Table of Contents

Preface

This book has been my attempt to share with the world the journey I took 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 your time on. Who knows; perhaps once you get to the end, you'll decide it's good enough to buy a physical copy for yourself or 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 Mastodon @hjwp (https://fosstodon.org/@hjwp). You can also check out the website and my blog (https://www.obeythetestinggoat.com).

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 have you written this book, and why should I read it?" I hear you ask.

I was lucky enough early on in my career to fall in with a bunch of test-driven development (TDD) fanatics, and it made such a big impact on my programming that I was burning to share it with everyone. You might say I had the enthusiasm of a recent convert, and the learning experience was still a recent memory for me, so that's what led to the first edition, back in 2014.

When I first learned Python (from Mark Pilgrim's excellent *Dive Into Python* (https://diveintopython3.net)), 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 seemed like a really sensible approach, a really good habit to get into—like regularly flossing your teeth.

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 thought I didn't really need TDD because the website was small, 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 eight 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 for 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) (https://martinfowler.com/bliki/ExtremeProgramming.html) was the norm. The people there 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 still came out seeing the wisdom of it all. I've lost count of the number of times I've thought, "Thanks, tests",[1] 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, nor 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 mocks. I'm not assuming any prior knowledge of any of these, so you should come out 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 four parts.

[part1] (Chapters 1 to 8): The Basics of TDD and Django

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

[part2] (Chapters 9 to 12): Going to Production

These chapters are all about deploying your web app to an actual server. We discuss how our tests, and the TDD practice of working incrementally, can take a lot of the pain and risk out of what is normally quite a fraught process.

[part3] (Chapters 13 to 16): Forms and Validation

Here, we get into some of the details of the Django Forms framework, implementing validation, and data integrity using database constraints. We discuss using tests to explore unfamiliar APIs, and the limits of frameworks.

[part4] (Chapters 17 to 27): Advanced Topics in Testing

Covers some of the more advanced topics in TDD, including spiking (where we relax the rules of TDD temporarily), mocking, working outside-in, and continuous integration (CI).

Now, onto 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 and 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 section. You will also encounter the following callouts:

This element signifies a tip or suggestion.

This element signifies a general note or aside.

This element indicates a warning or caution.

Submitting Errata

Spotted a mistake or a typo? The sources for this book are available on GitHub, and I'm always very happy to receive issues and pull requests: https://github.com/hjwp/Book-TDD-Web-Dev-Python.

Using Code Examples

Code examples are available at https://github.com/hjwp/book-example/; you'll find branches for each chapter there (e.g., https://github.com/hjwp/book-example/tree/chapter_03_unit_test_first_view). You can find a full list and some suggestions on ways of working with this repository in [appendix_github_links].

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 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*, 3rd edition, by Harry J.W. Percival (O'Reilly). Copyright 2025 Harry Percival, 978-1-098-14871-3".

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*.

O'Reilly Online Learning

For more than 40 years, *O'Reilly Media* (https://oreilly.com) has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *https://oreilly.com* (https://oreilly.com).

How to Contact Us

Please address comments and questions concerning this book to the publisher:

- O'Reilly Media, Inc.

- 141 Stony Circle, Suite 195
- Santa Rosa, CA 95401
- 800-889-8969 (in the United States or Canada)
- 707-827-7019 (international or local)
- 707-829-0104 (fax)
- *support@oreilly.com*
- *https://oreilly.com/about/contact.html* (https://oreilly.com/about/contact.html)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at https://oreil.ly/TDD-with-python-3e.

For news and information about our books and courses, visit https://oreilly.com.

Find us on LinkedIn: https://linkedin.com/company/oreilly-media.

Watch us on YouTube: https://youtube.com/oreillymedia.

Companion Video

I’ve recorded a 10-part video series to accompany this book (https://learning.oreilly.com/videos/test-driven-development/9781491919163).[2] It covers the content of [part1]. If you find that 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 and explaining the thought process as we go.

Plus, I’m wearing a delightful yellow t-shirt.

License for the Free Edition

If you're reading the free edition of this book hosted at http://www.obeythetestinggoat.com, then the license is Creative Commons Attribution-NonCommercial-NoDerivatives (https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode).[3] I want to thank O'Reilly for its fantastic attitude towards licensing; most publishers aren't so forward-thinking.

I see this as a "try before you buy" scheme really. If you're reading this book it's for professional reasons, so I hope that if you like it, you'll buy a copy—if not for yourself, then for a friend! O'Reilly has been great, it deserves your support. You'll find links to buy back on the home page (https://www.obeythetestinggoat.com).

1. Thesis (https://oreil.ly/LGP3g).
2. The video has not been updated for the third edition, but the content is all mostly the same.
3. The no-derivs clause is there because O'Reilly wants to maintain some control over derivative works, but it often does grant permissions for things, so don't hesitate to get in touch if you want to build something based on this book.

Comments

ALSO ON **OBEY THE TESTING GOAT!**

Obey the Testing Goat!	**Source Code Examples**	**Making our App Production-Ready**	**Getting A For Deplo**
8 years ago • 12 comments	9 years ago • 2 comments	2 years ago • 8 comments	a year ago • 1
I'm still not great at selfies... Here you get two for the price of one tho. The second …	Try not to sneak a peek at the answers unless you're really, really stuck. Like I said at …	Static files aren't the same kind of things as the dynamic content that comes from …	With a PaaS your own se renting a "se

1 Comment Kgotso Koete ▾

Join the discussion…

♡ 2 **Share** **Best** **Newest** **Oldest**

S **Sasha Pazyuk**
9 years ago

Please make corrections:
(Chapters 1–6): The basics -> (Chapters 1–7): The basics
(Chapters 7–14): Web development essentials -> (Chapters 8–16): Web development essentials
(Chapters 15–20): More advanced topics -> (Chapters 17–26): More advanced topics

1 1 Reply Share ›

Subscribe **Privacy** **Do Not Sell My Data**