单元测试与系统测试之间。

系统测试 检查整个系统的测试，通常要求测试环境尽可能接近最终用户的使用环境。

功能测试 检查单个系统功能的测试。就 Tasks 项目而言，针对增加、删除、更新一个任务条目的测试就属于功能测试。

皮下测试 不针对最终用户界面，而是针对用户界面以下的接口的测试。本书中的大部分测试都是针对 API 的，而不是针对 CLI 的，因此这些测试都属于皮下测试。

本书结构
How This Book Is Organized

第 1 章介绍 pytest 的安装，同时会介绍 Tasks 项目的数据结构部分（名为 `Task` 的 `namedtuple`），并用它作为测试示例。我们会学习如何指定测试文件运行，以及 pytest 常用的命令行命令，包括重新运行失败测试、遇到失败即停止

所有测试、控制堆栈跟踪、控制日志输出，等等。

第 2 章将使用 pip 在本地安装 Tasks 项目，学习在 Python 项目中如何组织测试目录，这样才能针对实际项目编写测试用例。这一章的所有示例都依赖外部程序，包括数据库写入。第 2 章的重点是测试函数，你将学习在 pytest 中高效使用断言语句。这一章还会讲解 marker 标记功能的用法，marker 标记可以将测试进行归类或分组，方便一起运行，也可以将某些测试标记为 skip（跳过不执行），marker 标记还可以告诉 pytest 我们知道某些测试是一定会失败的。如果希望运行指定的测试子集，除了使用 marker，还可以将测试代码组织成测试目录、测试模块、测试类，然后运行。

并非所有的测试代码都要放到测试函数中。第 3 章介绍如何将测试数据、启动逻辑、销毁逻辑放入 fixture（pytest 定义的一种测试脚手架）。设置系统（或子系统、系统单元）是软件测试的重要环节，第 3 章将介绍用一个简单的 fixture 完成这方面的工作（包括对数据库进行初始化，写入数据以备测试之用）。Fixture 模块的功能非常强大，你可以利用它简化测试代码，从而提高代码的可读性和可维护性。Fixture 像测试函数一样，也有参数。利用参数，你只需要编写一份代码，就可以针对 TinyDB 和 MongoDB（或其他 Tasks 项目支持的数据库）开展测试。

第 4 章介绍 pytest 内置的 fixture 以满足测试中常见的一些需求，包括生成和销毁临时目录、截取输出流（通过日志判定结果）、使用 monkey patch、检查是否发出警告，等等。

第 5 章讲解如何在 pytest 中添加命令行选项，如何改进打印输出，如何打包分发自己编写的插件，如何共享定制化的 pytest（包括 fixture）。这一章开发的插件可以改善 Tasks 项目测试失败时的输出呈现方式。你还将学习测试自己的测试插件(元测试)。读完这一章，想必你已经等不及编写自己的插件了。附录 C 收集了一些热门的社区插件，可供参考。

第 6 章讲解通过 pytest.ini 文件修改默认配置，自定义 pytest 的运行方式。pytest.ini 文件可以存放某些命令选项，从而减少你重复输入命令的次数；利用它还可以指定 pytest 忽略某些测试目录，或者指定 pytest 的最低版本，等等。使用 tox.ini 和 setup.cfg 文件也可以实现同样的功能。

第 7 章（最后一章）介绍 pytest 与其他工具的结合使用。我们将借助 tox 让 Tasks 项目在多个 Python 版本上运行；学习如何测试 Tasks 项目的 CLI 部分，而不必 mock 系统的其余部分；借助 coverage.py 检查 Tasks 项目代码块的测试覆盖情况；通过 Jenkins 发起测试并实时显示结果。最后，还会学习如何让 pytest 运行基于 unittest 的测试用例，以及把 pytest 的 fixture 共享给 unittest 的测试用例使用。

阅读基础
What You Need to Know

Python

你不必精通 Python，本书的示例语法很常见。

pip

本书使用 pip 安装 pytest 及其组件。获取新版本的 pip，请查阅附录 B。

命令行

我自己使用的是 Mac 笔记本和 bash，实际上，我只使用了 cd（进入指定目录）和 pytest 两条命令。Windows 系统和所有 UNIX 派生系统都有 cd 命令，所以本书的示例也应该适用于这些操作系统。

满足以上条件，就可以阅读本书了。不擅长编程的人一样可以学习使用 pytest 进行自动化测试。

示例代码与共享资源
Example Code and Online Resources

本书示例代码是使用 Python 3.6 和 pytest 3.2 编写的。pytest 3.2 支持 Python 2.6、Python 2.7、Python 3.3 及以后的版本。

Tasks 项目的源代码及测试用例可以从 pragprog.com 网站上本书的页面下载[1]。如果只是为了理解书中的测试，不必下载源代码，这些代码都通俗易懂。但如果你希望深入地理解 Tasks 项目，以便将来更好地运用学到的技巧，那就有必要下载源代码。我们在本书的页面上还提供了最新的勘误信息[2]。

我编程已有二十余年，从未遇到过像 pytest 这样优秀的测试工具。希望各位阅读本书后能够有所收获，也喜欢上 pytest。

---

[1] https://pragprog.com/titles/bopytest/source_code
[2] https://pragprog.com/titles/bopytest/errata

目录

Contents