Introduction to Python

Jorge Pérez de Acha Chávez

Chapter 1

Installing Python

1 Windows

To install Python on your Windows operating system, download the latest version of Python 2.7 for Windows from the Python.org website. An .msi file will begin to download. Once it has finished, double click it to begin the installation.

By default, Python 2.7 will install in the `C:\Python27` directory. Typing the full name of the Python interpreter each time can get very tedious, so we can change the `PATH` environment variable to simplify things a bit. To change the `PATH` via the PowerShell, open a new Command Prompt and type `powershell` and press Enter. You will notice the prompt change from `C:\Users\yourUser` to `PS C:\Users\yourUser`. Go ahead and run the following:

```
[Environment]::SetEnvironmentVariable("Path", "$env:Path;C:\
    Python27\;C:\Python27\Scripts\", "User")
```

Type `exit` and press Enter to exit the `powershell`, and close the Command Prompt window for the change to take effect.

Open a new Command Prompt window, type `python` and press Enter. If the installation was successful, you should see something similar to this

```
C:\Users\jorge>python
Python 2.7.13 (v2.7.13, Dec 17 2016, 20:42:59)
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

TA-DAH! You now have Python installed and ready to go. Notice how the prompt has changed from your current directory to `>>>`. This means that you are now in the Python interpreter and no longer in the Command Line

interpreter. To exit the Python interpreter, simply type `exit()` and press Enter. The prompt should now display your current directory again.

Useful packages

pip

pip is a great tool for acquiring and managing additional Python packages. At this point, you might be wondering why you would need to install anything else other than Python itself. Well, there are several numerical, scientific, and visual packages available that far exceed the capabilities of Python's native tools. All right, but now you might be wondering why these tools are not already included within Python. As of the writing of this manual, there are over 115,000 packages available in PyPI. Most of these packages are independently developed and maintained by various teams around the world. It would be impossible for Python to develop such a large repository and still be concerned with maintaining the language itself. Besides, most people might not need to use all 115,000+ packages, so why include them in every distribution?

To get pip, download get-pip.py (i.e copy and paste the file's content on a blank Notepad document and save the file as get-pip.py). Place the file on your working directory (e.g. your Desktop), open a new Command Prompt window and place yourself in the same directory as get-pip.py (e.g. `C:\Users\yourUser\Desktop`). Type

```
python get-pip.py
```

and press Enter.

Visual C++ Compiler

In order for some Python packages to function properly, a C compiler must be installed. Head over to Microsoft's website to download and install the Microsoft Visual C++ Compiler for Python 2.7

2 Mac

A version of Python 2.7 is already included in every MacOS from version 10.2 (Jaguar) onwards. To see what version is installed, open a new Terminal window and type

```
$ python --version
```

then press Enter.

Your current distribution of Python 2.7 is most likely out of date, which is why installing a newer distribution is strongly recommended. You can download the latest version of Python 2.7 for MacOS from the Python.org website.

After installing the new distribution of Python, open a new Terminal window and type `python` and press Enter. If the installation was successful, you should see something similar to this

```
Jorges-Mac:~ jorge$ python
Python 2.7.13 (v2.7.13, Dec 17 2016, 20:42:59)
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

TA-DAH! You now have the newest Python installed and ready to go. Notice how the prompt has changed from your current directory to `>>>`. This means that you are now in the Python interpreter and no longer in the Command Line interpreter. To exit the Python interpreter, simply type `exit()` and press Enter. The prompt should now display your current directory again.

Useful packages

pip

pip is a great tool for acquiring and managing additional Python packages. At this point, you might be wondering why you would need to install anything else other than Python itself. Well, there are several numerical, scientific, and visual packages available that far exceed the capabilities of Python's native tools. All right, but now you might be wondering why these tools are not already included within Python. As of the writing of this manual, there are over 115,000 packages available in PyPI. Most of these packages are independently developed and maintained by various teams around the world. It would be impossible for Python to develop such a large repository and still be concerned with maintaining the language itself. Besides, most

people might not need to use all 115,000+ packages, so why include them in every distribution?

To get pip, begin by opening a new Terminal window, move to your working directory (e.g. Desktop) and run

```
curl -O https://bootstrap.pypa.io/get-pip.py
```

Now type in the same Terminal window

```
python get-pip.py
```

and press Enter.

C Compiler

In order for some Python packages to function properly, a C compiler must be installed. The easiest way is to download Xcode from the Mac App Store.

After Xcode has finished downloading (or if you already have Xcode installed), open a new Terminal window and type

```
xcode-select --install
```

and press Enter. A window will appear that reads:

```
The xcode-select command requires the command line developer
   tools. Would you like to install the tools now?
```

Click Install and wait for it to finish. Then go ahead and type

```
sudo xcodebuild -license
```

in the Terminal and press Enter. At this point you will need to enter your password. Follow the instructions on the screen until you type `agree`.

Virtual environments (virtualenv)

Virtual environments create isolated virtual Python environments. By creating different environments, packages and dependencies that belong to different projects can be stored in different places (i.e. different *environments*).

Why is this useful? Imagine you have two projects, Project X and Project Y. Suppose Project X works fine with AwesomePackage version 2.3, but you need AwesomePackage version 2.5 to keep developing Project Y. However, Project X is incompatible with version 2.5! If you do a systemwide AwesomePackage update, Project X will stop working. But if you don't, you won't be able to develop Project Y. Virtual environments prevent this conundrum from happening.

`virtualenv` is a widely used package for activating and managing virtual environments.

Installing virtualenv

To install `virtualenv` using `pip` execute

```
$ pip install virtualenv
```

on the command line. To verify the installation run

```
$ virtualenv --version
```

Using virtualenv

1. Create a directory for your project and place yourself in it

   ```
   $ mkdir project_directory
   $ cd project_directory
   ```

2. Create the virtual environment

   ```
   $ virtualenv environment_name
   ```

   The command `virtualenv environment_name` creates a new folder inside `project_directory` with a new and isolated Python executable and includes a copy of `pip` to install additional packages.

3. To activate the virtual environment run on

   Windows:

   ```
   environment_name\Source\activate
   ```

   Posix systems[1]:

   ```
   $ source environment_name/bin/activate
   ```

   The command prompt will now display the environment's name in parentheses to the left, indicating the environment is now active.

   ```
   (environment_name) C:\Users\jorge\project_directory>
   ```

   or

   ```
   (environment_name) Jorges-Mac:project_directory jorge$
   ```

   As long as the virtual environment is active, any packages or dependencies installed via `pip` will be stored in the `environment_name`'s Python files inside the `project_directory` directory, and not on your system's Python path.

   The virtual environment will remain active even though we move to directories other than `project_directory`. It is possible to have more than one virtual environment running at the same time, given each environment has its own command line window.

[1] Posix is a family of standards that includes MacOS and several Linux distributions. To activate virtualenv on other systems refer to the documentation.

4. To deactivate the environment, simply execute `deactivate`.

If you want to delete an environment, you just need to delete the `environment_name` directory (i.e move to Trash).

Chapter 2

Jupyter

As you might imagine, using the Python interpreter on the command line interface is quite tedious. The Jupyter Notebook is a rich interface that makes using Python a lot easier, specially for beginners. Jupyter allows to not only execute code, but to also add images and rich markdown comments that make for a much nicer code and execution presentation.

To install Jupyter execute

```
$ pip install jupyter
```

on the command line. To open the Notebook, execute

```
$ jupyter notebook
```

The command `jupyter notebook` opens a new notebook on the same directory where the command was executed.

kernels

Jupyter permits the use of other *kernels* besides the one included with the installation, IPython. This way, we can create kernels based on our virtual environments to be used with the Notebook.

To create a kernel based on a virtual environment:

1. Activate the virtual environment:

   Windows:

   ```
   environment_name\Source\activate
   ```

   Posix:

```
$ source environment_name/bin/activate
```

2. Execute

```
python -m ipykernel install --user --name
    environment_name --display-name "Environment Name"
```

This environment's kernel will now be available on the Jupyter kernel list.

Chapter 3

SciPy Stack

Jupyter is part of the SciPy stack, a stack of mathematics, scientific, and engineering packages for Python. I strongly recommend installing these packages (in an individual virtual environment, of course).

Windows

To install the stack, execute:

```
pip install numpy matplotlib ipython pandas sympy
    nose
```

on a new Command Line window.

There is an additional package, `scipy`, that has installation issues with Windows.

Windows users must download the Numpy+MKL extension package and SciPy extension package from Cristoph Gohlke's website. Choose the appropriate version for your Python distribution (in this case the files that contain **cp27m** for Python 27) and for your operating system (**win32** for 32-bit and **win_amd64** for 64-bit)[1].

After downloading them, run

```
pip install numpy-1.13.1+mkl-cp27-cp27m-win32
pip install scipy-0.19.1-cp27-cp27m-win32
```

[1] Click here for help.

It is important that you install Numpy+MKL first, as SciPy depends on this package. If your installation was successful, open the Python interpreter (run `python` on the command line) and type

```
import numpy
import scipy
```

to verify the packages are properly installed and readable.

Mac

To install the stack execute:

```
pip install numpy scipy matplotlib ipython pandas
    sympy nose
```

on a new Terminal window.