Instructions: Getting Started With Python

Adapted from "*Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*" by Aurélien Géron

The goal of the following instructions is to get you ready with the **essential python libraries**. After going through the steps in this document, you should refer to Jupyter notebook series on "***Data Scientist's Handbook***".

Essential Libraries

NumPy: Base *n*-dimensional array package
SciPy: Fundamental library for scientific computing
Matplotlib: Comprehensive 2D/3D plotting
Pandas: Data structures and analysis
Scikit-Learn: Production-ready Python framework to implement many Machine Learning algorithms efficiently
TensorFlow: An open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning.
Keras: A high-level Deep Learning API. TensorFlow comes bundled with its own Keras implementation, tf.keras.

Windows Users: Use the Anaconda package manager to install Python (https://docs.anaconda.com/anaconda/install/windows/). This will install all necessary packages and give you a pre-configured Anaconda Terminal with all of the paths set. Mac and Linux users should follow the following instructions but are welcome to use Anaconda.

1. **Install Python**
https://www.python.org/

2. **Create the Workspace**

Next you need to create a workspace directory for your Machine Learning code and datasets. Open a terminal and type the following commands (after the $ prompts):

```
$ export ML_PATH="$HOME/ml"      # You can change the path if you prefer
$ mkdir -p $ML_PATH
```

3. **Install Python Modules**

You will need a number of Python modules: Jupyter, NumPy, Pandas, Matplotlib, Scikit-

Learn, TensorFlow 2, and Keras.

To install these modules, use Python's own packaging system "pip" (Windows uses may use conda). pip is a recursive acronym that can stand for either "Pip Installs Packages" or "Pip Installs Python". pip is a ***package management system*** used to install and manage software packages written in Python. One major advantage of pip is the ease of its command-line interface, which makes installing Python software packages as easy as issuing one command. Python 3.4 and later include pip (pip3 for Python 3) by default.

First check whether pip is already installed.

```
$ pip3 --version
```

You should make sure to have a recent version of pip installed, at the very least >1.4 to support binary module installation (a.k.a. wheels). To upgrade the pip module, type:

```
$ pip3 install --upgrade pip
```

Now you can install all the required modules and their dependencies using this simple pip command:

```
$ pip3 install --upgrade jupyter matplotlib numpy pandas scipy scikit-learn tensorflow keras
```

To check your installation, try to import every module like this:

```
$ python3 -c "import jupyter, matplotlib, numpy, pandas, scipy, sklearn"
```

After successful installation, explore these modules starting with NumPy and Pandas.

Note on NumPy: Make sure that you are well-versed in Linear Algebra operations with NumPy.

Note on Pandas: The Pandas library in built on NumPy. It is one of the most preferred tools for data scientists to do data manipulation and analysis, next to matplotlib for data visualization and NumPy, the fundamental library for scientific computing in Python on which Pandas was built. There should be no output and no error.

Pandas deals with the following three data structures: Series, DataFrame and Panel. These data structures are built on top of NumPy array, which means they are fast. Frequently you will use its **DataFrame** structure.

A DataFrame is a general 2D labeled, size-mutable tabular structure with potentially heterogeneously typed columns. Building and handling two or more dimensional arrays is a tedious task, burden is placed on the user to consider the orientation of the data set when writing functions. But using Pandas data structures, the mental effort of the user is reduced.

Note on Jupyter Notebook: The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no Internet access (as described in this document) or can be installed on a remote server and accessed through the Internet.

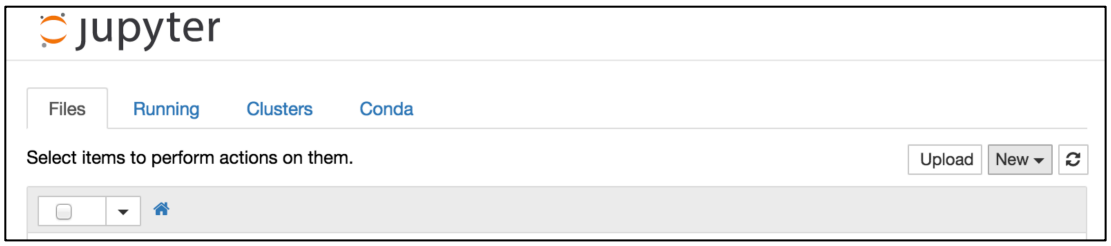
With this brief background on the Python modules, you are ready to fire up Jupyter by typing:

```
$ jupyter notebook
```

A Jupyter server is now running in your terminal, listening to port 8888. You can visit this server by opening your web browser to *http://localhost:8888/* (this usually happens automatically when the server starts). You should see your empty workspace directory.

4. **Create Python Notebook**

Create a new Python notebook by clicking on the "New" button and selecting the appropriate Python version.



This does three things.

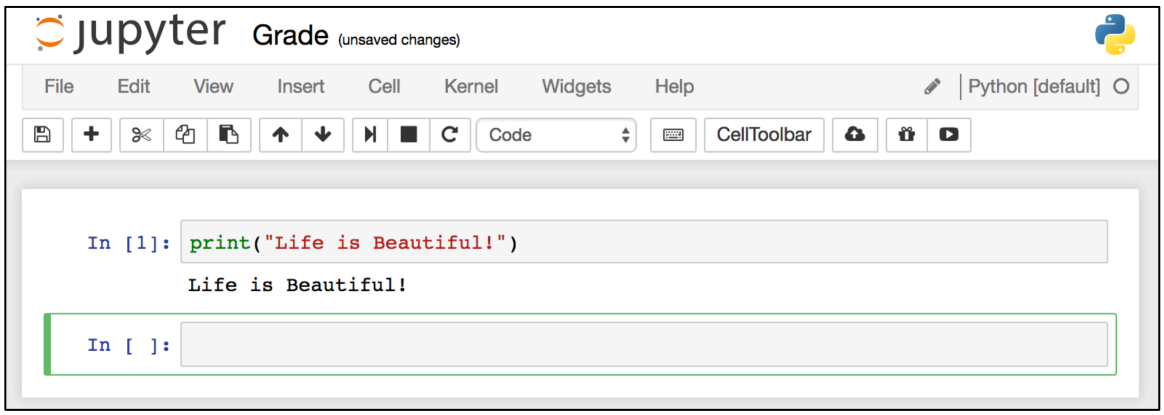
First, it creates a new notebook file called *Untitled.ipynb* in your workspace.

Second, it starts a Jupyter Python kernel to run this notebook.

Third, it opens this notebook in a new tab. You should start by renaming this notebook to, for example, "Grade" (this will automatically rename the file to *Grade.ipynb*) by clicking Untitled and typing the new name.

Note: A notebook contains a list of cells. Each cell can contain executable code or formatted text. Right now the notebook contains only one empty code cell, labeled "In [1]:". Try typing **print("Life is Beautiful!")** in the cell, and click on the play button or ***press Shift-Enter***. This sends the current cell to this notebook's Python kernel, which runs it and returns the output. The result is displayed below the cell, and since we reached the end of the notebook, a new cell is automatically created.

Go through the User Interface Tour from Jupyter's Help menu to learn the basics.



If you have finished up to this point, then turn to jupyter notebook series on "Data Scientist's Handbook".