IPython Cheat Sheet

Information compiled by J.Vitrià, Universitat de Barcelona, July, 2014. In order to install and work with the IPython environment, you must follow these steps:

- Install Anaconda (only the first time).
- Open up a terminal/command promt.
- `cd` to your working directory.
- Run the following command: `ipython notebook`.
- Open in your browser an existing notebook or create a new one.

Anaconda

`http://docs.continuum.io/anaconda/`

What is Anaconda? - Anaconda is a free collection of powerful packages for Python that enables large-scale data management, analysis, and visualization for Business Intelligence, Scientific Analysis, Engineering, Machine Learning, and more.

Product Specifications - Anaconda is completely free to use and distribute.

Supported Platforms - Linux and Windows (64-bit and 32-bit); Mac OS X (64-bit).

Linux Install - After downloading the Anaconda installer from `http://continuum.io/downloads`, run the following command from a terminal:

```
bash <downloaded file>
```

After accepting the license terms, you will be asked to specify the install location (which defaults to ∼`/anaconda`).
Note: You do NOT need root privileges to install Anaconda, if you select a user writable install location, such as ∼`/anaconda`.
After the self extraction is finished, you should add the anaconda binary directory to your PATH environment variable.
As all of Anaconda is contained in a single directory, uninstalling Anaconda is easy (you simply remove the entire install location directory):

```
rm -rf ~/anaconda
```

Windows Install - After downloading the Anaconda installer from `http://continuum.io/downloads`, double click on the installer application icon and run it. Follow the instructions in the installer.
Note: If you encounter any issues, please try disabling your antivirus software.
As of version 1.4, Anaconda supports uninstall using the standard Windows mechanism. Click on "Add or remove Program" in the Control Panel, and select "Python 2.7 (Anaconda)".

IPython

`http://ipython.org/`

What is IPython? - IPython is an interactive shell for the Python programming language that offers enhanced introspection, additional shell syntax, tab completion and rich history.

IPython Install - On Windows and Mac, first download and install Anaconda. Then, update IPython to the current version, using the Terminal/Command prompt:

```
conda update conda
conda update ipython
```

On Ubuntu or other Debian-based distributions, type at the shell:

```
sudo apt-get install ipython-notebook
```

On Fedora 18 and newer related distributions, use:

```
sudo yum install python-ipython-notebook
```

Running IPython Notebook - You can work with Python either directly via the interactive Python console, or by writing and running Python programs using an IDE.

However, there are other ways to work with Python. IPython is a set of tools originally developed to make it easier for scientists to work with Python and data. It allows you to combine interactive Python exploration with prewritten programs and even text and equations for documentation.

For Mac and Linux users, open up your terminal. Windows users need to open up their Command Prompt. On Windows, you can find a launcher for IPython Notebook under Anaconda in the Start menu.

Change directories in the terminal (using the `cd` command) to the working directory where you want to store your IPython Notebook data.
To run IPython Notebook, enter the following command:

```
ipython notebook
```

It may take a minute or two to set itself up, but eventually IPython Notebook will open in your default web browser. Even though you're interacting with IPython Notebook using your browser, Notebook is running right there on your computer in that window. Only you have access to it.

First Steps with Notebook - In your browser, click the "New Notebook"button and a new notebook will open up. The empty box at the top is the first 'cell' for entering Python code. Try typing something like `print('Hello World')` into the cell. To run the code in the cell and see the output, click the Run button (play icon) on the toolbar, or type Shift-Enter. You'll see that whenever you run a cell, a new empty cell appears where you can enter another set of Python statements.

IPython Notebook

`http://ipython.org/`

IPython Notebook Directory - When IPython Notebook starts up it prints a line like this in your terminal: `[NotebookApp] Serving notebooks from C:\Anaconda`
This is the directory that it was started from, and it's the working directory for loading Python files, data files, etc. It can be helpful to make sure this directory is the directory where you plan to keep files related to your work. On Windows with Anaconda you can use a directory that is called 'IPython Notebooks' inside 'My Documents', and is created when you install Anaconda. On OS X, Linux or other Windows installations it's up to you which directory you use.

If you're launching from the command line, you can 'cd' to this directory before you launch Notebook. If you're using the command line on Windows, you can use Explorer to find your directory, then hold shift+right click. The context menu that comes up should have an option to "Start a command prompt here".

Loading Notebook Files - You can also load IPython Notebooks that you or other people have created, saved as IPython Notebook files (File extension `.ipynb`.)

After you download the Notebook file, move it into your IPython Notebook working directory and then choose `File -> Open` in Notebook to open it.

Loading Python Files - You can also load a pre-existing Python file into an IPython Notebook cell by typing

```
%load "myprogram.py"
```

into a cell and running it. This loads up a new cell containing the contents of `myprogram.py`.

There is one other useful built-in tool for working with Python files:

```
%run "myprogram.py"
```

This will run `myprogram.py` and load the output into a Notebook cell.

IPython Notebook Commands

`http://ipython.org/ipython-doc/stable/interactive/tutorial.html`
`http://damontallen.github.io/IPython-quick-ref-sheets/`

Tab completion - Tab completion, especially for attributes, is a convenient way to explore the structure of any object you're dealing with.

To use completion, type a pattern you would like the shell to match, followed by the Tab key.

Simply type `object_name.` to view the object's attributes

Besides Python objects and keywords, tab completion also works on file and directory names.

Macros - IPython macros are great for executing the same code over and over. Macros allow a user to associate a name with a section of Python code so the code can be run later by referring to the name. They are editable via the `%edit` magic command.

Using the Python debugger - The Python debugger (pdb) is a powerful interactive debugger which allows you to step through code, set breakpoints, watch variables, etc. With automatic pdb calling enabled, the Python debugger will start automatically when Python encounters an unhandled exception. The current line in the debugger will be the line of code on which the exception occurred.

If you start IPython with the `--pdb` option then you can call the Python pdb debugger every time your code triggers an uncaught exception. This feature can also be toggled at any time with the `%pdb` magic command.

Magic commands - IPython 'magic' commands are conventionally prefaced by %, but if the flag `%automagic` is set to on (which is default), then one can call magic commands without the preceding %. IPython checks if the command that you typed against its list of magic keywords. If the command is a magic keyword, IPython knows what to do with it. If it's not a magic keyword, it lets Python figure out to do with it.

lsmagic List all built in commands, called magic commands. These are prefixed with % to differentiate between variables if they have the same name.

%quickref Shows wich 'magic' commands that are availble in IPython.

? The '?' is very useful. If you type in ? after a function name, you will see the documentation about the function. Typing ? after a name will give you information about the object attached to that name,

%reset Resets the interactive environment.

%hist Allows you to see any part of your input history.

%paste Use text that you have in the clipboard, for example if you have copied code with Ctrl+C. The command cleans up certain characters and tries to find out how the code should be formatted.

%edit The `%edit` command (and its alias `%ed`) will invoke the editor set in your environment as EDITOR.

%%writefile 'file.py' The `%%writefile` magic is a very useful tool that writes the cell contents as a named file.

%who This function list objects, functions, etc. that have been added in the current namespace, as well as modules that have been imported.

System shell access - Any input line beginning with a `!` character is passed verbatim (minus the !) to the underlying operating system. For example, typing `!dir` will run 'dir' in the current directory.

Spyder

`https://code.google.com/p/spyderlib/`

What is Spyder? - Spyder is a powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features.

Install - You can easily install Spyder on all major platforms through Anaconda.

Opening Spyder - To open Spyder, enter the following command in a terminal/command prompt:

```
1  spyder
```

When it opens up, there are three different parts in the screen: the console,the editor and the inspector/explorer.

Console - The Console, on the bottom right. This is where you can enter, interact with and visualize data, inside a command interpreter.

Inpector - There are various inspectors and explorers on the top right. These are:

- Object Inspector – This automatically shows documentation of any function that you call. For example, if you type `sin(`, then as soon as you type the open bracket '(', the object explorer will tell you all about the 'sine' function available in NumPy. While this gives useful information, it can sometimes give (a lot) more detail than you need, especially when starting out.
- Variable Explorer – This keeps track of all of the variables that you have defined. As you define other variables, they will be visible here.
- File Explorer – This is similar to the usual file manager on a computer and it lets you explore and open the files in your directories. Note that the default setting is to only let you see python files.

Editor - The Editor is on the left hand side. This is a code editor where you can open, explore and write code (in .py files). It is also straightforward to run codes that you have written in the editor.