**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 [*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 the first edition 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 3.6. If you’re on an earlier version, you will find minor differences (the f-string syntax, for example), 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 [[appendix1]](http://www.obeythetestinggoat.com/book/appendix_I_PythonAnywhere.html) 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, 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, particularly if you’re using a virtualenv, which we will be. |

**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 <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.

**Django**

The book uses the Django framework, which is (probably) the most well-established web framework in the Python world. I’ve written the 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. The [official tutorial](https://docs.djangoproject.com/en/1.11/intro/tutorial01/) is also excellent for more experienced programmers.

Read on for instructions on installing Django.

**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.

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.

**Required Software Installations**

Aside from Python, you’ll need:

**The Firefox web browser**

Selenium can actually drive any of the major browsers, but Firefox is the best 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/>. On Windows, this comes with the *Bash* command line, which is needed for the book.

**A virtualenv with Python 3, Django 1.11, and Selenium 3 in it**

Python’s virtualenv and pip tools now come bundled with Python 3.4+ (they didn’t always used to, so this is a big hooray). Detailed instructions for preparing your virtualenv follow.

**Geckodriver**

This is the driver that will let us remotely control Firefox via Selenium. I’ll point to a download link in [Installing Firefox and Geckodriver](http://www.obeythetestinggoat.com/book/pre-requisite-installations.html#firefox_gecko).

Windows Notes

Windows users can sometimes feel a little neglected in the open source world, since macOS and Linux are so prevalent, making 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. Also in the Git installer, choose **"Use Windows' default console"**; otherwise, Python won’t work properly in the Git-Bash window.
3. When you install Python 3, unless you already have Python 2 and want to keep it as your default, tick the option that says **"Add Python 3.6 to PATH"** as in [Add Python to the system path from the installer](http://www.obeythetestinggoat.com/book/pre-requisite-installations.html#add-python-to-path), so that you can easily run Python from the command line.

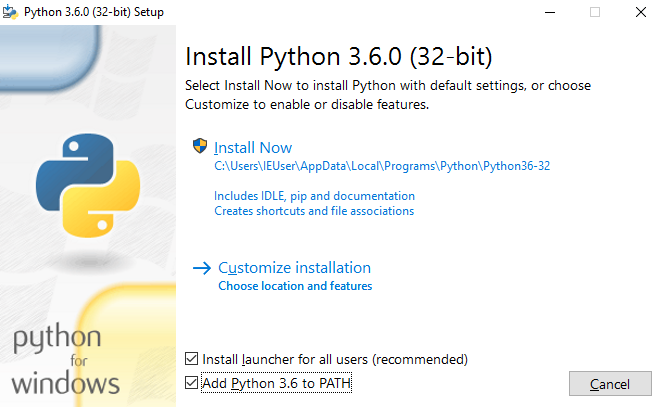


Figure 1. Add Python to the system path from the installer

|  |  |
| --- | --- |
|  | 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. |

MacOS Notes

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

* Python 3.6 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 pip 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.[[1](http://www.obeythetestinggoat.com/book/pre-requisite-installations.html#_footnote_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. |

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 3.6 isn’t available directly from your package manager:

* On Ubuntu I recommend the [Deasnakes PPA](https://launchpad.net/~fkrull/+archive/ubuntu/deadsnakes). Make sure you apt install python3.6-venv as well as just python3.6 to un-break the default Debian version of Python.
* Alternatively, compiling Python 3.6 from source is actually surprisingly easy!

**Git’s Default Editor, and Other Basic Git Config**

>RKW: note I selecte nano

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).

**Installing Firefox and Geckodriver**

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

Geckodriver is available from <https://github.com/mozilla/geckodriver/releases>. You need to download and extract it and put it somewhere on your system path.

* For Windows, you can just put it in the same folder as your code for this book, or if you put it in your Python *Scripts* folder it’ll be available for other projects.

>RKW: Added geckodriver to virtual env envSuperlists for note and VS2017 see below for explenations.

* For macOS or Linux, one convenient place to put it is */usr/local/bin* (you’ll need sudo for this).

To test that you’ve got this working, open up a Bash console and you should be able to run:

$ **geckodriver --version**

geckodriver 0.19.1

The source code of this program is available at

https://github.com/mozilla/geckodriver.

This program is subject to the terms of the Mozilla Public License 2.0.

You can obtain a copy of the license at https://mozilla.org/MPL/2.0/.

**Setting Up Your Virtualenv**

A Python virtualenv (short for virtual environment) is how you set up your environment for different Python projects. It allows 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 Python 3 virtualenv called "superlists"[[2](http://www.obeythetestinggoat.com/book/pre-requisite-installations.html#_footnote_2)] I’m assuming you’re working in a folder called **python-tdd-book**, but you can name your work folder whatever you like. Stick to the name "virtualenv" for the virtualenv though.

>RKW: note

I use this to test the proposed set-up of the Harry’s TDD book.

Create Directory: C:\Users\Public\devTddMilPub\note

Open GIT Bash: R-Click on note directory, which is empty

Type bash: **python -m venv venvSuperlists** # set-up venv

Type bash: **source venvSuperlists/Scripts/activate** # activates venv

Type bash: **which python**  # test venv

Type bash: **deactivate**

Type bash: **which python**  # test venv

This adds 45 MB to the note directory.

>RKW: vs2017

How can I use VS2007 to develop Django apps, Python scripts, Data Science ?

How I can I use the know how to apply in Life Science ?

What learn from this experience for the Devlopment in the JAVA world ?

**HYPER-V Windows Enterprise Admin account**

Python, GIT, VS2007 without python or anaconda, gecko driver, mozilla

Neues Projekt > Python > Web > LeerProjDjangoVS

C:\Users\Public\devTddMilPub\vs2017\

mappeProjTDD

Can create project with Python installed by myself into \Progrmme\Python3

Can create a second project: R-CLICK on mapperProjTDD > Select Project Type ….

Creation of project does not affect python path in GIT-BASH

**MilRai**

Python and anaconda installed in VS20017 shared directory

Neues Projekt > Python > Web > LeerProjDjangoVS

C:\Users\Public\devTddMilPub\vs2017\

mappeProjTDD

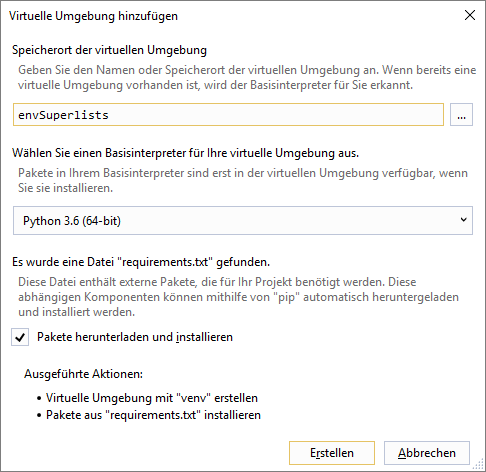
Creates envSuperlists

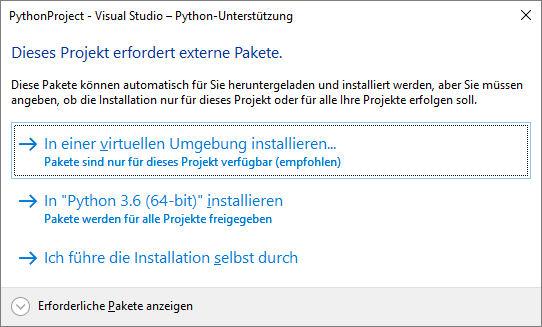
Can activate envSuperlists

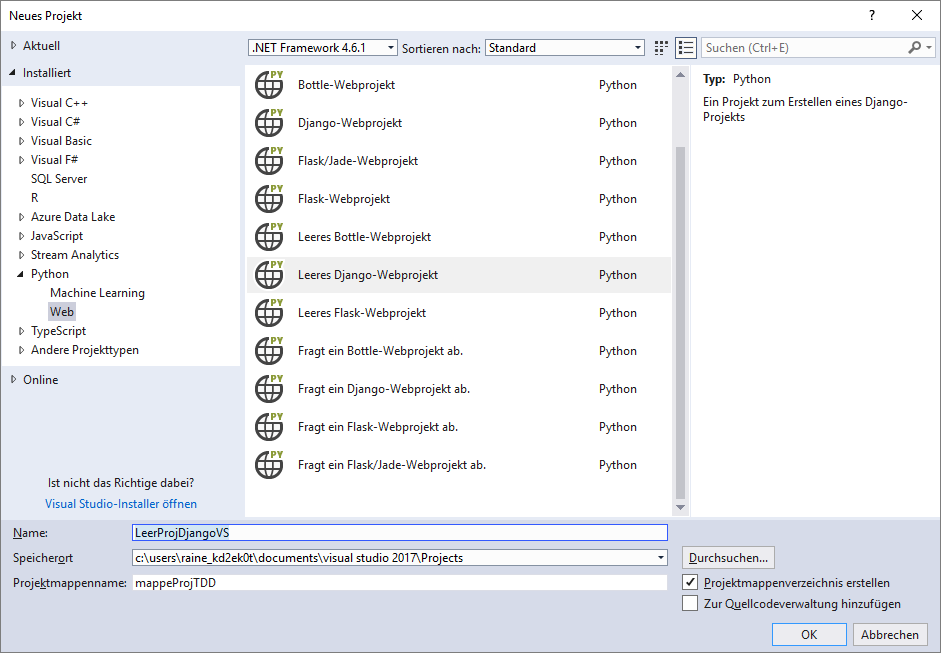
Can run Django server with python manage.py runserver.

Can run server with VS2017.

FIG.: Settings to start a new Python project in VS2017







on Windows:

$ **cd python-tdd-book**

$ **py -3.6 -m venv virtualenv**

On Windows the py executable is a shortcut for different Python versions. On Mac or Linux we use python3.6:

on Mac/Linux:

$ **cd python-tdd-book**

$ **python3.6 -m venv virtualenv**

**Activating and Deactivating the Virtualenv**

Whenever you work on the book, you’ll want to make sure your virtualenv has been "activated". You can always tell when your virtualenv is active because you’ll see (virtualenv) in parentheses, in your prompt. 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 virtualenv/Scripts/activate on Windows and source virtualenv/bin/activate on Mac/Linux. The command to deactivate is just deactivate.

Try it out like this:

on Windows

$ **source virtualenv/Scripts/activate**

(virtualenv)$

(virtualenv)$ **which python**

/C/Users/harry/python-tdd-book/virtualenv/Scripts/python

(virtualenv)$ **deactivate**

$

$ **which python**

/c/Users/harry/AppData/Local/Programs/Python/Python36-32/python

on Mac/Linux

$ **source virtualenv/bin/activate**

(virtualenv)$

(virtualenv)$ **which python**

/home/myusername/python-tdd-book/virtualenv/bin/python

(virtualenv)$ **deactivate**

$

$ **which python**

/usr/bin/python

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

Activate not working on Windows?

If you see an error like this:

bash: virtualenv/Scripts/activate: No such file or directory

First, double-check you’re in the right folder. Assuming you are, or if you see an error like this:

bash: @echo: command not found

bash: virtualenv/Scripts/activate.bat: line 4: syntax error near unexpected token `(

bash: virtualenv/Scripts/activate.bat: line 4: `if not defined PROMPT ('

Then you’ve probably run into a old bug where Python wouldn’t install an activate script that was compatible with Git-Bash. Reinstall the latest Python 3, making sure you have 3.6.3 or later, then delete and re-create your virtualenv.

**Installing Django and Selenium**

We’ll install Django 1.11 and the latest Selenium, Selenium 3.

Remember to make sure your virtualenv is active first!

>RKW: VS2017

Mappe-Exp > R-CLICK on envSuperlists > selenium<4

(virtualenv) $ **pip install "django<1.12" "selenium<4"**

Collecting django==1.11.7

Using cached Django-1.11.7-py2.py3-none-any.whl

Collecting selenium<4

Using cached selenium-3.7.0-py2.py3-none-any.whl

Installing collected packages: django, selenium

Successfully installed django-1.11.7 selenium-3.7.0

**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:

ImportError: No module named selenium

Or:

ImportError: No module named django.core.management

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

Here’s a couple more, for good measure:

bash: virtualenv/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 tdd-python-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, you need to upgrade Python:

bash: @echo: command not found

bash: virtualenv/Scripts/activate.bat: line 4: syntax error near unexpected token `(

bash: virtualenv/Scripts/activate.bat: line 4: `if not defined PROMPT ('

Final one! If you see this:

'source' is not recognized as an internal or external command,

operable program or batch file.

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**.

Apart from anything else I don’t know enough about it to help you debug any problems with it if they occur!

Happy coding!

|  |  |
| --- | --- |
|  | Did these instructions not work for you? Or have you got better ones? Get in touch: [obeythetestinggoat@gmail.com](mailto:obeythetestinggoat@gmail.com)! |

[1](http://www.obeythetestinggoat.com/book/pre-requisite-installations.html#_footnoteref_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](http://www.obeythetestinggoat.com/book/pre-requisite-installations.html#_footnoteref_2). Why superlists, I hear you ask? No spoilers! You’ll find out in the next chapter.

Last updated 2017-11-21 06:25:12 GMT