Testing in Python

Robust Automation for Professionals

Noah Gift & Alfredo Deza

Testing in Python

Noah Gift and Alfredo Deza

This book is for sale at http://leanpub.com/testinginpython

This version was published on 2020-06-26

Contents

Introduction **1**
Why test at all? 1
Leveling up from simple scripts to robust implementations 1
Python, Pytest, Tox supported versions in this book 1
About the cover 1

Chapter 1: Configuring the environment **2**
Setting up and using Git 2
Setting up and using Virtualenv 3
Installing packages and dependencies 4
Setup Visual Code code 5
Setup and use Vim 8
Setup Makefile 13
Setup and Use ZSH/Bash 16
Using Cloud-based development environments 16

Chapter 2: Testing Conventions **18**
Directories 18
Files 18
Functions, Classes, and test methods 18
Special test class methods 18
Good naming patterns 19

Chapter 3: Introduction to Pytest **20**
The most simple test possible 20
Why is Pytest important? 20
The power of `assert` 20
Pytest vs. Unittest 20

Chapter 4: Test Classes **21**
Setting up and teardown of xunit-style tests 21

Chapter 5: Reporting **22**
PyTest Reporting 22
Code Quality 22

Linting . . . 23
Code Formatting with Python Black . . . 23

Chapter 6: Debugging with Pytest . . . 24
How to debug code . . . 24
Using a debugger . . . 24
Python Debugger (PDB) integration . . . 24

Chapter 7: Pytest fixtures and plugins . . . 25
What are fixtures? . . . 25
Creating new fixtures . . . 25
Built-in Fixtures . . . 25
Advanced Fixture usage . . . 25
Parametrizing . . . 26

Chapter 8: Monkeypatching . . . 27
Why and when to monkeypatch? . . . 27
monkeypatching is hard . . . 27
The simplest monkeypatching . . . 27
Automatic and global monkeypatching . . . 27
Other patching . . . 27
When not to monkeypatch . . . 27

Chapter 9: Testing matrix with Tox . . . 29
Testing different Python versions . . . 29
Expanding the testing matrix . . . 29
Linting and other validations . . . 29

Chapter 10: Continuous Integration and Continuous Delivery . . . 30
What is Continuous Integration and Continuous Delivery and Why Do They Matter? . . . 30
Jenkins . . . 30
CircleCI . . . 30
GCP Cloud Build . . . 31
Continuous Delivery for Hugo Static Site from Zero using AWS Code Pipeline . . . 31
Github Actions . . . 32

Chapter 11: Case Studies and War Stories . . . 33
Testing Click Commandline Tools . . . 33
War Story: The Health Check that wasn't wrong . . . 33
War Story: The Nine Circles of Hell While Parsing XML . . . 33
War Story: The Mysterious Vanishing Servers . . . 33

Chapter 12: Essays . . . 34
Writing clean, testable, high quality code in Python . . . 34
Increase reliability in data science and machine learning projects with CircleCI . . . 35

Data science project quality 35

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Why test at all?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Leveling up from simple scripts to robust implementations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Python, Pytest, Tox supported versions in this book

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

About the cover

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 1: Configuring the environment

Noah Gift

Creating a local development environment that is simple, repeatable, and powerful is an essential first step to developing a testable software project. This technique is also a skill that translates to any future software development project.

Setting up and using Git

Source control is a mandatory part of any professional software project. There are many types of version control available including subversion[1], mercurial[2] and git[3]. The most popular of these is `git`.

Git can be standalone, or as part of a hosted solution. Two popular solutions for hosting git projects include: Github[4] and Gitlab[5].

Setup and use Github

To set up and use Github, you need a Github account and internet access. The minimal steps to start are:

1. Create a repository, for example, `hello`.
2. Add an SSH key to your Github account[6].
3. Clone the repository locally, for example:

Clone a repo

```
git clone git@github.com:paiml/hello.git
```

4. Create a change and push it. This process would be an example of a good first change (inside the cloned repo).

[1] https://subversion.apache.org/
[2] https://www.mercurial-scm.org/
[3] https://git-scm.com/
[4] https://github.com/
[5] https://about.gitlab.com/
[6] https://help.github.com/en/github/authenticating-to-github/adding-a-new-ssh-key-to-your-github-account

Clone a repo

```
echo "# hello" >> README.md
git add README.md
git commit -m "adding name of repo to README"
git push
```

Setting up and using Virtualenv

Surprise, the Python standard library includes a module called venv[7]. A virtual environment solves a significant problem in Python. It isolates the Python interpreter to a specific directory. In this example, a virtual environment works in a user's home directory.

Create Hello World Virtual Environment in Python

```
python3 -m venv ~/.hello
```

To use this "virtual environment," it needs to be activated.

Activate Hello World Virtual Environment in Python

```
source  ~/.hello/bin/activate
```

Using a repeatable convention to create virtual environments

Conventions are a powerful way to simplify complex software engineering tasks in a series of easy to remember steps. A convention-based workflow with virtual environments can also dramatically simplify using them. Here is a simple convention to use:

1. Create a virtual environment with a `~/.[reponame]` format

This process removes the decision about where to put the virtual environment and what to name it. If your git repository is called `hello`, then you would run the following command:

```
python3 -m venv ~/.hello
```

Note that the `.` makes the virtual environment invisible. This process will prevent your home directory overflowing with virtual environments when you open it in a GUI or list the contents with `ls -l`.

2. Create an alias in your Bash or ZSH environment.

With ZSH, the config file to edit would be `~/.zshrc` in Bash, it would be `~/.bashrc`. Inside of this config file add the following:

[7] https://docs.python.org/3/tutorial/venv.html

Create an alias for Git repo and virtual environment

```
## Hello repo
alias hello="cd ~/hello && source ~/.hello/bin/activate"
```

The next time you open your default shell, this alias will be available. Here is an example of what this workflow looks like on my ZSH environment, which uses a package called oh-my-zsh[8].

Use alias that performs `cd` and activates `hello` virtual environment

```
 hello
(.hello)  hello git:(master)
(.hello)  hello git:(master) which python
/Users/noahgift/.hello/bin/python
```

This convention-based workflow, if followed, makes a tedious and error-prone process easy to remember.

Installing packages and dependencies

There are several strategies available to install packages and dependencies. These are pip[9], conda[10] and docker format[11] containers. The most common is `pip`. Let's focus on that.

There is a convention to use with `pip` that will make you more productive.

1. Always use `pip` inside a virtual environment or a container.

Let's walk through a few scenarios:

- **Scenario A**: If you are using Google Colab notebooks[12], doing a pip install is fine since it is running in a container behind the scenes. Additionally, this is a managed environment designed for Python development, and the libraries are up to date for you. This notebook is a unique but useful environment. You could install it as follows: `!pip install pandas`. The `!` allows shell commands to run in a Jupyter notebook.
- **Scenario B**: If you provision an AWS Cloud9[13] development environment, this first thing you should do is create a virtual environment before you begin work. The native Amazon Linux or Ubuntu operating system needs to be isolated from what you want to do for a particular development task.

[8] https://ohmyz.sh/
[9] https://pip.pypa.io/en/stable/user_guide/
[10] https://docs.conda.io/en/latest/
[11] https://www.docker.com/
[12] https://colab.research.google.com/
[13] https://aws.amazon.com/cloud9/

- **Scenario C**: If you are developing a new project on your laptop, say an OS X laptop, you should use a virtual environment. This process will isolate your project from whatever installation occurs on the system.

2. In a new project, use a `requirements.txt` file. If the project deploys somewhere, it is a best practice to `freeze` the currently installed versions: `pip freeze > requirements.txt` and then install using `pip install -r requirements.txt`.

3. Run `make install` with `pip`. This snippet is an example of `Makefile`.

`Makefile` contents

```
install:
    pip install --upgrade pip &&\
        pip install -r requirements.txt
```

If this convention follows on all projects, it dramatically reduces the chance something will go wrong in installing software with `pip`. Additionally, there is little to remember to install software; a user runs `make install`.

Setup Visual Code code

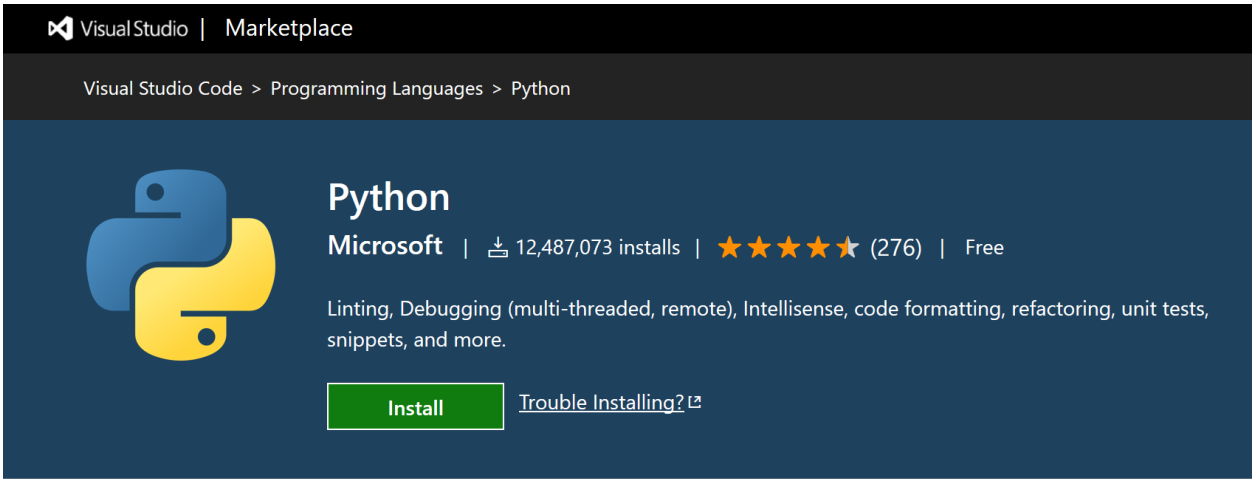
Having a reliable source code editor can dramatically improve the efficiency of software development. The Visual Studio Code[14] is one of the most popular editors for Python projects for a reason, it works! These are the steps to installing and using the Visual Studio Code toolkit.

1. Download the version for your platform[15].

Visual Studio Code has a version that works for Windows, Linux (Debian, Ubuntu, Red Hat, Fedora, and SUSE) and OS X. This is a significant advantage off the bat. You can use the same editor for any operating system.

2. Install the Visual Studio Code Python extension[16].

[14] https://code.visualstudio.com/
[15] https://code.visualstudio.com/#alt-downloads
[16] https://marketplace.visualstudio.com/items?itemName=ms-python.python



Python Visual Studio Code Extension

3. Install the correct Python interpreter for your operating system. This interpreter should be higher than Python 3.7 or later.

- If you are on Windows, you can install from the official Python Windows page[17].
- If you are on OS X, you should do a `brew install python3`. You should NOT use the install Python that comes with OS X. You will need to install Homebrew[18] if you haven't installed it. Note you can also upgrade the version of Python by using `brew upgrade python`.

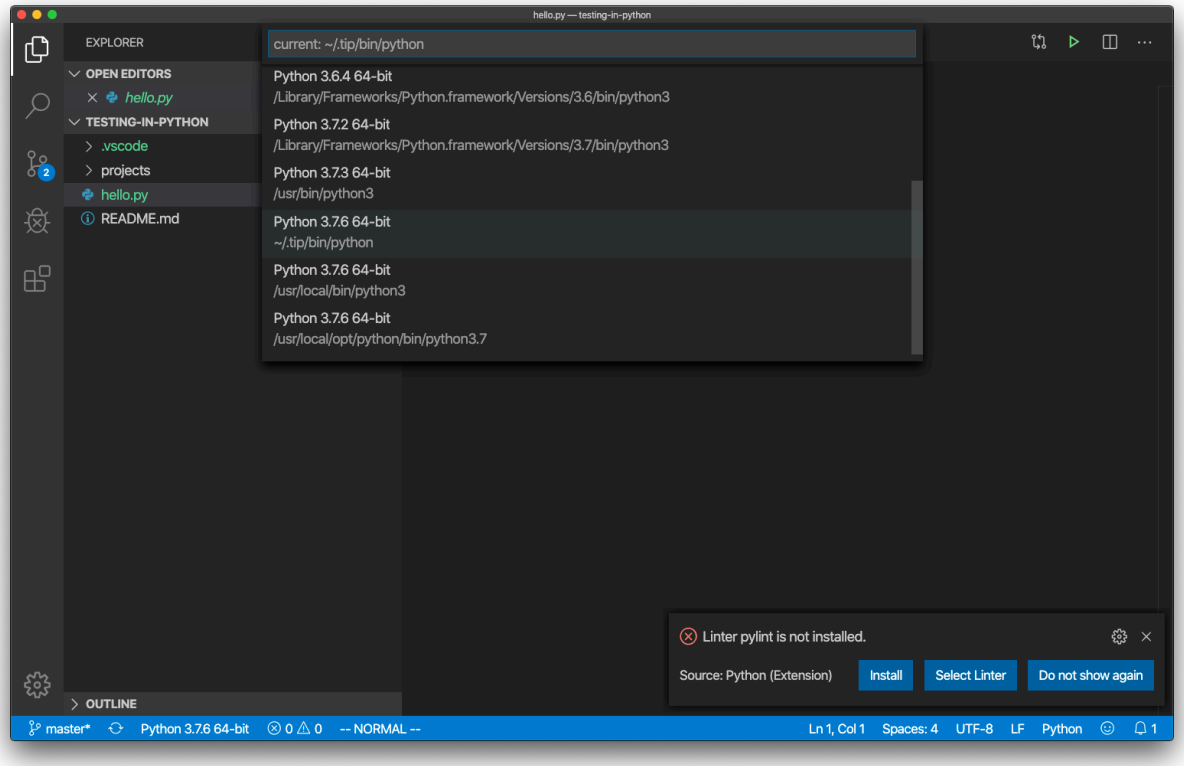
brew upgrade example

```
(.tip) > brew upgrade python
==> Upgrading 1 outdated package:
python 3.7.5 -> 3.7.6_1
==> Upgrading python
Warning: Building python from source:
  The bottle needs the Apple Command Line Tools to be installed.
  You can install them, if desired, with:
    xcode-select --install
```

- If you are on Linux, you may want to upgrade to a specific version of Python using the native package management system. If `pip` needs to install, you can install it with `get-pip.py`[19].

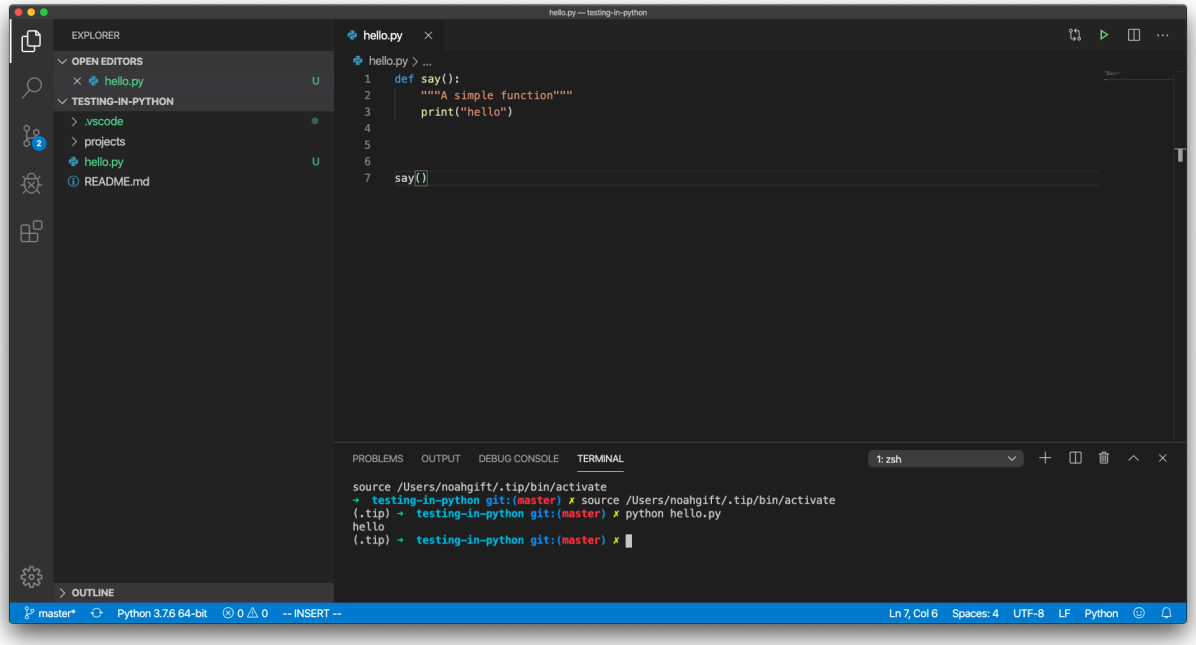
4. Use the virtual environment setup instructions described early in the chapter to `activate` a virtual environment. Next, launch visual studio code inside the activated environment: `code .`. This step will start Visual Studio code within your virtual environment. Note, it is essential to double-check that Visual Studio Code has the correct interpreter. It toggles as shown.

[17] https://docs.python.org/3.7/using/windows.html
[18] https://brew.sh/
[19] https://pip.pypa.io/en/stable/installing/#installing-with-get-pip-py



Python interpreter

Also, note that this same interpreter can be selected to test the code as shown.



Python interpreter

Here is the example code for you to test on your own. Paste parts of the code to see how linting, syntax highlighting, and auto-completion works.

A Hello World Function to Try in Visual Studio code

```
def say():
    """A simple function"""
    print("hello")

say()
```

Setup and use Vim

A kitchen has many types of knives. There is a steak knife to cut food like a steak while eating. There is a Chef's Knife that may be very high quality, large and expensive. Its use is for tasks the require power, like chopping an onion. A paring knife solves problems of precision. One example use case of a paring knife is to peel a tomato.

Likewise, `vim`, the editor serves a particular purpose. It is an editor that is ubiquitous, and it makes many tasks simple to solve. Every developer needs a little vim in their software kitchen.

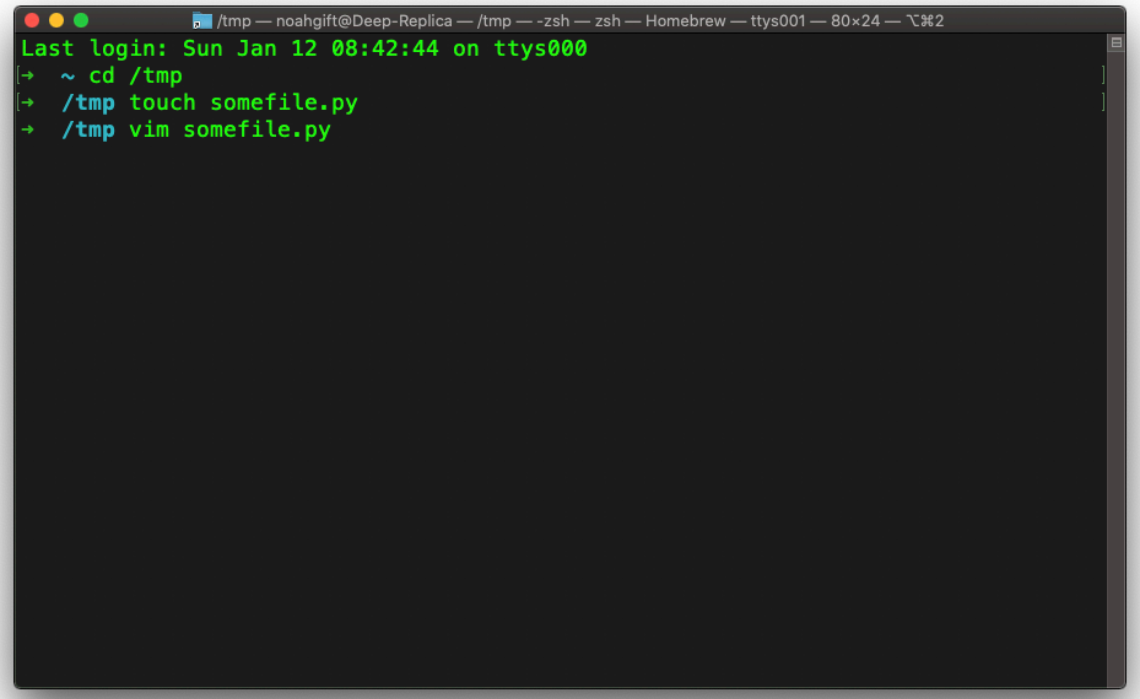
Install Vim

The `vim` program preinstalls on many machines. If `vim` is not available, you refer to the download instructions from the official vim website[20].

Use Vim

Mastering `vim` can take years, but there is a minimalistic approach to be effective immediately. This process is a convention to follow:

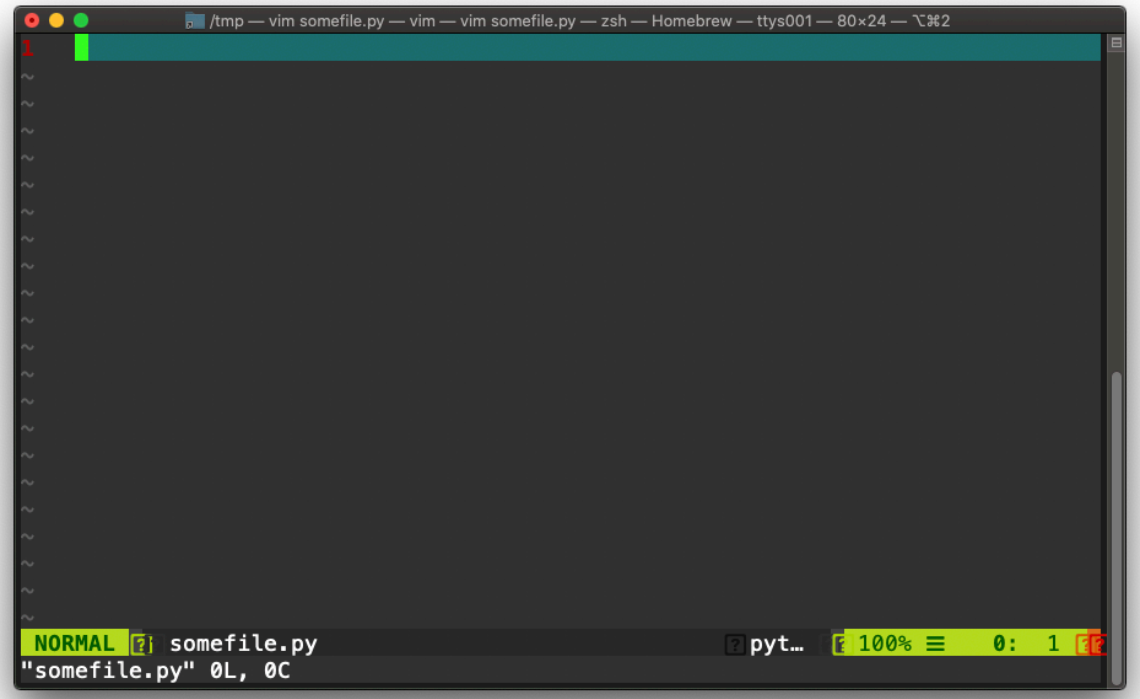
1. Create a file: `touch somefile.py`
2. Edit the file in vim: `vim somefile.py`



Create file

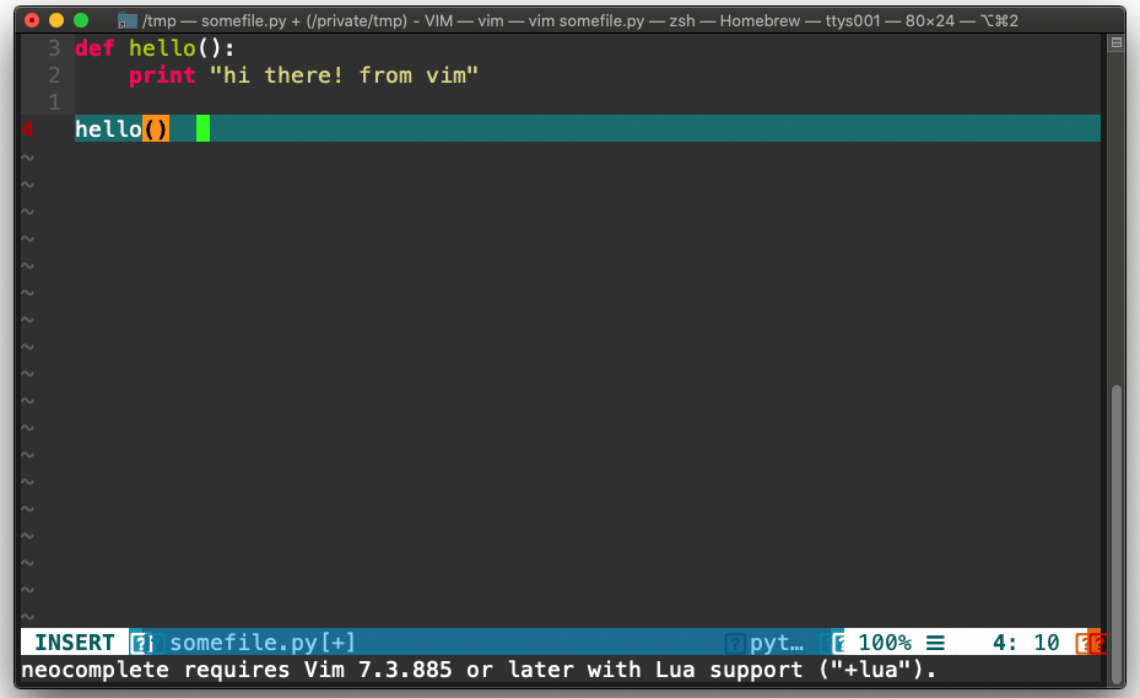
[20] https://www.vim.org/download.php

3. Switch to “insert mode” by pressing the key `i`. You will start in ‘normal’ mode.



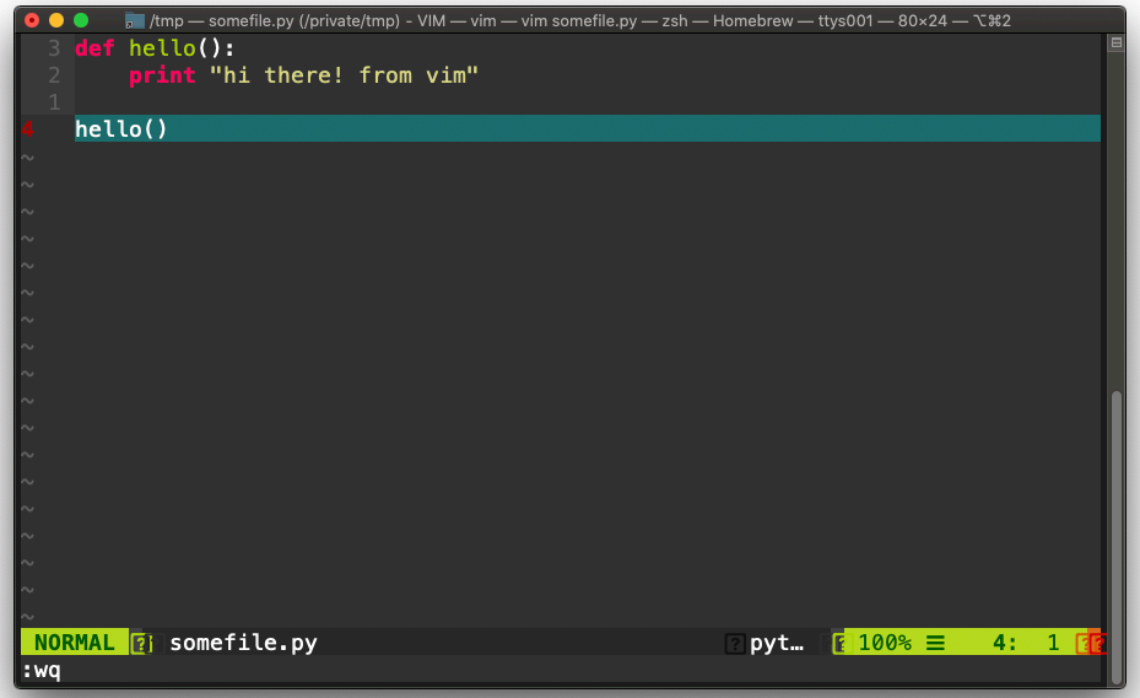
Insert mode

4. Edit the file.



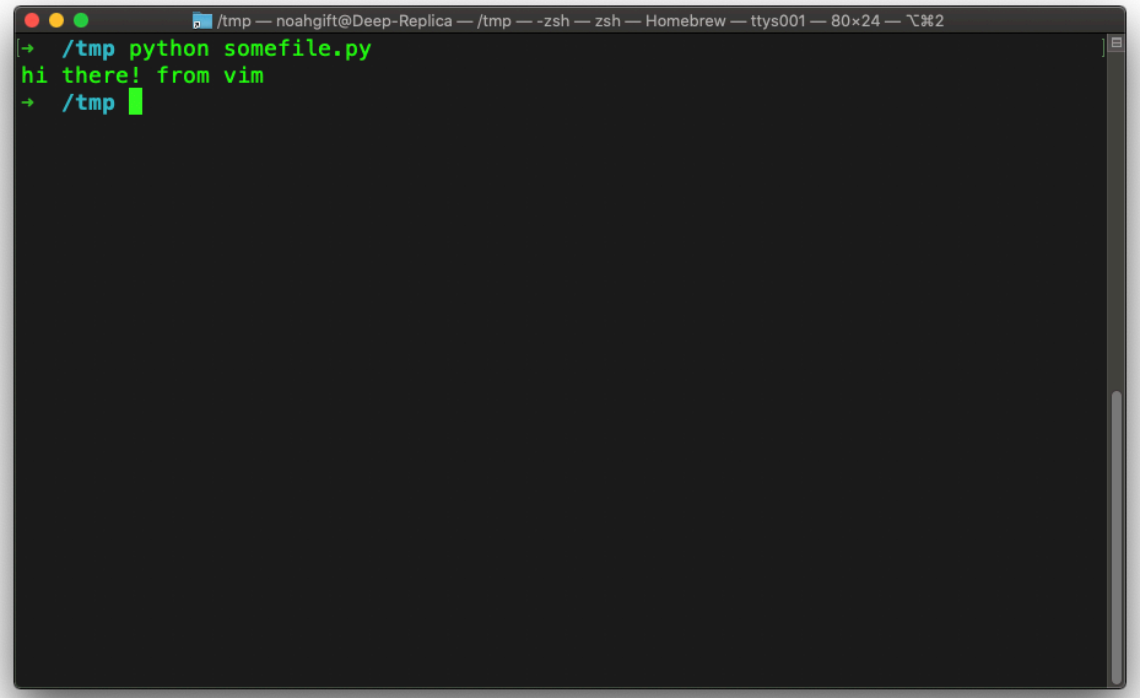
Edit file

5. Save the file by pressing Escape key and typing: `:wq`.



Save

6. Run the file.



Create file

If you get yourself in a mode that seems to be confusing, often the best way out is to press the Escape key. This step is the best way to solidify this knowledge; to do this workflow many times on many different types of machines. It will cement the learning.

Later feel free to get more advanced and learn new `vim` tricks, but first master the basics, and this will add an entirely new capability that will open up many workflows.

Setup Makefile

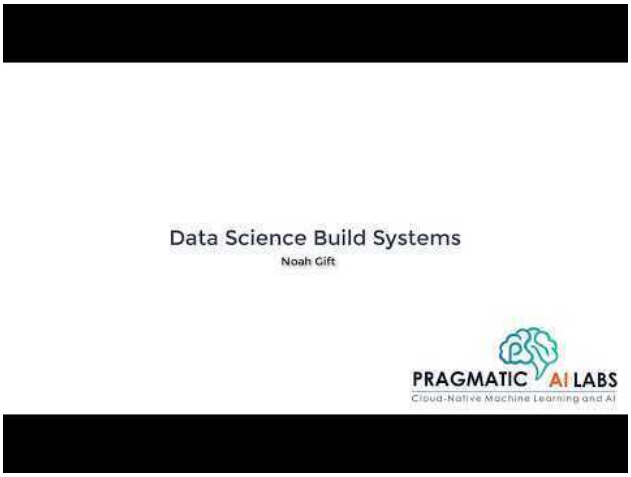
Just like `vim`, mastering `Makefiles` can take years, but a minimalistic approach provides immediate benefits. The main advantage of a `Makefile` is the ability to enforce a convention. If every time you work a project, you follow a few simple steps, it reduces the possibility of errors in building and testing a project.

A typical Python project improves by adding a `Makefile` with the following steps: `make setup`, `make install`, `make test`, `make lint` and `make all`.

Example Makefile

```
setup:
    python3 -m venv ~/.myrepo

install:
    pip install --upgrade pip &&\
        pip install -r requirements.txt

test:
    python -m pytest -vv --cov=myrepolib tests/*.py
    python -m pytest --nbval notebook.ipynb


lint:
    pylint --disable=R,C myrepolib cli web

all: install lint test
```

This example is from a tutorial repository called `myrepo`[21]. There is also an article about how to use it from CircleCI[22].



View this Video at https://www.youtube.com/watch?v=xYX7n5bZw-w[23].

Data Science Build Systems

The general idea is that a convention eliminates the need to think about what to do. For every project, there is a common way to install software, a common way to test software, and a common way to test and lint software. Just like `vim`, a `Makefile` build system is often already on a Unix or Linux

[21] https://github.com/noahgift/myrepo
[22] https://circleci.com/blog/increase-reliability-in-data-science-and-machine-learning-projects-with-circleci/

system. Even Microsoft uses the Linux operating system in Azure[24], and the result is that Linux is the preferred deployment target for most software.

Extending a Makefile for use with Docker Containers

Beyond the simple `Makefile`, it is also useful to extend it to do other things. An example of this is as follows:

Example Makefile for Docker and Circleci

```
setup:
    python3 -m venv ~/.container-revolution-devops

install:
    pip install --upgrade pip &&\
        pip install -r requirements.txt

test:
    #python -m pytest -vv --cov=myrepolib tests/*.py
    #python -m pytest --nbval notebook.ipynb

validate-circleci:
    # See https://circleci.com/docs/2.0/local-cli/#processing-a-config
    circleci config process .circleci/config.yml

run-circleci-local:
    # See https://circleci.com/docs/2.0/local-cli/#running-a-job
    circleci local execute

lint:
    hadolint demos/flask-sklearn/Dockerfile
    pylint --disable=R,C,W1203,W1202 demos/**/**.py

all: install lint test
```

A `Dockerfile` linter is called `hadolint`[25] checks for bugs in a `Dockerfile`. A local version of the CircleCI build system[26] allows for testing in the same environment as the SaaS offering. The minimalism is still present: `make install`, `make lint` and `make test`, but the `lint` step is complete and authoritative with the inclusion of `Dockerfile` as well as `Python` linting.

[24] https://azure.microsoft.com/en-us/overview/linux-on-azure/
[25] https://github.com/hadolint/hadolint
[26] https://circleci.com/docs/2.0/local-cli/

Notes about installing `hadolint` and `circleci`: If you are on OS X you can `brew install hadolint` if you are on another platform follow the instructions from `hadolint`[27]/ To install the local version of `circleci` on OS X or Linux you can run `curl -fLSs https://circle.ci/cli | bash` or follow the official instructions for local version of the CircleCI build system[28]

Setup and Use ZSH/Bash

The shell environment of Bash or ZSH is a given for working with Python. As discussed previously, most deployment targets are now Linux. With the widespread adoption of containers, even Windows is now a Linux target. What is the format for authoring `Dockerfiles`? The format is largely Bash Commands[29]. Additionally, Bash[30] and ZSH[31] are largely compatible with a few small differences.

Use oh-my-zsh

Let's focus mainly on ZSH through the use of an open-source framework for ZSH called Oh My Zsh[32]. The way to install it via the following command:

```
sh -c "$(curl -fsSL https://raw.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

After installation, you will get many convenient features: automatic `cd` (you don't need to type anymore), enhanced shell completion, and environment context recognition. Notice the `zsh` prompt I am using the write the book.

```
(.tip) ›  testing-in-python-book git:(chapter1) □
```

The `oh-my-zsh` plugins allow for automatic recognition of the Python Virtual Environment as well as the fact that I am in a git repository and what branch I have.

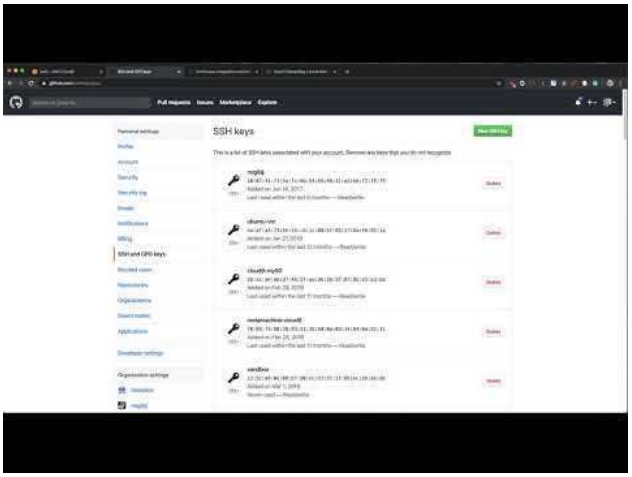
Using Cloud-based development environments

Just as many environments are Linux, it is also true that most deployment environments are in the cloud. Three of the largest cloud providers are: AWS[33], Azure[34] and GCP[35]. To write software that deploys on Cloud Computing environments, it often makes more sense to write, test, and build code in cloud-specific development environments. Let's discuss two of these environments.

[27] https://github.com/hadolint/hadolint
[28] https://circleci.com/docs/2.0/local-cli/
[29] https://docs.docker.com/engine/reference/builder/
[30] https://www.gnu.org/software/bash/
[31] https://en.wikipedia.org/wiki/Z_shell
[32] https://ohmyz.sh/
[33] https://aws.amazon.com/
[34] https://azure.microsoft.com/en-us/
[35] https://cloud.google.com/

AWS Cloud9

The AWS Cloud9 Environment[36] is an IDE that allows a user to write, run, and debug code (including serverless code in Python) in the AWS cloud. This step simplifies many workflows, including security and network bandwidth. You can watch a walkthrough video here that creates a new AWS Cloud9 environment.



View this Video at https://www.youtube.com/watch?v=4SIFF1PAMbw[37].

Setup CI Pipeline with AWS Cloud9 and CircleCI

GCP Cloud Shell

The GCP Cloud Shell[38] environment allows a user to develop software directly inside of the GCP ecosystem. This step is completely free and provides for many simplifications of cloud-based development tasks.

[36] https://aws.amazon.com/cloud9/
[38] https://cloud.google.com/shell/

Chapter 2: Testing Conventions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Directories

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Tests outside of the package

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Tests inside of the package

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Files

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Functions, Classes, and test methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Special test class methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Utilities in classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Good naming patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 3: Introduction to Pytest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

The most simple test possible

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Why is Pytest important?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

The power of `assert`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Pytest vs. Unittest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 4: Test Classes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Setting up and teardown of xunit-style tests

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 5: Reporting

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

PyTest Reporting

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Verbose output

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

PyTest Coverage Reports

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

PyTest with Jenkins and JUnit

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Code Quality

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

The people analytics of a team

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Churn

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Linting

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Flake8

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Pylint

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Code Formatting with Python Black

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 6: Debugging with Pytest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

How to debug code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Using a debugger

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Python Debugger (PDB) integration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 7: Pytest fixtures and plugins

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

What are fixtures?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Creating new fixtures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Built-in Fixtures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

capsys

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

tmpdir

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Advanced Fixture usage

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Teardown

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Scope

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Parametrizing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 8: Monkeypatching

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Why and when to monkeypatch?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

monkeypatching is hard

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

The simplest monkeypatching

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Automatic and global monkeypatching

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Other patching

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

When not to monkeypatch

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Patching builtin modules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 9: Testing matrix with Tox

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Testing different Python versions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Expanding the testing matrix

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Understanding variables better

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Using factors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Linting and other validations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 10: Continuous Integration and Continuous Delivery

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

What is Continuous Integration and Continuous Delivery and Why Do They Matter?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Jenkins

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

CircleCI

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Setup CircleCI inside AWS Cloud9

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Extending a Makefile for use with Docker Containers and CircleCI

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

GCP Cloud Build

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Cloud Build Continuous Deploy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Continuous Delivery for Hugo Static Site from Zero using AWS Code Pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Hugo AWS Continuous Delivery Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Setting up SSL for CloudFront

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

CloudFront configurations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Integrating Route53 with CloudFront distribution:

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Building Hugo Sites Automatically Using AWS CodeBuild

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Creating IAM Role

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Github Actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 11: Case Studies and War Stories

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Testing Click Commandline Tools

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

War Story: The Health Check that wasn't wrong

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

War Story: The Nine Circles of Hell While Parsing XML

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

War Story: The Mysterious Vanishing Servers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Chapter 12: Essays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Writing clean, testable, high quality code in Python

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

A clean code hypothetical problem

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

What is cyclomatic complexity?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

A clean code hypothetical solution

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Increase reliability in data science and machine learning projects with CircleCI

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Data science project quality

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Data science project automated testing setup

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/testinginpython.