Advanced Python

Getting Started

Welcome! Since you can already write and run Python programs, at minimum you only need to download and open/run the course files, and make sure you have installed the modules we'll need for the course. However if you would like to refresh your setup, or use the same version of Python and the text editor we will be using in class, you may follow the Optional Steps.

Summary

Required Software: Overview

For the course, you must have installed and be able to use:

- a *Python distribution*. Any 3.8+ version can be used. If you need to install Python, I can recommend **Miniconda Python**. Additional modules must be installed (detailed below).
- a *text editor or IDE*. Use your preferred editor; I will be using **PyCharm** in class.
- **Jupyter Notebook**, another tool for writing and running programs.

These are the modules to install -- you may have them already:

- **requests**, a web client library
- **bs4** (Beautiful Soup), a web scraping library
- **pandas**, a data processing library
- **matplotlib**, a data visualization library
- **Jupyter notebook** (this is not a module but is needed for pandas)

Optional Steps: Installing Python and a Code Editor/IDE

We will be using Miniconda Python and the PyCharm IDE in class. If you would like to take these optional steps to install Miniconda Python and/or PyCharm, you may refer to the document titled "Installing Python and the PyCharm Editor/IDE".

However, any 3.8+ version of Python and any editor will work if you can already use them to open and run programs. (You must also have installed the modules noted above, or install them as described in "Installing Required Python Modules", below.)

Downloading and Installing the Course Lab Files

The lab files contain all of the inclass exercises and data files that we will use in this course.

1. Download and expand the zip file.

    a. Please visit the class website.

    b. At the top of the home page find the link **download source data** and click it. The download should begin automatically, usually to your **Downloads** directory. The file is called **python_data_apy.zip**.

    c. You may leave the zip file where it is, or you may move it to a more familiar location, but you should be able to navigate to it in an upcoming step.

    d. Unzip the file:

        i. Mac: double-click the file; a folder named **python_data_apy** should appear.

        ii. Windows: right-click the **.zip** file and select **Extract All…** and allow Windows to suggest the folder name **python_data_apy**. (Note that Windows may behave in varying ways when double-clicking, so I recommend this method.)

    The folder should have the following the following structure:

    ```
    python_data_apy/
       session_00_working_files/
       session_01_working_files/
       ..etc..
       session_10_working_files/
    ```

    Please make sure that your overall folder is named **python_data_apy/** and that there are 11 folders within.

2. Open the expanded folder in your IDE and run a test script.

    a. Use your IDE or editor to open the **python_data_apy/** folder so that it appears as a single project. (If you are using a simple editor instead of an IDE you can skip this step.)

    b. Expand the **session_00_working_files/** folder and run the **hello_version.py** script. You should see your Python installation's version number.

    c. If this does not work as expected (cannot run, or version number incorrect), you may get in touch with me for assistance, or if you would start over with Python and/or PyCharm, please refer to "Installing Python and the PyCharm IDE". This guide covers all the steps needed to install Python and PyCharm, and to connect PyCharm to Python. After following these steps, please proceed to the next section here.

Installing Required Python Modules

Windows: open a new Anaconda Prompt if you have it, or Command Prompt
Mac: open a new Terminal window

If you are using Miniconda or Anaconda, you should use the **conda** install utility to download and install these modules. Otherwise, you should use **pip**. Choose the appropriate column below and issue each command separately. Each command may take several seconds to minutes to complete; you'll see text output and you'll be asked to confirm each by typing **y**.

Miniconda / Anaconda	Other Python Distributions
`conda install requests`	`pip install requests`
`conda install pandas`	`pip install pandas`
`conda install matplotlib`	`pip install matplotlib`
`conda install bs4`	`pip install bs4`
`conda install jupyter`	`pip install jupyter`

If you are unable to run **conda** or **pip** please let me know.

Using Jupyter Notebook

Jupyter Notebook is another tool used to write and run Python - short snippets are run within individual *cells* inside a web browser. Its advantage over PyCharm is its ability to run snippets of code separately from other code in the notebook. You may find this tool to be useful for some of our in-class exercises, though you are not required to use it. **If you already know how to use Jupyter Notebook, you can skip this section.**

Please note that Jupyter should not be considered a replacement for PyCharm or any other full-blown Integrated Development Environment. Jupyter is used for experimentation, demonstration, tutorial and in-class exercises. **It should not be used to write Python applications or projects for this course.**

1. Launch Jupyter Notebook.
    a. Windows:
        i. In the Windows search box, type **jupyter**. Windows displays the selection **Jupyter Notebook (Anaconda3)**.
        ii. Click the selection **Jupyter Notebook (Anaconda3)**. Jupyter opens a Command Prompt window, then a browser window showing a list of files and the **jupyter** logo at the top left.
    b. Mac:
        i. Search for and launch **Terminal**.
        ii. At the prompt, type **jupyter lab**. A Terminal text window appears, and then your browser should launch displaying some icons and menu items.

2. View your filesystem through Jupyter and launch a Jupyter Notebook file.

    a. At the top of the left menu bar, click the folder icon. A file browser panel appears on the left. The base directory will be your home directory, or whatever your present working directory was in the Terminal when you launched jupyter.

    b. Using the file browser, navigate to the **python_data_apy** folder you unzipped earlier.

    c. Double-click the **session_00_working_files** link. Jupyter allows us to click around the filesystem folders through links.

    d. Double-click the **test_notebook.ipynb** file. The **.ipynb** extension denotes Jupyter Notebook files. Jupyter opens the notebook in the browser window.

3. Run the Jupyter Notebook cells. Each grey box you see (with **In [ ]** in front of each) is a Jupyter *cell*. Each cell can be run separately. At the same time, any cell that is run may affect the result of a subsequently-run cell.

    a. Click inside the first grey cell, and then type **Ctrl-[Enter]**. Jupyter displays the result of cell execution -- you should see **hello, jupyter!** This is the easiest way to execute a cell.

    b. Click inside the second grey cell and select **Cell > Run Cells**. Jupyter displays the result of the cell execution -- you should see **the number is 5**. This menu selection is another way to execute a cell.

    c. Click inside the third grey cell and type **Ctrl-[Enter]** to run the cell -- you should see the value **50**. Note two things:

        i. the value of **a** (assigned in the previous cell) was retained and is understood by this cell, showing that although each cell is run separately, the cells run *cumulatively*, meaning they are aware of what cells were run previously, as if the whole notebook was a single running program, that is still in the process of being executed (as you execute more cells).

        ii. The result of the last statement in the cell will be shown in the output. In other words, we don't necessarily need to print a value to see it in the output - we need only to reference the value at the end to see it.

    d. Save and checkpoint your notebook. You can save the notebook as an **.ipynb** file and send it to yourself or others.

        i. Click on **File > Save and checkpoint** to save the file.

        ii. Click on **File > Close and Halt** to stop the notebook and return to the files view.

    e. Clear memory and restart the notebook. As mentioned earlier, executed cells work together and all variables are remembered the notebook and accessible across all cells. For this reason, errors can occur that may be confusing if we re focusing on the code in

only one cell. To make sure that some other cell isn't affecting the cell we're working on, we can restart the notebook so all memory is cleared. Our code will not change.

Simply click the circular arrow in the menu bar to "restart the kernel". Confirm, and memory will be cleared. Note that any changes we made to the notebook will not be reset or erased.

f. Special note: don't ever call **exit()** from a notebook. You can **import sys** and call **sys.exit()** to simulate a program exit, although the notebook will continue to be available.

g. Please remember, Jupyter notebook should not be used for homework! It has significant quirks that can cause strange behaviors and create confusion. It is also not appropriate for project work, except for prototyping and testing. We will use Jupyter for exercises only.

You're a real (snake) Charmer!

Fantastic -- you're ready to continue your journey. See you in class! Please send any questions or feedback to me at **dbb212@nyu.edu**.