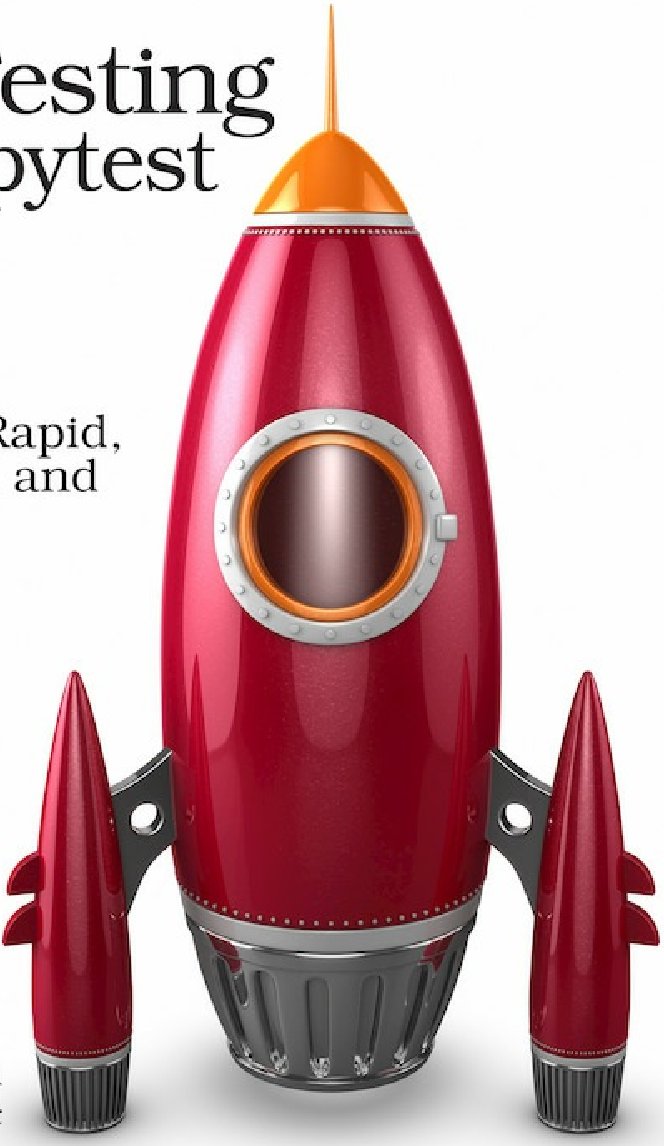
The Pragmatic Programmers

# Python Testing with pytest

Simple, Rapid, Effective, and Scalable



Brian Okken

*edited by Katharine Dvorak*

# Python Testing with pytest

## Simple, Rapid, Effective, and Scalable

## by Brian Okken

Version: P1.0 (September 2017)

Copyright © 2017 The Pragmatic Programmers, LLC. This book is licensed to the individual who purchased it. We don't copy-protect it because that would limit your ability to use it for your own purposes. Please don't break this trust—you can use this across all of your devices but please do not share this copy with other members of your team, with friends, or via file sharing services. Thanks.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and The Pragmatic Programmers, LLC was aware of a trademark claim, the designations have been printed in initial capital letters or in all capitals. The Pragmatic Starter Kit, The Pragmatic Programmer, Pragmatic Programming, Pragmatic Bookshelf and the linking *g* device are trademarks of The Pragmatic Programmers, LLC.

Every precaution was taken in the preparation of this book. However, the publisher assumes no responsibility for errors or omissions, or for damages that may result from the use of information (including program listings) contained herein.

# About the Pragmatic Bookshelf

The Pragmatic Bookshelf is an agile publishing company. We’re here because we want to improve the lives of developers. We do this by creating timely, practical titles, written by programmers for programmers.

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at http://pragprog.com.

Our ebooks do not contain any Digital Restrictions Management, and have always been DRM-free. We pioneered the beta book concept, where you can purchase and read a book while it’s still being written, and provide feedback to the author to help make a better book for everyone. Free resources for all purchasers include source code downloads (if applicable), errata and discussion forums, all available on the book's home page at pragprog.com. We’re here to make your life easier.

## New Book Announcements

Want to keep up on our latest titles and announcements, and occasional special offers? Just create an account on pragprog.com (an email address and a password is all it takes) and select the checkbox to receive newsletters. You can also follow us on twitter as @pragprog.

## About Ebook Formats

If you buy directly from pragprog.com, you get ebooks in all available formats for one price. You can synch your ebooks amongst all your devices (including iPhone/iPad, Android, laptops, etc.) via Dropbox. You get free updates for the life of the edition. And, of course, you can always come back and re-download your books when needed. Ebooks bought from the Amazon Kindle store are subject to Amazon's polices. Limitations in Amazon's file format may cause ebooks to display differently on different devices. For more information, please see our FAQ at pragprog.com/frequently-asked-questions/ebooks. To learn more about this book and access the free resources, go to https://pragprog.com/book/bopytest, the book's homepage.

Thanks for your continued support,

Andy Hunt
The Pragmatic Programmers

The team that produced this book includes: Andy Hunt (Publisher)Janet Furlow (VP of Operations)Katharine Dvorak (Development Editor)Potomac Indexing, LLC (Indexing)Nicole Abramowitz (Copy Editor)Gilson Graphics (Layout)

For customer support, please contact support@pragprog.com.

For international rights, please contact rights@pragprog.com.

# Table of Contents

1. Acknowledgments
2. Preface
    1. What Is pytest?
    2. Learn pytest While Testing an Example Application
    3. How This Book Is Organized
    4. What You Need to Know
    5. Example Code and Online Resources
3. 1. Getting Started with pytest
    1. Getting pytest
    2. Running pytest
    3. Running Only One Test
    4. Using Options
    5. Exercises
    6. What’s Next
4. 2. Writing Test Functions
    1. Testing a Package
    2. Using assert Statements
    3. Expecting Exceptions
    4. Marking Test Functions
    5. Skipping Tests
    6. Marking Tests as Expecting to Fail
    7. Running a Subset of Tests
    8. Parametrized Testing
    9. Exercises
    10. What’s Next
5. 3. pytest Fixtures
    1. Sharing Fixtures Through conftest.py
    2. Using Fixtures for Setup and Teardown
    3. Tracing Fixture Execution with –setup-show
    4. Using Fixtures for Test Data
    5. Using Multiple Fixtures
    6. Specifying Fixture Scope
    7. Specifying Fixtures with usefixtures
    8. Using autouse for Fixtures That Always Get Used
    9. Renaming Fixtures
    10. Parametrizing Fixtures
    11. Exercises
    12. What’s Next
6. 4. Builtin Fixtures
    1. Using tmpdir and tmpdir_factory
    2. Using pytestconfig
    3. Using cache
    4. Using capsys
    5. Using monkeypatch
    6. Using doctest_namespace
    7. Using recwarn

    8. Exercises
    9. What’s Next
7. 5. Plugins
    1. Finding Plugins
    2. Installing Plugins
    3. Writing Your Own Plugins
    4. Creating an Installable Plugin
    5. Testing Plugins
    6. Creating a Distribution
    7. Exercises
    8. What’s Next
8. 6. Configuration
    1. Understanding pytest Configuration Files
    2. Changing the Default Command-Line Options
    3. Registering Markers to Avoid Marker Typos
    4. Requiring a Minimum pytest Version
    5. Stopping pytest from Looking in the Wrong Places
    6. Specifying Test Directory Locations
    7. Changing Test Discovery Rules
    8. Disallowing XPASS
    9. Avoiding Filename Collisions
    10. Exercises
    11. What’s Next
9. 7. Using pytest with Other Tools
    1. pdb: Debugging Test Failures
    2. Coverage.py: Determining How Much Code Is Tested
    3. mock: Swapping Out Part of the System
    4. tox: Testing Multiple Configurations
    5. Jenkins CI: Automating Your Automated Tests
    6. unittest: Running Legacy Tests with pytest
    7. Exercises
    8. What’s Next
10. A1. Virtual Environments
11. A2. pip
12. A3. Plugin Sampler Pack
    1. Plugins That Change the Normal Test Run Flow
    2. Plugins That Alter or Enhance Output
    3. Plugins for Static Analysis
    4. Plugins for Web Development
13. A4. Packaging and Distributing Python Projects
    1. Creating an Installable Module
    2. Creating an Installable Package
    3. Creating a Source Distribution and Wheel
    4. Creating a PyPI-Installable Package
14. A5. xUnit Fixtures
    1. Syntax of xUnit Fixtures
    2. Mixing pytest Fixtures and xUnit Fixtures
    3. Limitations of xUnit Fixtures

Copyright © 2017, The Pragmatic Bookshelf.

# Early praise for *Python Testing with pytest*

I found *Python Testing with pytest* to be an eminently usable introductory guidebook to the pytest testing framework. It is already paying dividends for me at my company.

→ Chris Shaver
VP of Product, Uprising Technology

Systematic software testing, especially in the Python community, is often either completely overlooked or done in an ad hoc way. Many Python programmers are completely unaware of the existence of pytest. Brian Okken takes the trouble to show that software testing with pytest is easy, natural, and even exciting.

→ Dmitry Zinoviev
Author of Data Science Essentials in Python

This book is the missing chapter absent from every comprehensive Python book.

→ Frank Ruiz
Principal Site Reliability Engineer, Box, Inc.

## Plugins for Web Development

Web-based projects have their own testing hoops to jump through. Even pytest doesn't make testing web applications trivial. However, quite a few pytest plugins help make it easier.

### pytest-selenium: Test with a Web Browser

Selenium is a project that is used to automate control of a web browser. The pytest-selenium plugin[51] is the Python binding for it. With it, you can launch a web browser and use it to open URLs, exercise web applications, and fill out forms. You can also programmatically control the browser to test a web site or web application.

### pytest-django: Test Django Applications

Django is a popular Python-based web development framework. It comes with testing hooks that allow you to test different parts of a Django application without having to use browser-based testing. By default, the builtin testing support in Django is based on unittest. The pytest-django plugin[52] allows you to use pytest instead of unittest to gain all the benefits of pytest. The plugin also includes helper functions and fixtures to speed up test implementation.

### pytest-flask: Test Flask Applications

Flask is another popular framework that is sometimes referred to as a microframework. The pytest-flask plugin[53] provides a handful of fixtures to assist in testing Flask applications.

**Footnotes**

[35]

https://pypi.python.org/pypi/pytest-repeat

[36]

https://pypi.python.org/pypi/pytest-xdist

[37]

https://pypi.python.org/pypi/pytest-timeout

[38]

https://pypi.python.org/pypi/pytest-instafail

[39]

https://pypi.python.org/pypi/pytest-sugar

[40]

https://pypi.python.org/pypi/pytest-emoji

[41]

https://pypi.python.org/pypi/pytest-html

[42]

https://www.python.org/dev/peps/pep-0008

[43]

https://pypi.python.org/pypi/pycodestyle

[44]

https://pypi.python.org/pypi/pytest-pycodestyle

[45]

https://pypi.python.org/pypi/pep8

[46]

https://pypi.python.org/pypi/pytest-pep8

[47]

https://pypi.python.org/pypi/flake8

[48]

https://pypi.python.org/pypi/pytest-flake8

[49]

https://pypi.python.org/pypi/flake8-docstrings

[50]

https://www.python.org/dev/peps/pep-0257

[51]

https://pypi.python.org/pypi/pytest-selenium

[52]

https://pypi.python.org/pypi/pytest-django

[53]

```
>>> some_func()
42
>>> exit()
```

That’s a minimal setup, but it’s not realistic. If you’re sharing code, odds are you are sharing a package. The next section builds on this to write a setup.py file for a package.