Configuring the Python Scientific Computation Environment

Introduction

Scientific computation in Python usually require the following packages:

1. NumPy, which provides the **`ndarray`** data structure along with a number of useful operators and algorithms.
2. Matplotlib, which depends on NumPy and provides graph plotting capabilities.
3. SciPy, which depends on NumPy for the **`ndarray`** data structure and implements several algorithms.
4. Pandas, which provides the **`DataFrame`** data structure and provides a features to explore, filter and transform tabular data along with several popular data input and output formats. It depends on NumPy for the **`ndarray`** data structure.
5. Jupyter Notebook, which provides a programming environment within the web browser.

Installing Python 3

While Python 3 can be downloaded from Python home page https://www.python.org, and in the case of GNU/Linux distributions from the operating system's package manager, this document presents an alternative approach which is the preferred way for Microsoft Windows users.

Let us download the minimal Python distribution of Anaconda Python called Miniconda 3. Unlike Anaconda Python distribution, Miniconda contains a minimal set of packages along with the Anaconda package manager **conda**. Having installed Miniconda 3, you can create a virtual environment and install all required packages into the virtual environment. This helps you keep multiple Python environments with different packages or different versions of the same packages and choose the virtual environment in which you wish to work. Check the following for your hardware and operating system: 32-bit or 64-bit version of Microsoft Windows 8/10.

Based on the Operating system, download the correct version of Python 3 for your operating system from https://docs.conda.io/en/latest/miniconda.html. Verify the SHA256 hash for the downloaded file.

Install the downloaded file and choose to install only for you and not for all users and choose to add the Python 3 installation to your **PATH** variable. After installation is complete, check that installation is correct. To do this, use the following steps:

1. Press the Windows key (⊞ or ⊞) and type Anaconda in the search bar. This will display the **Anaconda Prompt** menu item. Click on Anaconda Prompt to open the command window. You should see the prompt starting with the default environment named **`(base)`**.
2. Update the installed components. With the computer connected to the Internet, type the following command at the command prompt: **`(base) C:>conda update –all`** answer **yes** where asked and wait for all installed components to upgrade to the most recent version.

Creating a Virtual Environment

Choose a name for your virtual environment, say, **`scipy`**.

1. Create the virtual environment with the **conda** package manager with the command: **`(base) C:>conda create –n scipy`**.
2. Activate the newly created virtual environment **`scipy`** with the command: **`(base) C:>conda activate scipy`**. Now the prompt will change and begin with **`(scipy)`**, to indicate the current virtual environment.
3. Install all required packages into the scipy virtual environment with the command: **`(scipy) C:>conda install numpy matplotlib pandas notebook`**
4. To deactivate a virtual environment and return to the base environment, type the command: **`(scipy) C:>conda deactivate`**.

Starting Jupyter Notebook Server and Creating Jupyter Notebooks

Jupyter Notebook is a browser hosted *programming and documentation environment* with the Notebook web server running on the local machine. It provides the facility of writing and executing code and seeing the output within the web browser. It also permits writing documentation in *Markdown format* and can render LaTeX equations as part of documentation.

Procedure to use Jupyter Notebooks:

1. Open Anaconda Prompt and activate **`scipy`** virtual environment as described in the section on *Creating a Virtual Environment* by typing **`conda activate scipy`**.
2. Switch to a folder, using **`cd`** command, where you wish to store your Jupyter Notebooks. Create the folder if necessary.
3. Type the following command within the **`scipy`** virtual environment:
   ```
   (scipy) C:>jupyter notebook
   ```
4. Your default web browser must automatically open and display the Jupyter Notebook dashboard. If this does not happen, or if you wish to use a web browser other than the default, then open the web browser and enter the URL **`localhost:8888/tree`** into the address bar.
5. Create a new Notebook by clicking the **New** dropdown at the top right side of the dashboard and click on **Python 3**.
6. Rename the Notebook by clicking on the **Untitled** at top of the Notebook and typing an appropriate name for the Notebook and click **OK**.
7. Close the Notebook by clicking on File in the main menu and clicking om Close and Halt. You can switch to the dashboard and open a previously created Notebook or open an existing one.
8. To shutdown the Notebook serve, go to the command prompt from where the server was started and press **Ctlr+C** on the keyboard.

Working with Jupyter Notebooks

A Jupyter Notebook consists of **cells**. A cell can be one of the following types:

1. Code cells contain Python code that can be executed by pressing Ctrl+Enter or Chift+Enter.
2. Markdown cells can contain text written in the Markdown format. For more information about Mardown formatting, visit https://daringfireball.net/projects/markdown/. Visit https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet for a Markdown cheatsheet.
3. Heading Cells contain headings for text. You can create headings in Markdown cells with **#**, **##** etc.

Jupyter Notebook Keyboard Shortcuts

Jupyter Notebook can be in one of two modes – editing mode or command mode. Pressing the **Esc** key puts a Notebook into command mode and in command mode, you can use several keyboard shortcuts. If you don't prefer using keyboard shortcuts, please study the main menu and all options under each menu item. When in command mode, pressing the **Enter** key puts you in editing mode.

When your Notebook has multiple cells, the *current cell* is indicated by the thick coloured line at the left edge of the cell, *blue* if it is in command mode and *green* if it is in editing mode.

When in command mode, you can display the list of keyboard shortcuts by pressing **h** key on the keyboard (**Esc h** if in edit mode).

Using conda Package Manager

The **`conda`** package manager is a package manager and a virtual environment manager rolled into one. To learn how to use **`conda`**, type **`conda help`** at the Anaconda Prompt. The commonly used **`conda`** commands are:

	Command	**Description**
1.	`conda --help`	Print command line help
2.	`conda install --help`	Print help on the `conda` command `install`
3.	`conda update –all`	Update all installed packages in the current environment
4.	`conda update numpy`	Update the installed package, in the current virtual environment, named `numpy`
5.	`conda list`	List all installed packages in the current virtual environment and their versions
6.	`conda list matplotlib`	List installed package in the current virtual environment named `matplotlib`
7.	`conda list mat`	List installed packages in the current virtual environment containing the pattern `mat` in their name
8.	`conda search numpy`	Search Anaconda repository for a package named `numpy`
9.	`conda search mat`	Search Anaconda repository for a package containing the pattern `mat` in its name
10.	`conda install numpy`	Install a package named `numpy` into the current virtual environment from Anaconda repository, along with all its dependencies
11.	`conda install x y z`	Install packages named `x`, `y` and `z` from Anaconda repository in the current virtual environment, along with all their dependencies
12.	`conda update x y z`	Update packages named `x`, `y` and `z` in the current virtual environment from Anaconda repository, along with all their dependencies
13.	`conda update --all`	Update all installed packages, in the current virtual environment, from Anaconda repository, along with all their dependencies
14.	`conda create –n scipy`	Create a new virtual environment named `scipy`
15.	`conda activate scipy`	Activate an existing virtual environment
16.	`conda deactivate`	Deactivate the current virtual environment and return to the `base` virtual environment
17.	`conda install numpy –n scipy`	Install the package `numpy` in the virtual environment `scipy` even if the current virtual environment is not `scipy`
18.	`conda update numpy –n scipy`	Update the package `numpy` in the virtual environment `scipy` even if the current virtual environment is not `scipy`
19.	`conda env list`	List all existing virtual environments
20.	`conda env remove –n scipy`	Remove an existing virtual environment named `scipy` and all packages installed in it