TRAN DUC LOI

PYTHON FOR DESKTOP APPLICATIONS

Develop, pack and deliver Python apps with TkInter and Kivy

FIRST EDITION

PYTHON FOR DESKTOP APPLICATIONS

How to develop, pack and deliver Python applications

with TkInter and Kivy

First Edition

Copyright

First released: October 2020

Revision: 20210725

Asset files used in this book are created by ***Tran Duc Loi***.

About the Author

Tran Duc Loi aka ***Leo Tran*** is a fan of minimalism and an enthusiast Python programmer since 2014 when he accidentally asked to join a new Python community with some friends. Since then, he found that he has special interest with Python.

Beside Python, he also added Javascript to his stack with React JS for frontend and React Native for mobile development.

His style of writing is ***practical*** that's why you'll find his books full of illustrations, diagrams and examples with a little of theory.

Table of Contents

Preface

What this book covers

Chapter 1, Introduction shows you some fundamental concepts of Python such as pip, wheel, virtual environment, GIL, CLI and GUI, which tools we will use, how to set them up.

Chapter 2, Create a File Downloader with TKInter introduces how to develop a Python file downloader application with simple GUI using **`TKInter`** library. This chapter also guides you how to pack your application using **`PyInstaller`** and make a setup using **`NSIS`**.

Chapter 3, Create a Music Player with Kivy walks through how make a music player with **`Kivy`**. We will start with a very simple **`Kivy`** application then eventually build a more complex one. We also pack our music player up using **`PyInstaller`**.

Chapter 4, Debugging shows you how to debug your applications if something wrong. Useful tips and handy **`DependencyWalker`** debug tool guide. In this chapter, you will also be introduced to **`cx_Freeze`** to build/freeze a **`wx_Python`** application.

Please note that to keep the book briefly and to help readers focus on main target of the book, creating desktop applications, I will not cover tests in this edition. Depend on received feedback, I will consider to add more features in the next editions.

What you need for this book

In this book, we'll use:

- Python 3.7.9 64-bit
- Windows 10 64-bit
- PyInstaller 4.0
- NSIS 3.06.1
- Microsoft Visual Code 1.49.0 as main editor
- A Git client (Git-scm 2.27.0)
- Kivy 1.11.1
- cx_Freeze 6.2
- DependencyWalker 2.2 64-bit

All the packages/tools are the latest at the time of writing.

Who this book is for

This book is for any level of Python programmers.

If you have known Python before, it would be easier for you to read this book.

If you are a beginner, it still be fine for you to get start Python programming with this book. You need to read chapter 01 and explore as much as possible to understand some basic concepts: virtual environment, multithreading, pip, wheels.

At any level, I recommend you to try to explore hyperlinks if available. In every part of the book, there're a number of links to internal and external resources to provide more detail information.

How to use this book

This is a ***practical*** book. That means you shall need to get your hands dirty.

You need to:

1. Install the tools.
2. Clone the code from the book's repository. All examples of a chapter will be in a folder named of that chapter.
3. Run the examples to see how it works.
4. Dive into the code with your favourite IDE.
5. Read the notes to understand deeply, follow the links for more detail.
6. Complete the exercises if available.
7. Modify the code to adapt your purposes.

It's not about using the examples in this book to make commercial products, they are here for demonstration and being kept as simple as possible hence they are not shiny enough to be in production. You need to read them to understand the fundamental concepts and techniques.

Conventions

This book has some conventions that you should be familiar with in order to get more from this book.

One of my favourite wheel libraries is:
https://www.lfd.uci.edu/~gohlke/pythonlibs/

This is information box, tips and tricks. I usually explain things and answer some common questions here. It is an interesting box.

```
12  # ------------------------------------
13
14  import datetime, os, sys, platform
15
16  def hello():
17      print("Hello, this is Python version", s
```

This is source code of the examples. We need the line number as we will use line numbers as an anchor to explain the code.

```
L:\python4desktop\chapter01>python --version
Python 3.7.9

L:\python4desktop\chapter01>pip --version
pip 20.1.1 from c:\python37\lib\site-packages\
```

This is a command runs in PowerShell or Command line (in Windows) or Terminal (Linux).

If there is PS in the command, we are in Windows Powershell, if not, we are in command line.

`print()` command to print to output.
`sys.version` return the current version of the running
`platform.system()` returns the name of the operating
`datetime.datetime.now()` returns current date and tir
the computer's datetime

File names, commands, folder names usually use bold format with different font family than normal text just to emphasize them.

Reader feedback

All feedback about the content of the book should be sent to the Telegram group at https://t.me/py4da or emailed directly to the author at loitranduc@gmail.com. Let us know what you think about this book – what you liked or may have disliked.

To give any feedback about the source code, please:

- To find support interactively, chat with us at https://t.me/py4da (the official book's Telegram group)
- If you don't think you can provide a patch, open a New Issue at: https://github.com/loitd/python4desktop/issues
- If you think you can make some patch/fix the issue, fork the repository and make a Pull Request at: https://github.com/loitd/python4desktop/pulls.
- Please don't email me problems with source code. It should be done using our git tool.

For more information about Github flow, please refer to: https://docs.github.com/en/free-pro-team@latest/github/collaborating-with-issues-and-pull-requests

Reader feedback is important for us to develop titles that you really get the most out of.

Downloading example code

All the source code in the book can be found at https://github.com/loitd/python4desktop. You should use **`git clone https://github.com/loitd/python4desktop`** command or download directly from the github repository.

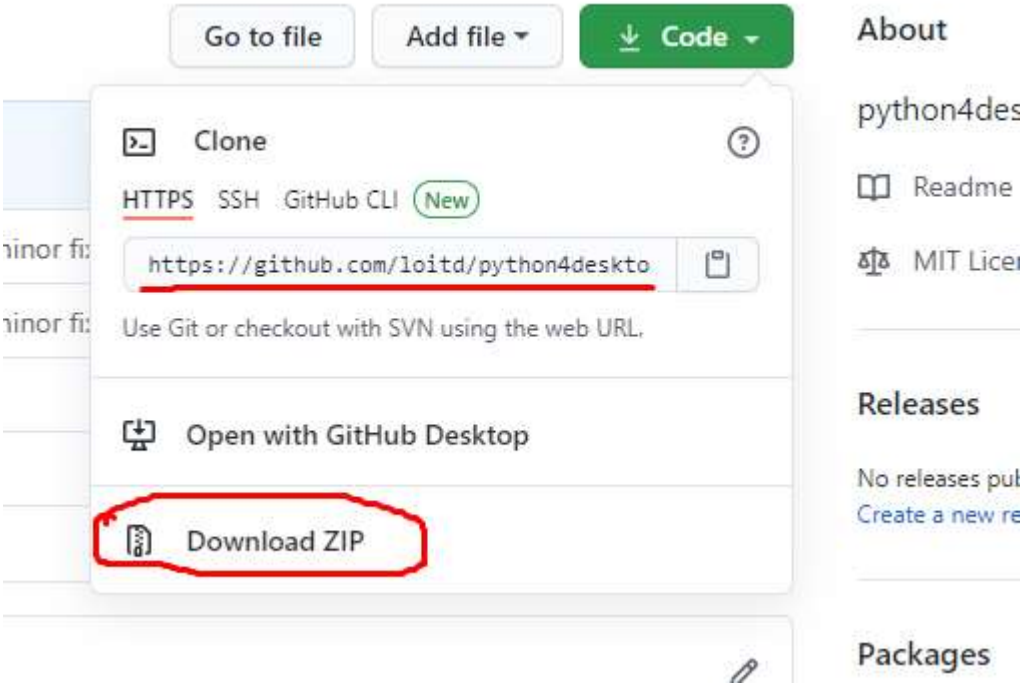

Figure 1. Downloading source code

This page intentionally left blank

Chapter 1: Introduction

1.1. Introduction

Python, in general, is an easy-to-learn yet powerful high level scripting programming language. Python is not clearly fall into interpreted nor compiled programming language, it has several implementations as follow:

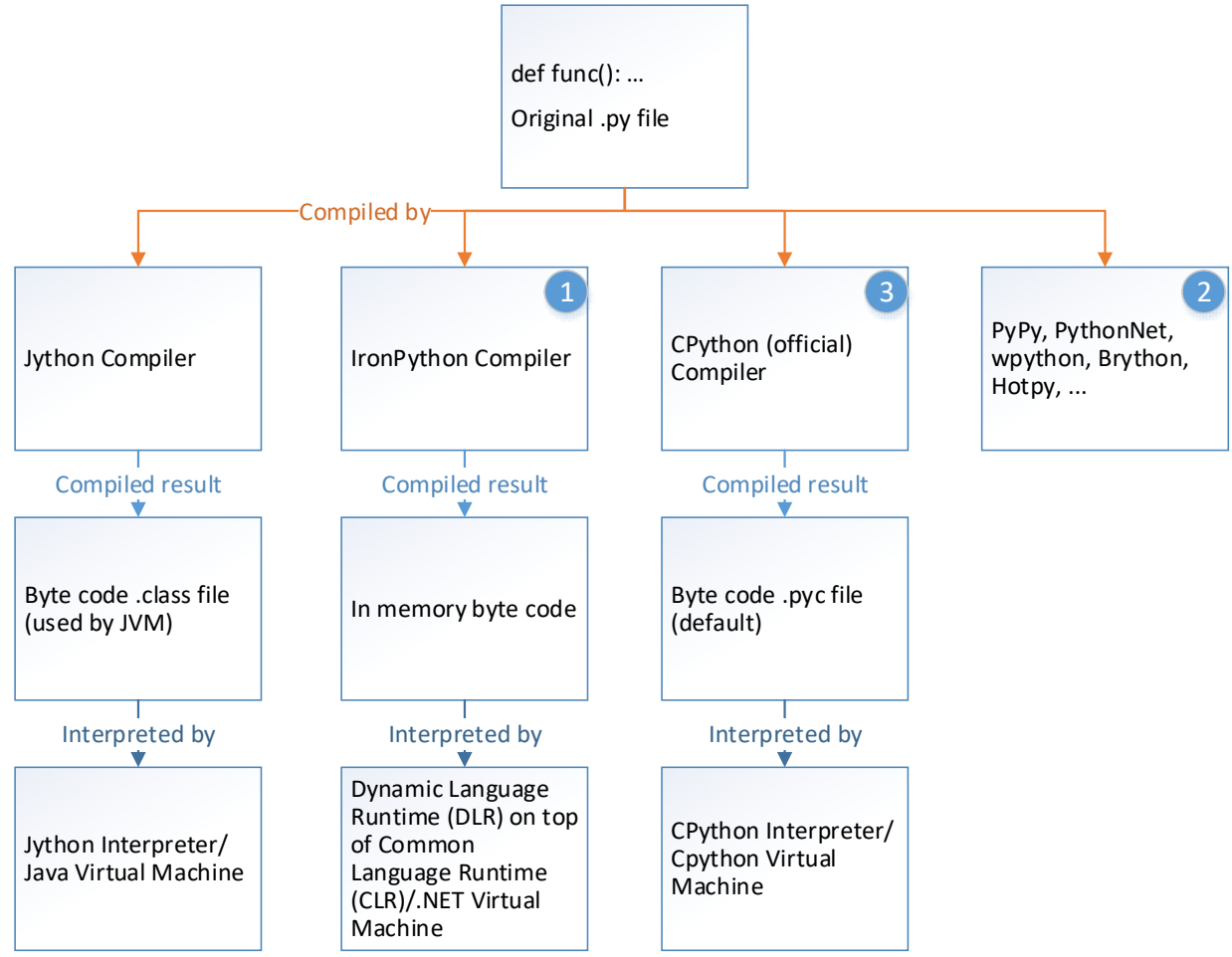

Figure 2. Python Implementations

Notes from above illustration:

1. At the time of writing, IronPython supports Python v2.7 and not officially support Python 3.

2. PyPi (Package Python Index) is a Python repository, should not be confused with PyPy. At the time of writing, PyPy v7.3.1 support Python 2.7 and 3.6.

   As PyPy wrote on their home page, PyPy is 4.2 times faster than CPython.

3 This is the official default implementation of Python. This book, without explicit mentioned, follows this implementation.

PIP

pip is a package installer for Python. You can use **pip** to install packages from the Python Package Index and other indexes.

By default, **pip** is shipped with Python installer so you don't have to install **pip** manually.

You can check your current pip version with **pip --version** command:

```
PS L:\> cd .\python4desktop\chapter01\
PS L:\python4desktop\chapter01> .\venv\Scripts\activate
(venv) PS L:\python4desktop\chapter01> pip --version
pip 20.1.1 from f:\code\src\github.com\loitd\python4desktop\chapter01\venv\li
b\site-packages\pip (python 3.7)
(venv) PS L:\python4desktop\chapter01>
```

Please note that **pip** will be created per virtual environment as you can see in the output above.

Wheels

Wheels are new standards of Python distribution. Please note that wheels are supported from **pip** >= 1.4 and **setuptools** >= 0.8.

Wheels end with **.whl** extension. You can easily find wheels at any project's download tab as below:

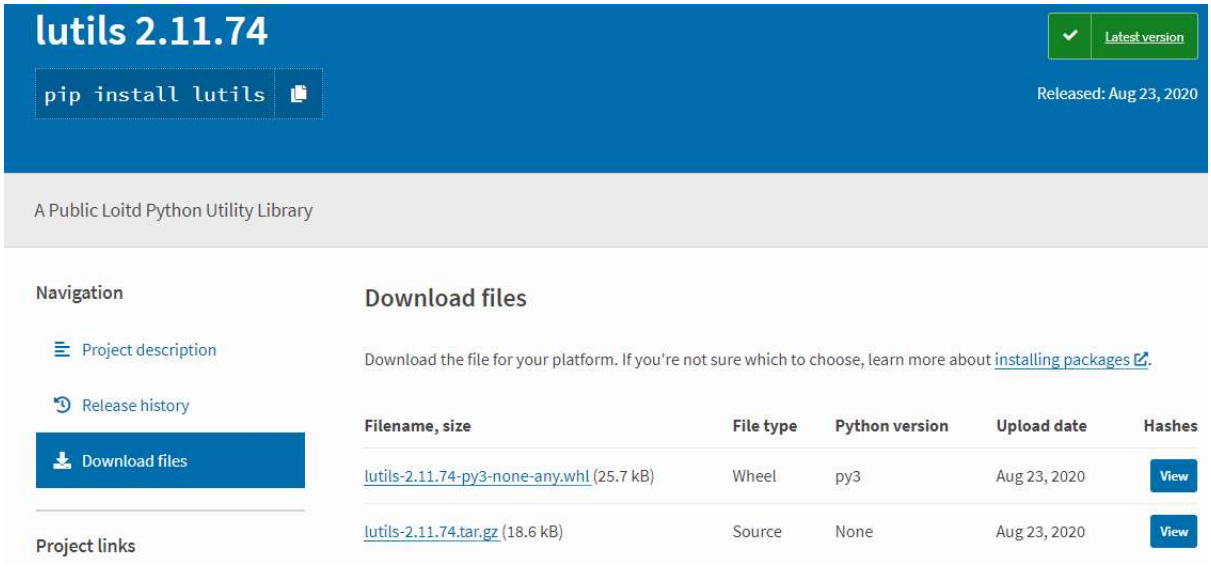

Figure 3. A Python wheel example

Once you have wheels downloaded, you can install the package locally using:

```
pip install file-name-of-the-wheel.whl
```

Wheels are extreme helpful for some kind of packages those don't have any wheels available at pypi.org. There is my experience with wheels at Python and choosing the right version at section 1.2 in this chapter.

Virtual Environment

Python virtual environment is a self-contained directory that contains a Python installation for a particular version of Python and a number of additional packages.

In this book, you will use a separated virtual environments for each chapter. E.g: chapter 01 will have its own virtual environment inside **`chapter01`** directory.

Python has a built-in virtual environment package named **`venv`** to allow you create a virtual environment:

```
pip -m venv venv
```

The above command creates a virtual environment into a folder inside current folder named **venv**.

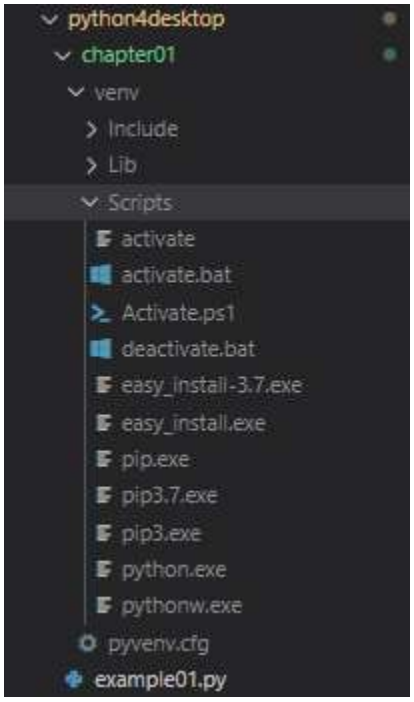

Figure 4. Python virtual environment structure in Windows

To use the newly created virtual environment, you have to activate it using **`activate`** command.

```
PS L:\> cd .\python4desktop\chapter01\
PS L:\python4desktop\chapter01> .\venv\Scripts\activate
```

After activating process completed, there will be a prefix **(venv)** at the beginning of command prompt to let you know that you are in a virtual environment.

To deactivate an active virtual environment, you need to use **`deactivate`** command.

```
(venv) L:\python4desktop\chapter01>venv\Scripts\deactivate.bat
L:\python4desktop\chapter01>
```

To install a package inside a virtual environment, you can use **`pip`** command as normal. The packages will be installed at **`venv/Lib`** directory.

To delete a virtual environment, you simply delete the created virtual environment folder (in this case **`venv`** folder).

Please note that:

- Python built-in **`venv`** package does NOT allow you to specify other Python versions. To overcome this, you need to specify full path to Python executable file when using the creation command. E.g:
    - **`C:\Python36-86\python -m venv venv86`**
    - **`C:\Python37-64\python -m venv venv64`**
- **`deactivate.bat`** currently not working in Windows Powershell.
- These above commands are used in Windows. In Linux/MacOS, you have to use **`source venv/bin/activate`** command instead.
- The structure of **`venv`** folders are different between Windows and Unix/MacOS environment.

GIL

`GIL` stands for **`Global Interpreter Lock`**, in **`CPython`**, is a mutex that protects access to Python objects, preventing multiple threads from executing Python bytecodes at once. This lock exists mainly because **`CPython`** memory management is not thread-safe.

Because of **`GIL`**, Python has a bottleneck and can't taking full advantage of multiprocessor systems.

Because of **`CPython`**'s GIL, several implementations have been introduced to remove GIL. Currently, **`Jython`** and **`IronPython`** are two implements that have no GIL.

Please take a look back at the beginning of this chapter for Python implementations.

CLI and GUI

Computers can display information and let the user give commands to it using two methods: a command line interface (CLI) or a graphical user interface (GUI).

In a command line interface or console, the user types commands using the keyboard to tell the computer to take an action.

In a graphical user interface, the user can use the computer mouse to click on buttons.

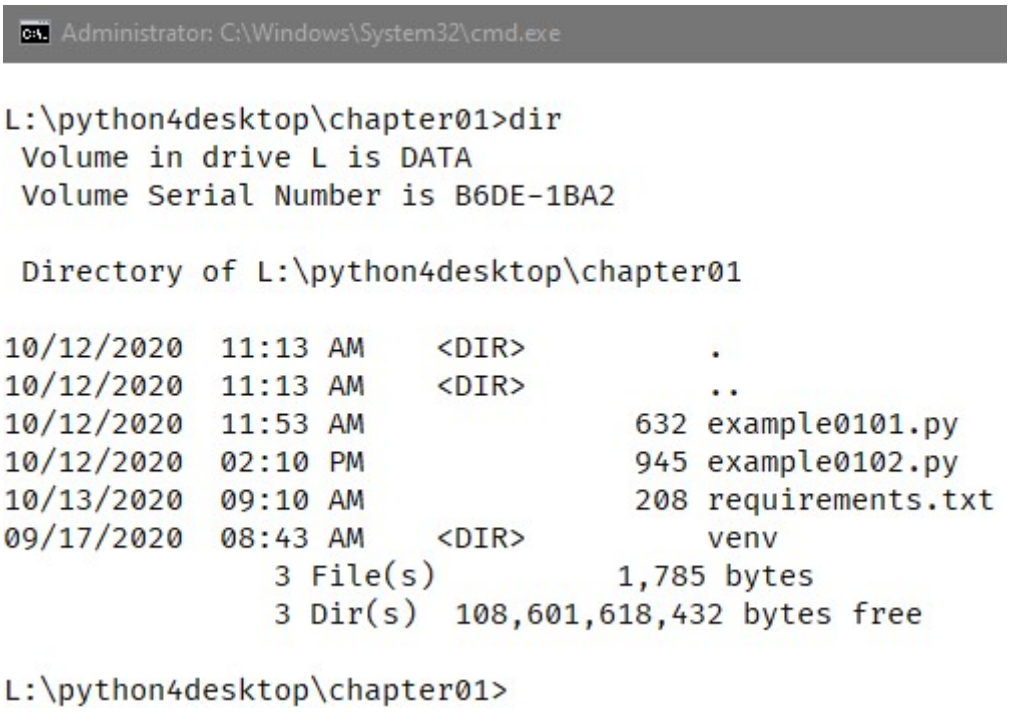

Figure 5. Console or CLI example

Figure 6. a GUI example

Those above 2 figures are examples of CLI/Console and GUI applications.