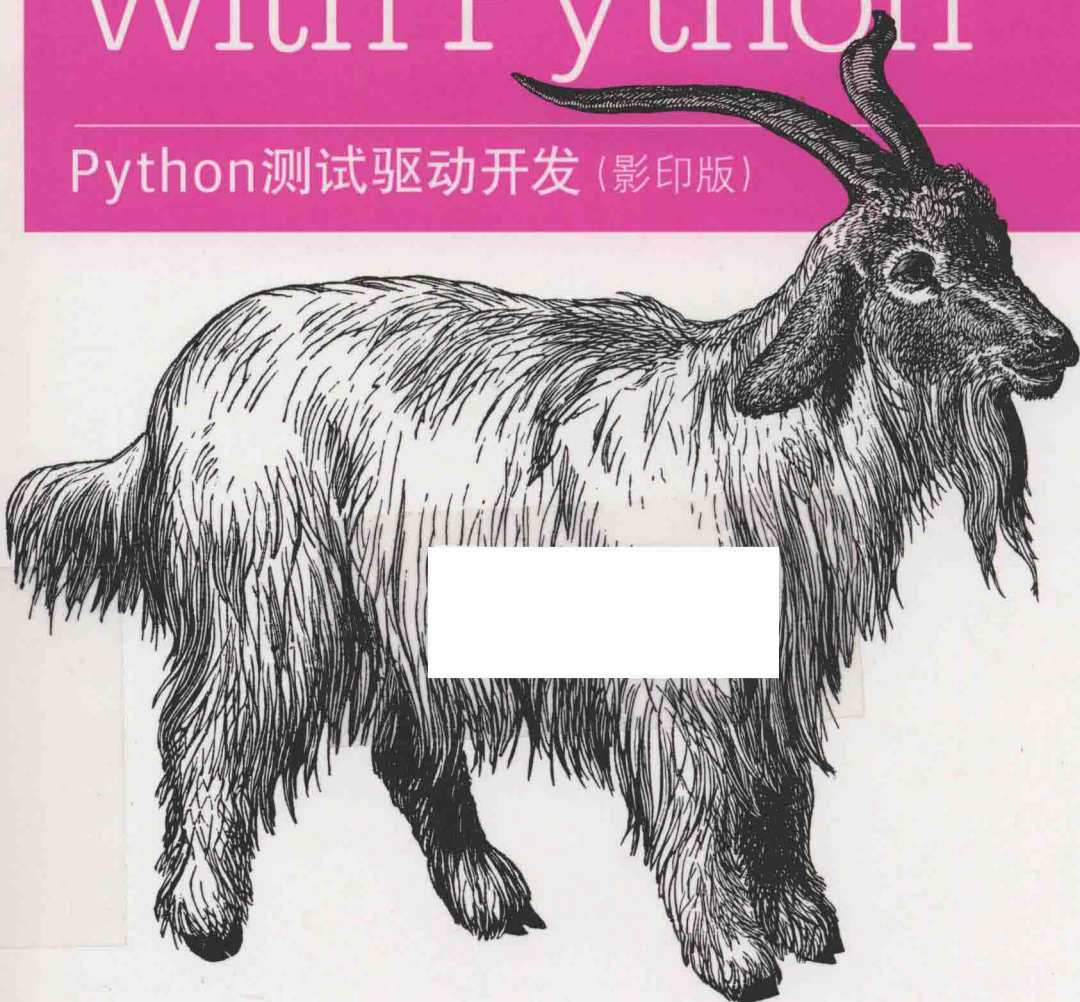


O'REILLY®

Test-Driven Development with Python

Python测试驱动开发 (影印版)



東南大學出版社

Harry J.W. Percival 著

Python (Python) 测试驱动开发

不其利也 (不) 文章, 竟礼也 复通, 竟礼也 复通

Table of Contents

Python 测试驱动开发 (影印版)

Table of Contents

Python测试驱动开发 (影印版)

Test-Driven Development with Python

Foreword 1

Introduction and Acknowledgments 3

Organization of the Book 5

Acknowledgments 7

Part I: The Basics of TDD and Python

1. Getting Django Set Up Using a Functional Test

 What the Testing Goal: Do Nothing, Check, Run Tests

 Creating a Project

 Running a CI Report

2. Extending Our Path Using a Functional Test

 The Python Setup

 Running a CI Report

3. Testing a Simple Model

 Our First Django App

 Unit Tests and Django

 Using Django's TestCase

 Inheritance: We Actually Write Some Application Code

4. Running a CI Report

5. Extending Our Django App

6. Running a CI Report

7. Running a CI Report

8. Running a CI Report

9. Running a CI Report

10. Running a CI Report

11. Running a CI Report

12. Running a CI Report

13. Running a CI Report

14. Running a CI Report

15. Running a CI Report

16. Running a CI Report

17. Running a CI Report

18. Running a CI Report

19. Running a CI Report

20. Running a CI Report

21. Running a CI Report

22. Running a CI Report

23. Running a CI Report

24. Running a CI Report

25. Running a CI Report

26. Running a CI Report

27. Running a CI Report

28. Running a CI Report

29. Running a CI Report

30. Running a CI Report

31. Running a CI Report

32. Running a CI Report

33. Running a CI Report

34. Running a CI Report

35. Running a CI Report

36. Running a CI Report

37. Running a CI Report

38. Running a CI Report

39. Running a CI Report

40. Running a CI Report

41. Running a CI Report

42. Running a CI Report

43. Running a CI Report

44. Running a CI Report

45. Running a CI Report

46. Running a CI Report

47. Running a CI Report

48. Running a CI Report

49. Running a CI Report

50. Running a CI Report

51. Running a CI Report

52. Running a CI Report

53. Running a CI Report

54. Running a CI Report

55. Running a CI Report

56. Running a CI Report

57. Running a CI Report

58. Running a CI Report

59. Running a CI Report

60. Running a CI Report

61. Running a CI Report

62. Running a CI Report

63. Running a CI Report

64. Running a CI Report

65. Running a CI Report

66. Running a CI Report

67. Running a CI Report

68. Running a CI Report

69. Running a CI Report

70. Running a CI Report

71. Running a CI Report

72. Running a CI Report

73. Running a CI Report

74. Running a CI Report

75. Running a CI Report

76. Running a CI Report

77. Running a CI Report

78. Running a CI Report

79. Running a CI Report

80. Running a CI Report

81. Running a CI Report

82. Running a CI Report

83. Running a CI Report

84. Running a CI Report

85. Running a CI Report

86. Running a CI Report

87. Running a CI Report

88. Running a CI Report

89. Running a CI Report

90. Running a CI Report

91. Running a CI Report

92. Running a CI Report

93. Running a CI Report

94. Running a CI Report

95. Running a CI Report

96. Running a CI Report

97. Running a CI Report

98. Running a CI Report

99. Running a CI Report

100. Running a CI Report

Harry J.W. Percival 著

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo **O'REILLY®**

O'Reilly Media, Inc. 授权东南大学出版社出版

南京 东南大学出版社

图书在版编目(CIP)数据

Python 测试驱动开发: 英文/(美)珀西瓦尔
(Percival, H.J.W.)著. —影印本. —南京: 东南大学出版社,
2015.9

书名原文: Test - Driven Development with Python
ISBN 978 - 7 - 5641 - 5915 - 3

I. ①P… II. ①珀… III. ①软件工具—程序设
计—英文 IV. ①TP311.56

中国版本图书馆 CIP 数据核字(2015)第 165738 号

江苏省版权局著作权合同登记
图字: 10 - 2015 - 239 号

© 2014 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press,
2015. Authorized reprint of the original English edition, 2015 O'Reilly Media, Inc., the owner of all rights to
publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2014。

英文影印版由东南大学出版社出版 2015。此影印版的出版和销售得到出版权和销售权的所有者
—— O'Reilly Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式复制。

Python 测试驱动开发(影印版)

出版发行: 东南大学出版社

地 址: 南京四牌楼 2 号 邮编: 210096

出 版 人: 江建中

网 址: <http://www.seupress.com>

电子邮件: press@seupress.com

印 刷: 常州市武进第三印刷有限公司

开 本: 787 毫米×980 毫米 16 开本

印 张: 31.25

字 数: 612 千字

版 次: 2015 年 9 月第 1 版

印 次: 2015 年 9 月第 1 次印刷

书 号: ISBN 978 - 7 - 5641 - 5915 - 3

定 价: 89.00 元

本社图书若有印装质量问题, 请直接与营销部联系。电话(传真): 025 - 83791830

Preface

This book is my attempt to share with the world the journey I've taken from “hacking” to “software engineering”. It's mainly about testing, but there's a lot more to it, as you'll soon see.

I want to thank you for reading it.

If you bought a copy, then I'm very grateful. If you're reading the free online version, then I'm *still* grateful that you've decided it's worth spending some of your time on. Who knows, perhaps once you get to the end, you'll decide it's good enough to buy a real copy for yourself or for a friend.

If you have any comments, questions, or suggestions, I'd love to hear from you. You can reach me directly via obeythetestinggoat@gmail.com, or on Twitter [@hjwp](https://www.twitter.com/hjwp) (<https://www.twitter.com/hjwp>). You can also check out the website and my blog (<http://www.obeythetestinggoat.com>), and there's a mailing list (<https://groups.google.com/forum/#!forum/obey-the-testing-goat-book>).

I hope you'll enjoy reading this book as much as I enjoyed writing it.

Why I Wrote a Book About Test-Driven Development

“Who are you, why are you writing this book, and why should I read it?” I hear you ask.

I'm still quite early on in my programming career. They say that in any discipline, you go from apprentice, to journeyman, and eventually, sometimes, on to master. I'd say that I'm—at best—a journeyman programmer. But I was lucky enough, early on in my career, to fall in with a bunch of TDD fanatics, and it made such a big impact on my programming that I'm burning to share it with everyone. You might say I have the enthusiasm of a recent convert, and the learning experience is still a recent memory for me, so I hope I can still empathise with beginners.

When I first learned Python (from Mark Pilgrim's excellent *Dive Into Python*), I came across the concept of TDD, and thought “Yes. I can definitely see the sense in that”.

Perhaps you had a similar reaction when you first heard about TDD? It sounds like a really sensible approach, a really good habit to get into—like regularly flossing your teeth or something.

Then came my first big project, and you can guess what happened—there was a client, there were deadlines, there was lots to do, and any good intentions about TDD went straight out of the window.

And, actually, it was fine. I was fine.

At first.

At first I knew I didn't really need TDD because it was a small website, and I could easily test whether things worked by just manually checking it out. Click this link *here*, choose that drop-down item *there*, and *this* should happen. Easy. This whole writing tests thing sounded like it would have taken *ages*, and besides, I fancied myself, from the full height of my three weeks of adult coding experience, as being a pretty good programmer. I could handle it. Easy.

Then came the fearful goddess Complexity. She soon showed me the limits of my experience.

The project grew. Parts of the system started to depend on other parts. I did my best to follow good principles like DRY (Don't Repeat Yourself), but that just led to some pretty dangerous territory. Soon I was playing with multiple inheritance. Class hierarchies 8 levels deep. `eval` statements.

I became scared of making changes to my code. I was no longer sure what depended on what, and what might happen if I changed this code *over here*, oh gosh, I think that bit over there inherits from it—no, it doesn't, it's overridden. Oh, but it depends on that class variable. Right, well, as long as I override the override it should be fine. I'll just check—but checking was getting much harder. There were lots of sections to the site now, and clicking through them all manually was starting to get impractical. Better to leave well enough alone, forget refactoring, just make do.

Soon I had a hideous, ugly mess of code. New development became painful.

Not too long after this, I was lucky enough to get a job with a company called Resolver Systems (now PythonAnywhere (<https://www.pythonanywhere.com>)), where Extreme Programming (XP) was the norm. They introduced me to rigorous TDD.

Although my previous experience had certainly opened my mind to the possible benefits of automated testing, I still dragged my feet at every stage. "I mean, testing in general might be a good idea, but *really*? All these tests? Some of them seem like a total waste of time ... What? Functional tests as *well* as unit tests? Come on, that's overdoing it! And this TDD test/minimal-code-change/test cycle? This is just silly! We don't need all

these baby steps! Come on, we can see what the right answer is, why don't we just skip to the end?"

Believe me, I second-guessed every rule, I suggested every shortcut, I demanded justifications for every seemingly pointless aspect of TDD, and I came out seeing the wisdom of it all. I've lost count of the number of times I've thought "Thanks, tests", as a functional test uncovers a regression we would never have predicted, or a unit test saves me from making a really silly logic error. Psychologically, it's made development a much less stressful process. It produces code that's a pleasure to work with.

So, let me tell you *all* about it!

Aims of This Book

My main aim is to impart a methodology—a way of doing web development, which I think makes for better web apps and happier developers. There's not much point in a book that just covers material you could find by googling, so this book isn't a guide to Python syntax, or a tutorial on web development *per se*. Instead, I hope to teach you how to use TDD to get more reliably to our shared, holy goal: *clean code that works*.

With that said: I will constantly refer to a real practical example, by building a web app from scratch using tools like Django, Selenium, jQuery, and Mock. I'm not assuming any prior knowledge of any of these, so you should come out of the other end of this book with a decent introduction to those tools, as well as the discipline of TDD.

In Extreme Programming we always pair-program, so I've imagined writing this book as if I was pairing with my previous self, having to explain how the tools work and answer questions about why we code in this particular way. So, if I ever take a bit of a patronising tone, it's because I'm not all that smart, and I have to be very patient with myself. And if I ever sound defensive, it's because I'm the kind of annoying person that systematically disagrees with whatever anyone else says, so sometimes it takes a lot of justifying to convince myself of anything.

Outline

I've split this book into three parts.

Part I (Chapters 1–6): The basics

Dives straight into building a simple web app using TDD. We start by writing a functional test (with Selenium), then we go through the basics of Django—models, views, templates—with rigorous unit testing at every stage. I also introduce the Testing Goat.

Part II (Chapters 7–14): Web development essentials

Covers some of the trickier but unavoidable aspects of web development, and shows how testing can help us with them: static files, deployment to production, form data validation, database migrations, and the dreaded JavaScript.

Part III (Chapters 15–20): More advanced topics

Mocking, integrating a third-party authentication system, Ajax, test fixtures, Outside-In TDD, and Continuous Integration (CI).

On to a little housekeeping...

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

Constant width bold

Shows commands or other text that should be typed literally by the user.

Occasionally I will use the symbol:

[...]

to signify that some of the content has been skipped, to shorten long bits of output, or to skip down to a relevant bit.



This element signifies a tip or suggestion.



This element signifies a general note or aside.



This element indicates a warning or caution.

Using Code Examples


Code examples are available at <https://github.com/hjwp/book-example/>; you'll find branches for each chapter there (eg, https://github.com/hjwp/book-example/tree/chapter_03). You'll also find some suggestions on ways of working with this repository at the end of each chapter.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Test-Driven Development with Python* by Harry Percival (O'Reilly). Copyright 2014 Harry Percival, 978-1-449-36482-3."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Safari® Books Online

 *Safari Books Online* is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM

Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

Contacting O'Reilly

If you'd like to get in touch with my beloved publisher with any questions about this book, contact details follow:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

You can also send email to bookquestions@oreilly.com.

You can find errata, examples, and additional information at <http://bit.ly/test-driven-python>.

For more information about books, courses, conferences, and news, see O'Reilly's website at <http://www.oreilly.com>.

Facebook: <http://facebook.com/oreilly>

Twitter: <http://twitter.com/oreillymedia>

YouTube: <http://www.youtube.com/oreillymedia>

Prerequisites and Assumptions

Here's an outline of what I'm assuming about you and what you already know, as well as what software you'll need ready and installed on your computer.

Python 3 and Programming

I've written the book with beginners in mind, but if you're new to programming, I'm assuming that you've already learned the basics of Python. So if you haven't already, do run through a Python beginner's tutorial or get an introductory book like *Dive Into Python* (<http://www.diveintopython.net/>) or *Learn Python the Hard Way* (<http://learnpythonthehardway.org/>), or, just for fun, *Invent Your Own Computer Games with Python* (<http://inventwithpython.com/>), all of which are excellent introductions.

If you're an experienced programmer but new to Python, you should get along just fine. Python is joyously simple to understand.

I'm using *Python 3* for this book. When I wrote it in 2013–14, Python 3 had been around for several years, and the world was just about on the tipping point at which it was the preferred choice. You should be able to follow this book on Mac, Windows, or Linux. Detailed installation instructions for each OS follow.



This book was tested against Python 3.3 and Python 3.4. If you're on 3.2 for any reason, you may find minor differences, so you're best off upgrading if you can.

I wouldn't recommend trying to use Python 2, as the differences are more substantial. You'll still be able to carry across all the lessons you learn in this book if your next project happens to be in Python 2. But spending time figuring out whether the reason your program output looks different from mine is because of Python 2, or because you made an actual mistake, won't be time spent productively.

If you are thinking of using PythonAnywhere (<http://www.pythonanywhere.com>) (the PaaS startup I work for), rather than a locally installed Python, you should go and take a quick look at Appendix A before you get started.

In any case, I expect you to have access to Python, to know how to launch it from a command line (usually with the command `python3`), and to know how to edit a Python file and run it. Again, have a look at the three books I recommended previously if you're in any doubt.



If you already have Python 2 installed, and you're worried that installing Python 3 will break it in some way, don't! Python 3 and 2 can coexist peacefully on the same system, and they each store their packages in totally different locations. You just need to make sure that you have one command to launch Python 3 (`python3`), and another to launch Python 2 (usually, just `python`). Similarly, when we install pip for Python 3, we just make sure that its command (usually `pip3`) is identifiably different from the Python 2 pip.

How HTML Works

I'm also assuming you have a basic grasp of how the web works—what HTML is, what a POST request is, etc. If you're not sure about those, you'll need to find a basic HTML tutorial—there are a few at <http://www.webplatform.org/>. If you can figure out how to create an HTML page on your PC and look at it in your browser, and understand what a form is and how it might work, then you're probably OK.

JavaScript

There's a little bit of JavaScript in the second half of the book. If you don't know JavaScript, don't worry about it until then, and if you find yourself a little confused, I'll recommend a couple of guides at that point.

Required Software Installations

Aside from Python, you'll need:

The Firefox web browser

A quick Google search will get you an installer for whichever platform you're on. Selenium can actually drive any of the major browsers, but Firefox is the easiest to use as an example because it's reliably cross-platform and, as a bonus, is less sold out to corporate interests.

The Git version control system

This is available for any platform, at <http://git-scm.com/>.

The pip Python package management tool

This comes bundled with Python 3.4 (it didn't always used to, this is a big hooray). To make sure we're using the Python3 version of pip, I'll always use `pip3` as the executable in my command-line examples. Depending on your platform, it may be `pip-3.4` or `pip-3.3`. Have a look at the detailed notes for each operating system for more info.

Windows Notes

Windows users can sometimes feel a little neglected, since OS X and Linux make it easy to forget there's a world outside the Unix paradigm. Backslashes as directory separators? Drive letters? What? Still, it is absolutely possible to follow along with this book on Windows. Here are a few tips:

1. When you install Git for Windows, make sure you choose “*Run Git and included Unix tools from the Windows command prompt*”. You'll then get access to a program called “Git Bash”. Use this as your main command prompt and you'll get all the useful GNU command-line tools like `ls`, `touch`, and `grep`, plus forward-slash directory separators.
2. When you install Python 3, make sure you tick the option that says “add python.exe to Path” as in Figure P-1, to make sure you can run Python from the command line.

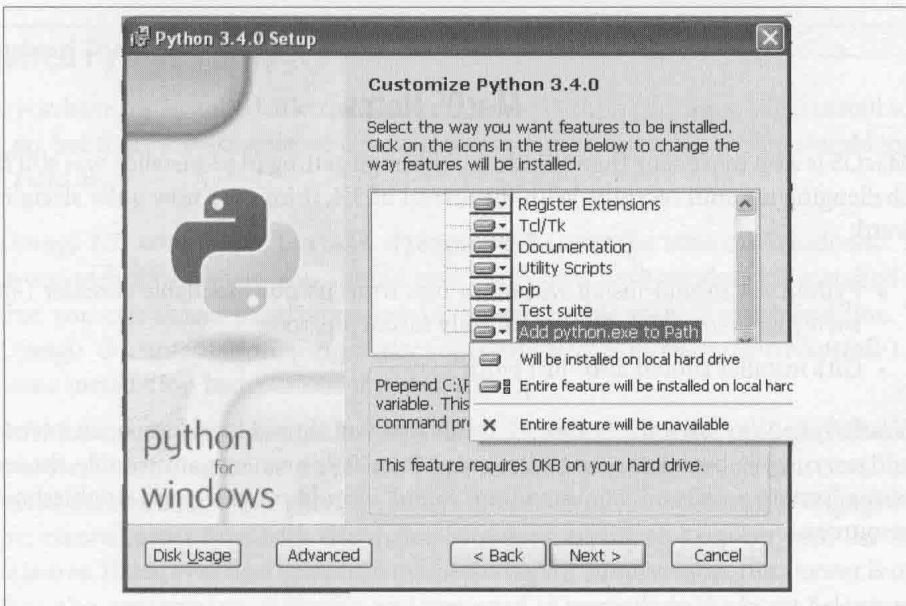


Figure P-1. Add python to the system path from the installer

3. On Windows, Python 3's executable is called `python.exe`, which is exactly the same as Python 2. To avoid any confusion, create a symlink in the Git Bash binaries folder, like this:

```
ln -s /c/Python34/python.exe /bin/python3.exe
```

You may need to right-click Git-Bash and choose “Run as an administrator” for that command to work. Note also that the symlink will only work in Git Bash, not in the regular DOS command prompt.

4. Python 3.4 comes with `pip`, the package management tool. You can check it's installed by doing a `which pip3` from a command line, and it should show you `/c/Python34/Scripts/pip3`.

If, for whatever reason, you're stuck with Python 3.3 and you don't have `pip3`, check <http://www.pip-installer.org/> for installation instructions. At the time of writing, this involved downloading a file and then executing it with `python3 get-pip.py`. *Make sure you use `python3` when you run the setup script.*



The test for all this is that you should be able to go to a Git-Bash command prompt and just run `python3` or `pip3` from any folder.

MacOS Notes

MacOS is a bit more sane than Windows, although getting `pip3` installed was still fairly challenging up until recently. With the arrival of 3.4, things are now quite straightforward:

- Python 3.4 should install without a fuss from its downloadable installer (<http://www.python.org>). It will automatically install `pip`, too.
- Git's installer should also “just work”.

Similarly to Windows, the test for all this is that you should be able to open a terminal and just run `git`, `python3`, or `pip3` from anywhere. If you run into any trouble, the search terms “system path” and “command not found” should provide good troubleshooting resources.



You might also want to check out Homebrew (<http://brew.sh/>). It used to be the only reliable way of installing lots of Unixy tools (including Python 3) on a Mac. Although the Python installer is now fine, you may find it useful in future. It does require you to download all 1.1 GB of Xcode, but that also gives you a C compiler, which is a useful side effect.

Git's Default Editor, and Other Basic Git Config

I'll provide step-by-step instructions for Git, but it may be a good idea to get a bit of configuration done now. For example, when you do your first commit, by default *vi* will pop up, at which point you may have no idea what to do with it. Well, much as *vi* has two modes, you then have two choices. One is to learn some minimal *vi* commands (*press the i key to go into insert mode, type your text, press <Esc> to go back to normal mode, then write the file and quit with :wq<Enter>*). You'll then have joined the great fraternity of people who know this ancient, revered text editor.

Or you can point-blank refuse to be involved in such a ridiculous throwback to the 1970s, and configure Git to use an editor of your choice. Quit *vi* using <Esc> followed by :q!, then change your Git default editor. See the Git documentation on basic Git configuration (<http://git-scm.com/book/en/Customizing-Git-Git-Configuration>).

Required Python Packages

Once you have *pip* installed, it's trivial to install new Python packages. We'll install some as we go, but there are a couple we'll need right from the beginning, so you should install them right away:

- *Django 1.7*, **sudo pip3 install django==1.7** (omit the **sudo** on Windows). This is our web framework. You should make sure you have version 1.7 installed and that you can access the `django-admin.py` executable from a command line. The Django documentation (<https://docs.djangoproject.com/en/1.7/intro/install/>) has some installation instructions if you need help.
- *Selenium*, **sudo pip3 install --upgrade selenium** (omit the **sudo** on Windows), a browser automation tool that we'll use to drive what are called functional tests. Make sure you have the absolute latest version installed. Selenium is engaged in a permanent arms race with the major browsers, trying to keep up with the latest features. If you ever find Selenium misbehaving for some reason, the answer is often that it's a new version of Firefox and you need to upgrade to the latest Selenium ...



Unless you're absolutely sure you know what you're doing, *don't* use a `virtualenv`. We'll start using one later in the book, in Chapter 8.

A Note on IDEs

If you've come from the world of Java or .NET, you may be keen to use an IDE for your Python coding. They have all sorts of useful tools, including VCS integration, and there are some excellent ones out there for Python. I used one myself when I was starting out, and I found it very useful for my first couple of projects.

Can I suggest (and it's only a suggestion) that you *don't* use an IDE, at least for the duration of this tutorial? IDEs are much less necessary in the Python world, and I've written this whole book with the assumption that you're just using a basic text editor and a command line. Sometimes, that's all you have—when you're working on a server for example—so it's always worth learning how to use the basic tools first and understanding how they work. It'll be something you always have, even if you decide to go back to your IDE and all its helpful tools, after you've finished this book.

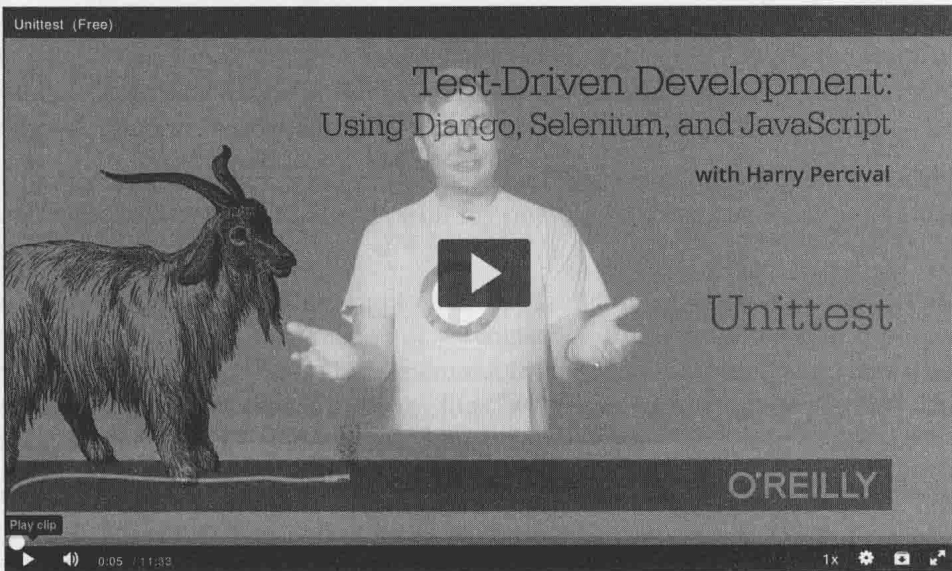


Did these instructions not work for you? Or have you got better ones? Get in touch: obeythetestinggoat@gmail.com!

Companion Video

I've recorded a 10-part video series to accompany this book (<http://oreil.ly/1svTFqB>). It covers the content of Chapters 1-6. If you find you learn well from video-based material, then I encourage you to check it out. Over and above what's in the book, it should give you a feel for what the "flow" of TDD is like, flicking between tests and code, explaining the thought process as we go.

Plus I'm wearing a delightful yellow T-shirt:



Acknowledgments

Lots of people to thank, without whom this book would never have happened, and/or would have been even worse than it is.

Thanks first to “Greg” at \$OTHER_PUBLISHER, who was the first person to encourage me to believe it really could be done. Even though your employers turned out to have overly regressive views on copyright, I’m forever grateful that you believed in me.

Thanks to Michael Foord, another ex-employee of Resolver Systems, for providing the original inspiration by writing a book himself, and thanks for his ongoing support for the project. Thanks also to my boss Giles Thomas, for foolishly allowing another one of his employees to write a book (although I believe he’s now changed the standard employment contract to say “no books”). Thanks also for your ongoing wisdom and for setting me off on the testing path.

Thanks to my other colleagues, Glenn Jones and Hansel Dunlop, for being invaluable sounding boards, and your patience with my one-track record conversation over the last year.

Thanks to my wife Clementine, and to both my families, without whose support and patience I would never have made it. I apologise for all the time spent with nose in computer on what should have been memorable family occasions. I had no idea when I set out what the book would do to my life (“write it in my spare time you say? That sounds reasonable...”). I couldn’t have done it without you.

Thanks to my tech reviewers, Jonathan Hartley, Nicholas Tollrvey, and Emily Bache, for your encouragements and invaluable feedback. Especially Emily, who actually conscientiously read every single chapter. Partial credit to Nick and Jon, but that should still be read as eternal gratitude. Having y’all around made the whole thing less of a lonely endeavour. Without all of you the book would have been little more than the nonsensical ramblings of an idiot.

Thanks to everyone else who’s given up some of their time to give some feedback on the book, out of nothing more than the goodness of their heart: Gary Bernhardt, Mark