Lab 01. Anaconda Installation and Importing Data Using Pandas

# Preface

This lab introduces steps of installing Anaconda and launching Jupyter Notebook.

# What is Python?

Python is a very-high-level programming language. It supports common elementary data types such as numbers and characters, as well as high level built-in data types such as sequences and dictionaries. It comes with many standard modules and third-party libraries. It is an interpreted language, which means you can run your Python statements or program directly without compilation. Python is freely available on Windows, Unix/Linux, and Mac OS. In short, Python is simple to learn yet very powerful and popular in data analysis and machine learning.

As an evolving programming language, Python has different versions. Python 2.x becomes legacy (yet popular), whilst Python 3.x is the current major version. This tutorial will use the latest stable version: **Python 3.6**.

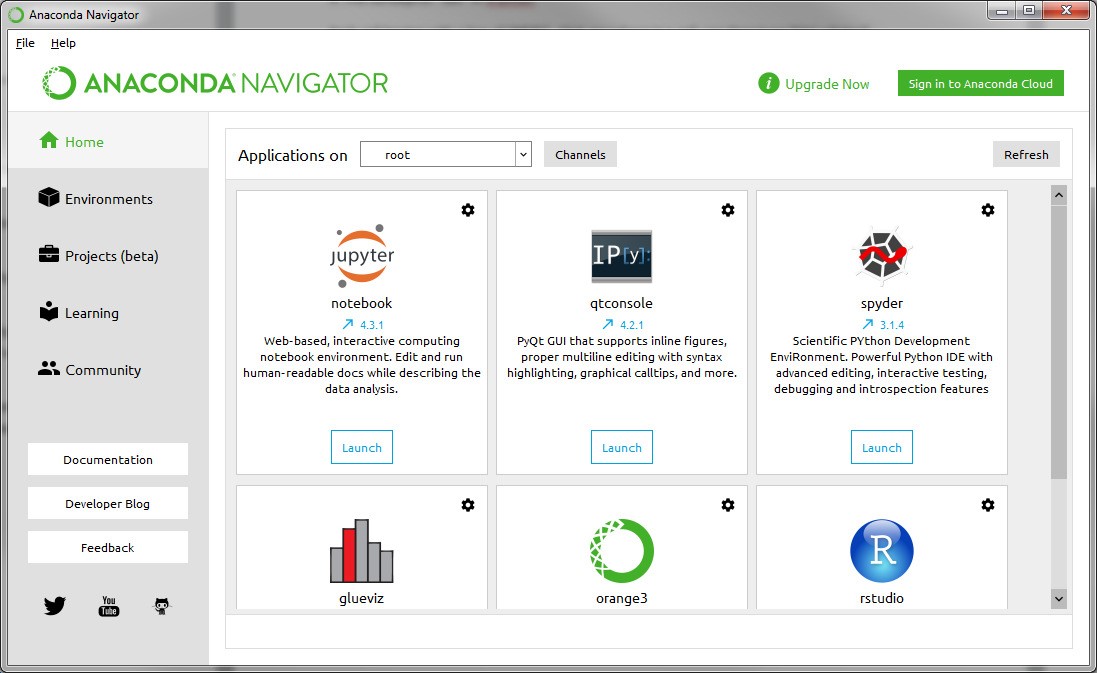
This tutorial will use **Anaconda**, and **Spyder** as the Python development environment.

* **Anaconda** ( https://anaconda.org/ ) is an open source software to manage Python development environments for Windows, Linux, and Mac OS. It provides easy management of a large collection of Python libraries.
* **Jupyter Notebook** ( <https://jupyter.org/> ) is an incredibly powerful tool for interactively developing and presenting data science projects. A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media.

Now let’s begin the journey of Python.

# Anaconda Installation and Basic Usage of Jupyter Notebook

**Step 1.** Open Anaconda Navigator. You should see a window similar to the following one:



There are five items on the left panel: Home, Environments, Projects (beta), Learning, and Community. Let’s introduce some of them:

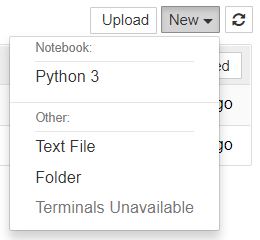
* Home: you can launch different IDEs to develop your Python programs, including Spyder.
* Environments: you can manage different development environments for your projects. E.g., the default root environment is using Python 3.6. If you also need to use Python 2.7 for another project, you can create a new environment that uses Python 2.7.
* Learning: You can find a plenty of learning materials and documentations about Python.

**Step 2.** Go back to “Home” and launch Jupyter Notebook; or On Windows, you can run Jupyter Notebook via the shortcut Anaconda adds to your start menu, which will open a new tab in your default web browser that should look something like the following screenshot.



Be aware that the dashboard will give you access only to the files and sub-folders contained within Jupyter's start-up directory; however, the start-up directory can be changed. It is also possible to start the dashboard on any system via the command prompt (or terminal on Unix systems) by entering the command “jupyter notebook”.

**Step 3.** Browse to the folder in which you would like to create your first notebook, click the "New" drop-down button in the top-right and select “Python 3”.



Your first Jupyter Notebook will open in new tab — each notebook uses its own tab because you can open multiple notebooks simultaneously. If you switch back to the dashboard, you will see the new file Untitled.ipynb and you should see some green text that tells you your notebook is running.

***So, What is an ipynb File?***

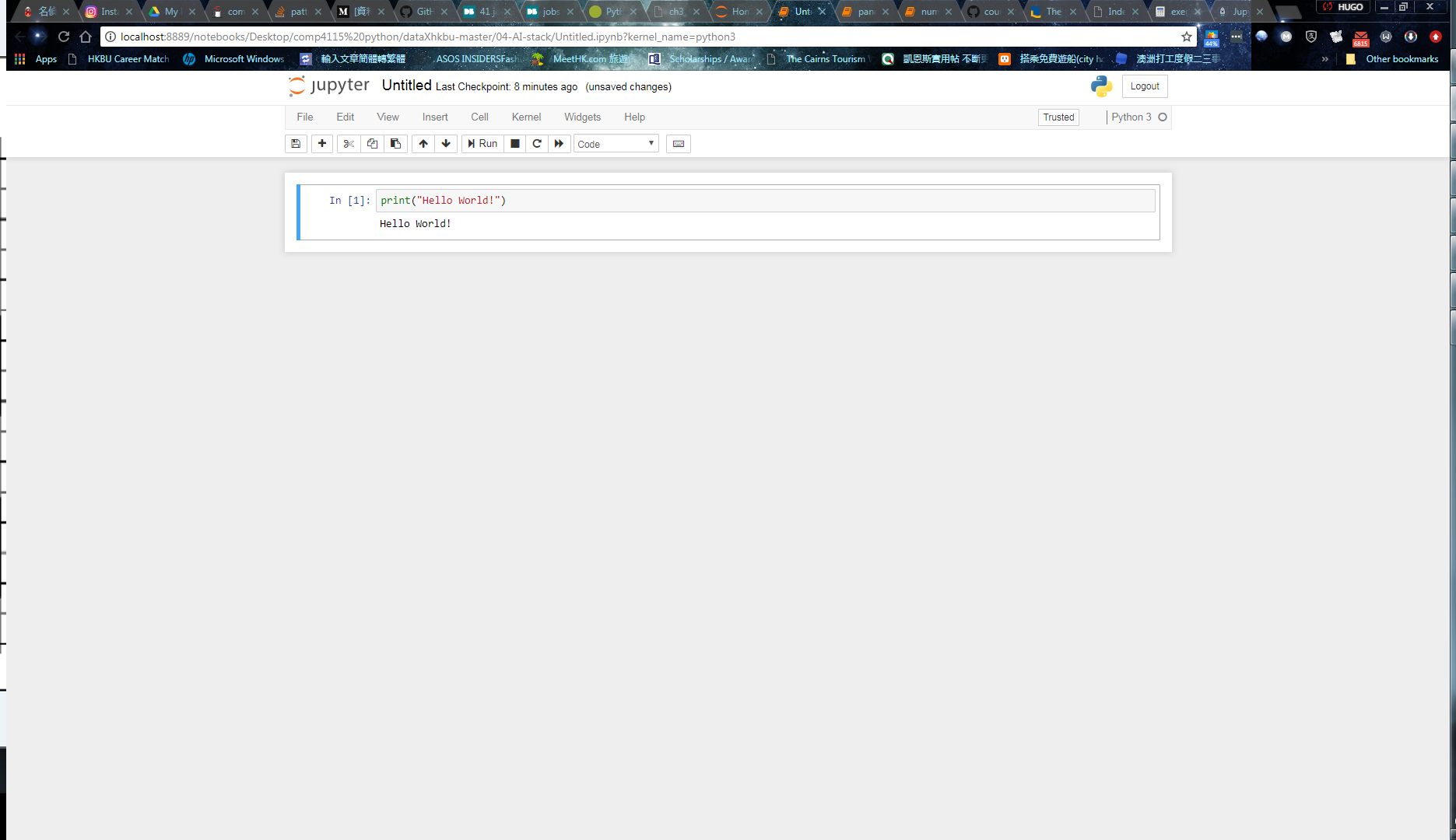
Each .ipynb file is a text file that describes the contents of your notebook in a format called JSON. Each cell and its contents, including image attachments that have been converted into strings of text, is listed therein along with some metadata.

There are two terms that you should notice: cells and kernels are both to understanding Jupyter and to what makes it more than just a word processor.

*A kernel is a "computational engine" that executes the code contained in a notebook document.*

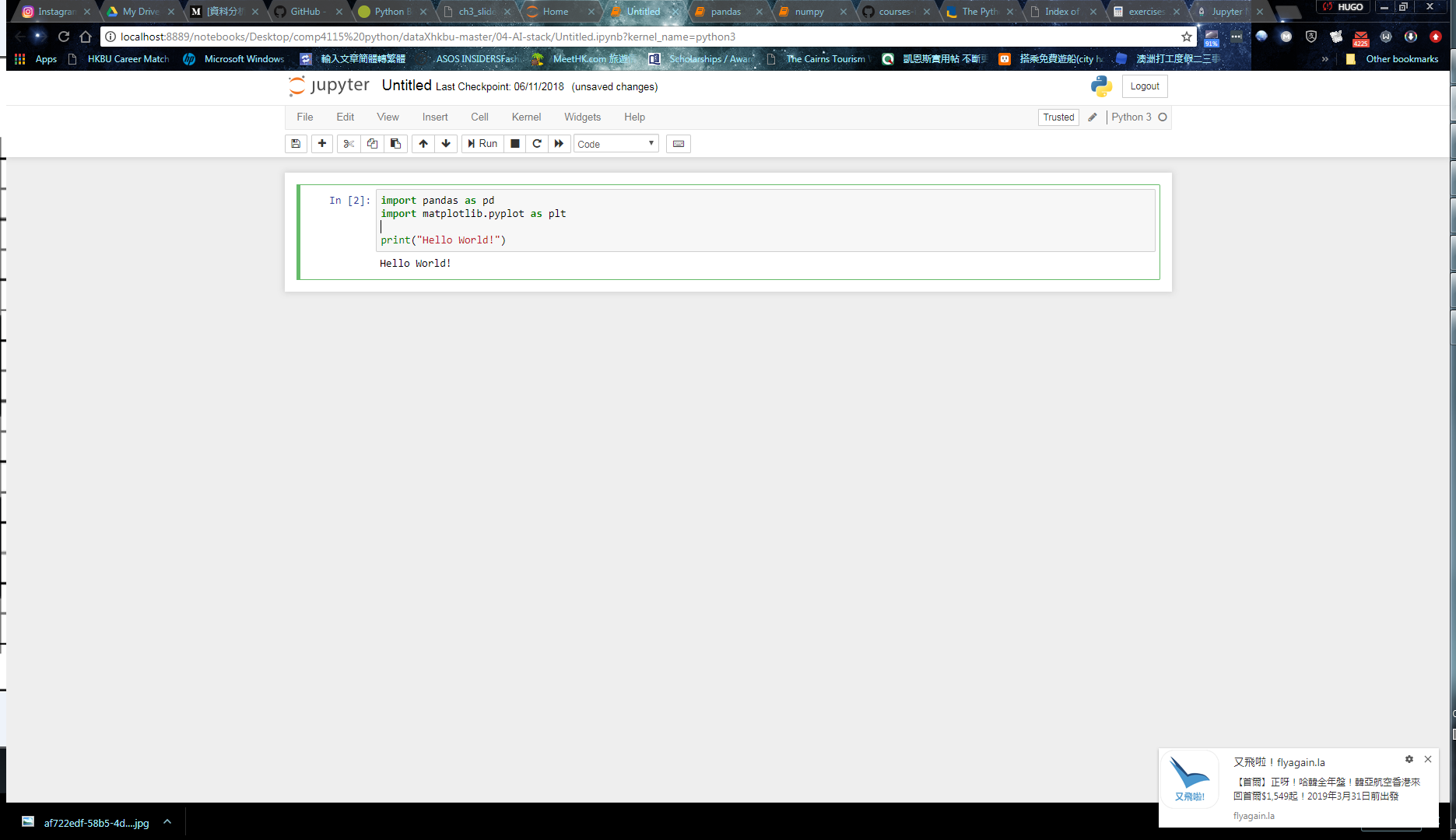
*A cell is a container for text to be displayed in the notebook or code to be executed by the notebook's kernel.*

**Step 4.** Type “print('Hello World!')” into the cell and click the run button Notebook Run Button in the toolbar above or press Ctrl + Enter. The result should look like this:



The label to its left will have changed from In [ ] to In [1]. The output of a code cell also forms part of the document. The "In" part of the label is simply short for "Input," while the label number indicates when the cell was executed on the kernel — in this case the cell was executed first. Run the cell again and the label will change to In [2] because now the cell was the second to be run on the kernel. It will become clearer why this is so useful later on when we take a closer look at kernels. Notice how Jupyter signifies that the cell is currently running by changing its label to In [\*].

It is common to start off with a code cell specifically for imports and setup, so that if you choose to add or change anything, you can simply edit and re-run the cell without causing any side-effects. For example, we import pandas to work with our data, Matplotlib to plot charts etc.



**Step 5.** Now, select a directory where the pandas.ipynb file located and open the file.