python-tutorials Documentation

Release 1.0.0

Jose A. Lerma III

Nov 19, 2018

GETTING STARTED

Persistent Python practice produces prodigious productivity.

CHAPTER

ONE

INTRODUCTION

Python is a great language for getting things done quickly; however, a good deal of resources (mainly RAM (Random Access Memory)) are recommended. There are ways to incorporate C/C++ from within Python, but some may find it easier to port it over.

For more thorough intros, get lost in Python's Beginner's Guide or Wikipedia's Python page for a day or so and come back.

Looks up, then puts down Steam Controller

You're back? Alright, let's continue.

CHAPTER

TWO

GETTING STARTED

There are many ways to install and use Python depending on platform and IDE (Integrated Development Environment) (if any). These docs cover the methods I frequently use.

2.1 Windows

For Windows, I use but one editor: *Atom*; however, I give *PyCharm* an honorable mention. The Atom setup is lightweight and portable while the Pycharm setup is extensible and full-featured.

I have Pycharm on a Windows 10 Technical Preview VM (Virtual Machine), and it works well, but is quite bloated for a Windows VM running in Windows (Windows-ception?).

2.2 Linux

For Linux, I have two main IDEs: *Vim* and *PyCharm*. The Vim setup is lightweight and available without too much effort while the PyCharm setup is extensible and full-featured.

While I have a couple of Linux boxes (at the moment), I am very security minded when it comes to my Linux machines, so I prefer to run a Development VM of Linux on Windows. Excessive, yes, but taking snapshots, cloning, and reinstalling on VMs is easier than on physical machines.

I've read that some python bots can be run on a Raspberry Pi. I would like to tinker with this concept a bit, but I am concerned that Raspberry Pis do not have enough RAM, so I will be sticking with VMs until I can get more tests done.

CHAPTER

THREE

DISCLAIMER

Though covered by the MIT License, I reiterate: executable programs written from code on the Internet can end up doing bad things.

> Read and understand all code you copy and paste before running it.

3.1 Atom

From the Atom.io page:

> Atom is a text editor that's modern, approachable, yet hackable to the core—a tool you can customize to do anything but also use productively without ever touching a config file.

Personally, I **have** had to edit a config file to setup a proxy, so YMMV (Your Mileage May Vary).

Atom is also surprisingly full-featured (e.g. plugins, themes, file system browsing) given that it can be installed in a portable configuration and is multi-platform.

3.1.1 Windows Setup

While Atom is multi-platform, I only use it on Windows.

As aforementioned, I tend to use the zipped Atom files along with the PortableApps.com Platform to create a portable base environment. Next, I extract the zipped Atom files into `X:\PortableApps\Atom\`, as an example.

Then, you'll need to get the atom-runner package so that you can run the Python programs with an `ALT + R` key combo. However, `atom-runner` will not work if you have to input data from terminal, so you will need either the built-in Command Prompt or a PA.com portable enhancement like Console Portable. When you first open Atom, an `.atom` folder will be created in `%USERPROFILE%`, this folder will need to be moved into `X:\PortableApps\` to keep your settings.

As for Python, I get the embeddable zip files and extract them into `X:\PortableApps\CommonFiles\python3\` to continue with the portable theme. If you want different versions of Python, you can make different folders e.g. `python2.7`, `python3.6`, `python3.5`

Finally, the easiest way to get Atom to find your portable Python installation is to use a shebang on the first line of code `#! X:\PortableApps\CommonFiles\python3\python.exe`

3.2 Vim

Vim is a configurable, open source, and cross platform text editor that is an improvement of the vi editor in most Linux distros.

It has nifty things like syntax highlighting, colorization, and a scripting language to make your own plugins, etc.

3.2.1 Setup

As aforementioned, it is cross platform (and open source), so it can run on anything (even Potato). Personally, I prefer to use it on Linux only because it is usually in the default repository and has both syntax highlighting and colorization, which are a great improvement upon vi in CLI.

Linux

If your distro does not have vim in its default repo, then I fear you will have to compile from source code.

Windows

If you should want to use Vim on Windows, and not use gVim at PA.com, then both binaries and executables are available for you.

Other

Believe it or not, Vim is available on even more architectures: Amiga, OS2, Macintosh, Android, iOS, WindowsCE, Cygwin, and others.

3.2.2 Plugins/Scripts

Vim has a library of thousands of powerful scripts that are easy to make if it is missing something you want. Personally, I do not use any since I mainly use Vim as a quick, light editor.

3.3 PyCharm

PyCharm is very much like Python itself: quick to develop on, full-featured, and resource heavy. PyCharm is a true IDE: a console and debugger are all built-in. I especially like the PEP8 checks.

3.3.1 Linux Setup

Though Pycharm is multi-platform, I mainly use it on Linux.

Unless you are willing to pay for a license, you are probably going to want the free community edition. Download the `tar.gz` file wherever you like, then extract the `pycharm-community-20xx.x.x` folder. Therein, run the `pycharm-community-20xx.x.x/bin/pycharm.sh` file from within terminal.

Once setup is complete, on the menu bar, go to "Tools>Create Desktop Entry..." to make it easier to open later. **Do not delete** the `pycharm-community-20xx.x.x` folder because that is where it is running from.

Upgrades follow the same procedure, except that you can delete the previous `pycharm-community-20xx.x.x` version folder.

3.3.2 Python Binaries

Installing Python will depend on your Linux distro. Most will have some version of Python either built-in or available from the package manager. PyCharm can auto-detect and use these installed versions. The Ubuntu Setup of my ClashCallerBot is an example of how easy setting up Python can be.

However, if you are unlucky, you will have to download the source and compile it yourself.

3.3.3 Windows Setup

Windows installation is very straightforward:

- Install Python with the Python executable installer.
- Install PyCharm using the Community Edition executable installer.

Any extra packages or modules would have to be added, but most programs can be run with the base installations.

3.4 CrackingCodesWithPython

3.4.1 CrackingCodesWithPython package

Subpackages

CrackingCodesWithPython.Chapter01 package

Submodules

CrackingCodesWithPython.Chapter01.PracticeQuestions module

CrackingCodesWithPython.Chapter01.caesarCipher module

CrackingCodesWithPython.Chapter01.caesarHacker module

CrackingCodesWithPython.Chapter01.config module

Module contents

CrackingCodesWithPython.Chapter07 package

Submodules

CrackingCodesWithPython.Chapter07.addNumbers module

CrackingCodesWithPython.Chapter07.helloFunction module

CrackingCodesWithPython.Chapter07.transpositionEncrypt module

Module contents

CrackingCodesWithPython.Chapter08 package

Submodules

CrackingCodesWithPython.Chapter08.transpositionDecrypt module

Module contents

CrackingCodesWithPython.Chapter11 package

Submodules

CrackingCodesWithPython.Chapter11.PracticeQuestions module

CrackingCodesWithPython.Chapter11.detectEnglish module

Module contents

CrackingCodesWithPython.Chapter13 package

Submodules

CrackingCodesWithPython.Chapter13.PracticeQuestions module

CrackingCodesWithPython.Chapter13.cryptomath module

Module contents

CrackingCodesWithPython.Chapter14 package

Submodules

CrackingCodesWithPython.Chapter14.PracticeQuestions module

CrackingCodesWithPython.Chapter14.affineCipher module

CrackingCodesWithPython.Chapter14.affineKeyTest module

Module contents

CrackingCodesWithPython.Chapter15 package

Submodules

CrackingCodesWithPython.Chapter19.PracticeQuestions module

CrackingCodesWithPython.Chapter19.freqAnalysis module

Module contents

CrackingCodesWithPython.Chapter20 package

Submodules

CrackingCodesWithPython.Chapter20.PracticeQuestions module

CrackingCodesWithPython.Chapter20.vigenereDictionaryHacker module

CrackingCodesWithPython.Chapter20.vigenereHacker module

Module contents

CrackingCodesWithPython.Chapter22 package

Submodules

CrackingCodesWithPython.Chapter22.PracticeQuestions module

CrackingCodesWithPython.Chapter22.primeNum module

Module contents

Submodules

CrackingCodesWithPython.pyperclip module

Module contents

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search