Table of Contents

Praise for *Test-Driven Development with Python*

Preface

Preface to the Third Edition: TDD in the Age of AI

Prerequisites and Assumptions

- Python 3 and Programming
- How HTML Works
- Django
- JavaScript
- Required Software Installations
- Setting Up Your Virtualenv

Acknowledgments

The Basics of TDD and Django

1. Getting Django Set Up Using a Functional Test
2. Extending Our Functional Test Using the unittest Module
3. Testing a Simple Home Page with Unit Tests
4. What Are We Doing with All These Tests? (And, Refactoring)
5. Saving User Input: Testing the Database
6. Improving Functional Tests: Ensuring Isolation and Removing Magic Sleeps
7. Working Incrementally
8. Prettification: Layout and Styling, and What to Test About It

Going to Production

9. Containerization aka Docker
10. Making Our App Production-Ready
11. Getting a Server Ready for Deployment
12. Infrastructure as Code: Automated Deployments with Ansible

Forms and Validation

13. Splitting Our Tests into Multiple Files, and a Generic Wait Helper
14. Validation at the Database Layer
15. A Simple Form
16. More Advanced Forms

More Advanced Topics in Testing

17. A Gentle Excursion into JavaScript
18. Deploying Our New Code
19. User Authentication, Spiking, and De-Spiking

20. Using Mocks to Test External Dependencies
21. Using Mocks for Test Isolation
22. Test Fixtures and a Decorator for Explicit Waits
23. Debugging and Testing Server Issues
24. Finishing "My Lists": Outside-In TDD
25. CI: Continuous Integration
26. The Token Social Bit, the Page Pattern, and an Exercise for the Reader
27. Fast Tests, Slow Tests, and Hot Lava
Appendix A: Obey the Testing Goat!
Appendix B: The Subtleties of Functionally Testing External Dependencies
Appendix C: Continuous Deployment (CD)
Appendix D: Behaviour-Driven Development (BDD) Tools
Appendix E: Test Isolation, and "Listening to Your Tests"
Appendix F: Building a REST API: JSON, Ajax, and Mocking with JavaScript
Appendix G: Cheat Sheet
Appendix H: What to Do Next
Appendix I: Source Code Examples
Appendix J: Bibliography

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 tried to write this 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 *The Quick Python Book* (https://www.manning.com/books/the-quick-python-book-third-edition) or *Think Python* (https://oreil.ly/think-python-3e), or (just for fun) *Invent Your Own Computer Games with Python* (https://inventwithpython.com/invent4thed)—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.

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 3.14. If you're on an earlier version, you will find minor differences in the way things look in my command output listings (tracebacks won't have the `^^^^^^` carets marking error locations, for example), so you're best off upgrading, ideally, if you can.

In any case, I expect you to have access to Python, to know how to launch it from a command line, 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.

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, and so on. If you're not sure about those, you'll need to find a basic HTML tutorial; there are a few at https://developer.mozilla.org/Learn_web_development. 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.

Django

The book uses the Django framework, which is (probably) the most well-established web framework in the Python world. I've written this book assuming that the reader has no prior knowledge of Django, but if you're new to Python *and* new to web development *and* new to testing, you may occasionally find that there's just one too many topics and sets of concepts to try and take on board. If that's the case, I recommend taking a break from the book, and taking a look at a Django tutorial. DjangoGirls (https://tutorial.djangogirls.org) is the best, most beginner-friendly tutorial I know of. Django's official tutorial (https://docs.djangoproject.com/en/5.2/intro/tutorial01) is also excellent for more experienced programmers.

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.

Read on for installation instructions.

Required Software Installations

Aside from Python, you'll need:

The Firefox web browser

Selenium can actually drive any of the major browsers, but I chose Firefox because it's the least in hock to corporate interests.

The Git version control system

This is available for any platform, at http://git-scm.com. On Windows, it comes with the *bash* command line, which is needed for the book. See Windows Notes.

A virtualenv with Python 3.14, Django 5.2, and Selenium 4 in it

Python's virtualenv and pip tools now come bundled with Python (they didn't always used to, so this is a big hooray). Detailed instructions for preparing your virtualenv follow.

MacOS Notes

MacOS installations for Python and Git are relatively straightforward:

- Python 3.14 should install without a fuss from its downloadable installer (https://www.python.org). It will automatically install `pip`, too.
- Git's installer should also "just work".
- You might also want to check out Homebrew (http://brew.sh). It's a fairly reliable way of installing common Unix tools on a Mac.[1] Although the normal Python installer is now fine, you may find Homebrew 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.
- If you want to run multiple different versions of Python on your Mac, tools like uv (https://docs.astral.sh/uv/guides/install-python) or pyenv (https://github.com/pyenv/pyenv) can help. The downside is that each is one more fiddly tool to have to learn. But the key is to make sure, when creating your virtualenv, that you use the right version of Python. From then on, you shouldn't need to worry about it, at least not when following this book.

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

resources.

Linux Notes

If you're on Linux, I'm assuming you're already a glutton for punishment, so you don't need detailed installation instructions. But in brief, if Python 3.14 isn't available directly from your package manager, you can try the following:

- On Ubuntu, you can install the deadsnakes PPA (https://oreil.ly/fHrpG). Make sure you `apt install python3.14-venv` as well as just `python3.14` to un-break the default Debian version of Python.
- Alternatively, uv (https://docs.astral.sh/uv/guides/install-python) and pyenv (https://github.com/pyenv/pyenv) both let you manage multiple Python versions on the same machine, but it is yet another thing to have to learn and remember.
- Alternatively, compiling Python from source is actually surprisingly easy!

However you install it, make sure you can run Python 3.14 from a terminal.

Windows Notes

Windows users can sometimes feel a little neglected in the open source world. As macOS and Linux are so prevalent, it's 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:

- When you install Git for Windows, it will include "Git Bash". Use this as your main command prompt throughout the book, and you'll get all the useful GNU command-line tools like `ls`, `touch`, and `grep`, plus forward-slash directory separators.
- During the Git installation, you'll get the option to choose the default editor used by Git. Unless you're already a Vim user (or are desperate to learn), I'd suggest using a more familiar editor—even just Notepad! See Choose a nice default editor for Git.
- Also in the Git installer, choose "Use Windows' default console"; otherwise, Python won't work properly in the Git Bash window.

- When you install Python, tick the option that says "Add python.exe to PATH" as in Add Python to the system path from the installer, so that you can easily run Python from the command line.

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

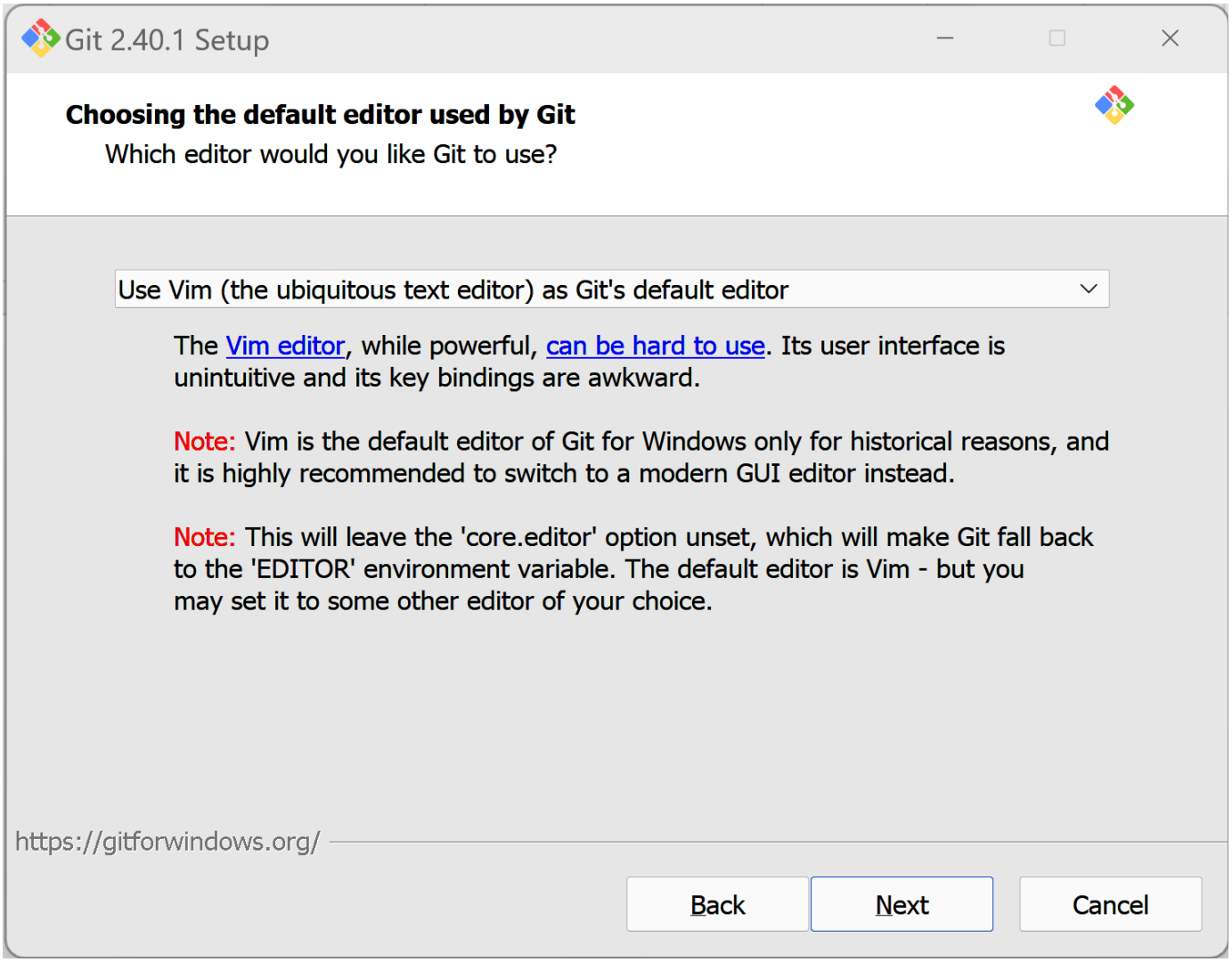

Figure 1. Choose a nice default editor for Git

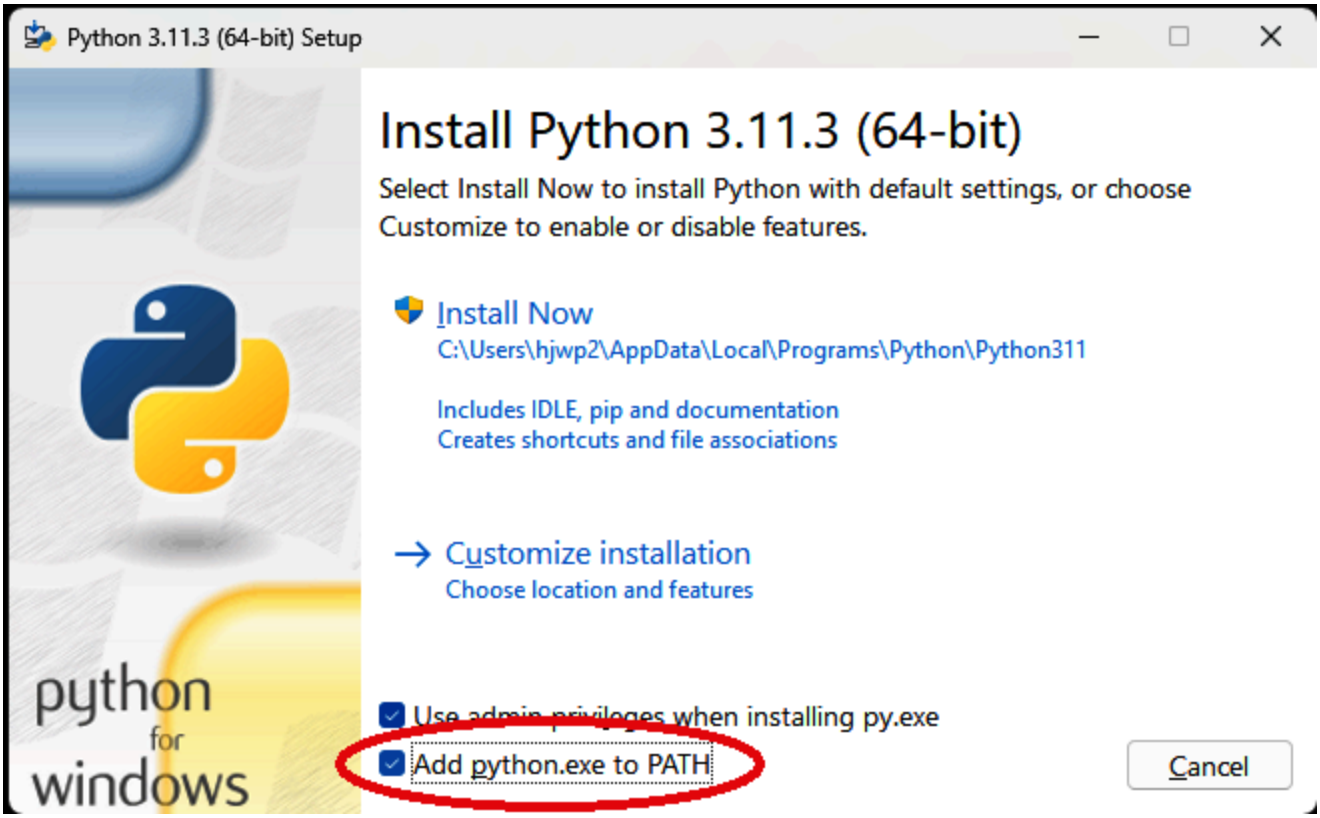

Figure 2. Add Python to the system path from the installer

Installing Firefox

Firefox is available as a download for Windows and macOS from *firefox.com* (https://www.firefox.com/). On Linux, you probably already have it installed, but otherwise your package manager will have it.

Make sure you have the latest version, so that the "geckodriver" browser automation module is available.

Setting Up Your Virtualenv

A Python virtualenv (short for virtual environment) is how you set up your environment for different Python projects. It enables you to use different packages (e.g., different versions of Django, and even different versions of Python) in each project. And because you're not installing things system-wide, it means you don't need root permissions.

Let's create a virtualenv. I'm assuming you're working in a folder called *goat-book*, but you can name your work folder whatever you like. Stick to the name ".venv" for the virtualenv, though:

on Windows:

```
$ cd goat-book
$ py -3.14 -m venv .venv
```

On Windows, the `py` executable is a shortcut for different Python versions. On Mac or Linux, we use `python3.14`:

on Mac/Linux:

```
$ cd goat-book
$ python3.14 -m venv .venv
```

Activating and Deactivating the Virtualenv

Whenever you're working through the book, you'll want to make sure your virtualenv has been "activated". You can always tell when your virtualenv is active because, in your prompt, you'll see `(.venv)` in parentheses. But you can also check by running `which python` to check whether Python is currently the system-installed one or the virtualenv one.

The command to activate the virtualenv is `source .venv/Scripts/activate` on Windows and `source .venv/bin/activate` on Mac/Linux. The command to deactivate is just `deactivate`.

Try it out like this, on Windows:

```
$ source .venv/Scripts/activate
(.venv)$
(.venv)$ which python
/C/Users/harry/goat-book/.venv/Scripts/python
(.venv)$ deactivate
$
$ which python
/c/Users/harry/AppData/Local/Programs/Python/Python312-32/python
```

Or like this, on Mac/Linux:

```
$ source .venv/bin/activate
(.venv)$
(.venv)$ which python
/home/harry/goat-book/.venv/bin/python
(.venv)$ deactivate
$
$ which python
/usr/bin/python
```

Always make sure your virtualenv is active when working on the book. Look out for the `(.venv)` in your prompt, or run `which python` to check.

Virtualenvs and IDEs

If you're using an IDE like PyCharm or Visual Studio Code, you should be able to configure them to use the virtualenv as the default Python interpreter for the project.

You should then be able to launch a terminal inside the IDE with the virtualenv already activated.

Installing Django and Selenium

We'll install Django 5.2 and the latest Selenium.[2] Remember to make sure your virtualenv is active first!

```
(.venv) $ pip install "django<6" "selenium"
Collecting django<6
  Downloading Django-5.2.3-py3-none-any.whl (8.0 MB)
     ---------------------------------------- 8.1/8.1 MB 7.6 MB/s eta 0:00:00
Collecting selenium
  Downloading selenium-4.24.0-py3-none-any.whl (6.5 MB)
     ---------------------------------------- 6.5/6.5 MB 6.3 MB/s eta 0:00:00
Collecting asgiref>=3.8.1 (from django<6)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django<6)Collecting sqlparse>=0.3.1 (from
django<6)
  [...]
Installing collected packages: sortedcontainers, websocket-client, urllib3,
typing_extensions, sqlparse, sniffio, pysocks, idna, h11, certifi, attrs,
asgiref, wsproto, outcome, django, trio, trio-websocket, selenium
Successfully installed asgiref-3.8.1 attrs-25.3.0 certifi-2025.4.26
django-5.2.3 [...]
selenium-4.32.0 [...]
```

Check that it works:

```
(.venv) $ python -c "from selenium import webdriver; webdriver.Firefox()"
```

This should pop open a Firefox web browser, which you'll then need to close.

If you see an error, you'll need to debug it before you go further. On Linux/Ubuntu, I ran into a bug (https://github.com/mozilla/geckodriver/issues/2010), which needs to be fixed by setting an environment variable called `TMPDIR`.

Some Error Messages You're Likely to See When You Inevitably Fail to Activate Your Virtualenv

If you're new to virtualenvs—or even if you're not, to be honest—at some point you're *guaranteed* to forget to activate it, and then you'll be staring at an error message. Happens to me all the time. Here are some of the things to look out for:

```
ModuleNotFoundError: No module named 'selenium'
```

Or:

```
ModuleNotFoundError: No module named 'django'
[...]
ImportError: Couldn't import Django. Are you sure it's installed and available
on your PYTHONPATH environment variable? Did you forget to activate a virtual
environment?
```

As always, look out for that `(.venv)` in your command prompt, and a quick `source .venv/Scripts/activate` or `source .venv/bin/activate` is probably what you need to get it working again.

Here's another, for good measure:

```
bash: .venv/Scripts/activate: No such file or directory
```

This means you're not currently in the right directory for working on the project. Try a `cd goat-book`, or similar.

Alternatively, if you're sure you're in the right place, you may have run into a bug from an older version of Python, where it wouldn't install an activate script that was compatible with Git Bash. Reinstall Python 3, and make sure you have version 3.6.3 or later, and then delete and re-create your virtualenv.

If you see something like this, it's probably the same issue and you need to upgrade Python:

```
bash: @echo: command not found
bash: .venv/Scripts/activate.bat: line 4:
      syntax error near unexpected token `(
bash: .venv/Scripts/activate.bat: line 4: `if not defined PROMPT ('
```

Final one! Consider this:

```
'source' is not recognized as an internal or external command,
operable program or batch file.
```

If you see this, it's because you've launched the default Windows command prompt, `cmd`, instead of Git Bash. Close it and open the latter.

On Anaconda

Anaconda is another tool for managing different Python environments. It's particularly popular on Windows and for scientific computing, where it can be hard to get some of the compiled libraries to install.

In the world of web programming, it's much less necessary, so *I recommend you do not use Anaconda for this book*.

Happy coding!

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

1. I wouldn't recommend installing Firefox via Homebrew though: `brew` puts the Firefox binary in a strange location, and it confuses Selenium. You can work around it, but it's simpler to just install Firefox in the normal way.
2. You might be wondering why I'm not mentioning a specific version of Selenium. It's because Selenium is constantly being updated to keep up with changes in web browsers, and as we can't really pin our browser to a specific version, we're best off using the latest Selenium. It was version 4.24 at the time of writing.

Comments

Join the discussion…

6 Share

Best Newest Oldest

James Evans

8 years ago

Something to be aware of: make sure that there are no spaces in the path to your working directory as this can make your virtualenv screwy. I had something like the following happen:

(virtualenv) $ pip install "django<1.12" "selenium<4"
bash: /home/james/Work Folder/project/virtualenv/bin/pip: No such file or directory

Deleting the virtualenv, then renaming the "Work Folder" folder to remove the space, then re-running "python3.6 -m venv virtualenv" fixed the problem

1 0 Reply Share ›

das-g

7 years ago

To see whether your versions of Firefox, Selenium and geckodriver are compatible to each other, refer to https://firefox-source-docs...

7 0 Reply Share ›