Anaconda Starter

Last Edited Sep 18, 2023

**Introduction**

[Anaconda](anaconda.com) is an open-source distribution of Python (and R, another programming language) and a suite of useful packages. It integrates a number of features that are used in the INMAS educational content, including an interactive user interface (jupyter) for running notebooks and installing Python packages.

In this brief tutorial we will show how to install Anaconda, how to verify a proper installation, and how to run the first jupyter notebook required for Workshop 1.

*Note: If you already have a preferred Python setup that allows you to run a jupyter notebook, we recommend that you use the latest version of Anaconda as many packages might have evolved since the version you have and not be compatible with the notebooks we developed.*

***Installation***

A screenshot of a computer

Description automatically generatedInstallation of Anaconda is relatively straightforward, just go to this [link](http://anaconda.com), and click the download icon A green and black sign

Description automatically generated which should automatically detect your operating system. Else, you could look for the *Anaconda installers* box at the bottom of the web page. At the time of this writing, the latest version is Anaconda 23.7.4 or 2023.07.2 using Python 3.11.

*Upgrade*

If you already have a version of Anaconda installed, it is recommended that you upgrade it to the latest version. The simplest way to update is to uninstall your current version of Anaconda and start with a fresh install as described above, as updating using conda will not update your version of Python but rather will require the creation of a whole new environment based on the newer Python version.

Checking the version of Anaconda and Python is relatively simple. Open a terminal window (anaconda on Windows, Terminal on Mac and Linux – more details below) and just type:

A computer screen with white text

Description automatically generatedconda info

You should then get something like this:

Make sure that the versions of both Python and Anaconda match the version described above and in this window.

***Opening the Navigator***

Following a successful installation, you should be able to open the Anaconda navigator.

*Windows Machine*

Click on the Windows Icon and search through the applications installed on your computer and open via the Anaconda Icon, which should look like the icon on the right.

*MacOS*

Open your Finder and go to the “Applications” Folder, look for “Anaconda-Navigator”, with the same icon you see above.

*Linux*

Open a terminal window and type anaconda-navigator.

Upon successfully opening the navigator you should see something resembling the following window.

A screenshot of a computer

Description automatically generated

The navigator can be used to start many applications. Those of interest to us are mainly the jupyter notebook and a terminal shell. The advantage of starting a command shell from the navigator is that all the environment variables are already pre-configured for Python. On Windows, we recommend using the PowerShell Prompt over CMD, as the former provides command line editing and history and is therefore more user friendly.

***Navigating using a command shell***

It is sometimes easier to use command line interface over a graphical user interface. Starting a terminal can be done through the anaconda navigator, through starting a terminal on Mac or Linux. Navigating files and directories consists of using the following five basic commands:

* ls list file in current directory
* cd change directory
* pwd print working directory
* mkdir make a new directory
* rmdir remove a directory

Note that when navigating, ‘.’ means the current directory, and ‘..’ means the parent directory. Therefore,

cd .

is a null operator, while

cd ..

moves to the parent directory. Please take a moment to navigate around the directory tree of your computer and familiarize yourself with these commands.

Help on any of these commands can be obtained by typing, for the command ls, for example,

man ls

where man stands for manual.

Files can be copied (cp), moved (mv), or deleted (rm). Use man to get more information on these commands.

Using a command line interface is often the preferred method for navigating directories, or projects with multiple files, especially when one is using a version control system such as git.

***Testing your anaconda installation***

First start a terminal shell from the Anaconda Navigator or otherwise. Then type

conda info

and make sure that the version is 23.7.4 with Python being 3.11.4.final. The command

conda list

will list all the packages installed on your computer and their version numbers. Notice that the packages contain matplotlib, numpy, pandas, and seaborn packages that we will learn about and use during the training. A successful installation should have a large number of packages available (including those mentioned) and should have the proper versions listed above.

***Downloading the notebooks***

The notebooks for the Workshop are available from github. You can download the material for the workshop at:

<https://github.com/INMAS-Math/Workshop-01.git>

Click on the code icon to get a zip file containing all the material. Alternatively, you can use a command line and git to clone the code as follows:

git clone https://github.com/INMAS-Math/Workshop-01.git

A screenshot of a notebook

Description automatically generated***Opening and running a jupyter notebook***

Start the anaconda navigator and launch the jupyter notebook interpreter by clicking on the Launch button of the icon shown on the right. The jupyter interpreter will start a virtual web server on your computer (localhost) that will be visualized through your web browser.

The opening page will show the files on your computer. By clicking on the directories, navigate to where you saved the notebooks from Workshop1 of INMAS. As before, ‘..’ stands for returning to the parent directory. Note that unlike other navigators (e.g., explorer) a single click is necessary to open a directory.

Jupyter notebooks end with the ‘.ipynb’ file extension. Clicking on the desired file will open the jupyter notebook interpreter of that file in a different tab of your browser. These notebooks contain intructions as well as code that can be edited and run. Each box, called cells, starting with the ‘In [ ]’ keywork is a block of code input waiting to be run. Other boxes, called markdown cells, are meant to give instructions. The number between brackets keeps track of the instruction numbers, showing the sequence of the instructions that were run, possibly out of order. A cell generating output will have an ‘Out[ ]’ with an associated sequence number in the brackets. An entire notebook can be reset by restarting the kernel and clearing all output. This is done through selecting ‘Restart and Clear Output’ under the ‘Kernel’ tab.

A screenshot of a computer

Description automatically generated

To run a block of code, click on the box and the surrounding perimeter of that box will turn green, indicating that it is ready to run with hitting Crtl-Enter. Alternatively, Shift-Enter will achieve the same result, except the mouse focus will jump to the next cell. Double-clicking on a markdown cell will switch the cell to the editing mode and show the raw (markdown) code for the text. Running the cell (Ctrl-Enter) will (re-)generate the nice text. Code cells, on the other end, can be edited directly once the mouse focus is on that cell (i.e., there is no need to double-click to edit those cells).

The menu at the top of the page contains the main command and will indicate the type of cell where the mouse is focusing. Clicking on run is another way to run the current cell, equivalent to Shift-Enter. It is also possible to run all the cells in the notebook by selecting ‘Run All’ under the ‘Cell’ heading. Notice how all the shortcuts are in these menus. Moreover, the ‘Help’ tab contains a ‘User Interface Tour’, as well as reference to the Python langage and its most popular packages such as pandas and NumPy, for example.

*Additional considerations on Windows*

On recent versions of Windows, Controlled folder acces might deny access to Python, resulting in the autosave feature of jupyter to be broken. If you receive a message in jupyter stating that the file you are using does not exist, the Controlled folder access feature of Windows is probably enabled. To fix, open Windows Security -> virus and threat protection -> virus and threat settings -> Controlled folder access -> Manage Controlled folder access -> Allow an app through Controlled folder access -> (click yes) -> Add an allowed app -> Recently blocked app -> +add python.exe. Alternatively, you can disable Control access folder for the duration of the training.

***A Note on File Paths in Python***

File paths have a different dialect depending on the operating system that you are using. In Linux and OS-X, all file paths start from the root directory which is referred to by “/”. Each level of subdirectory from there is separated by an additional “/”. For example:



Reference to a directory can start with “/”, in which case it is called an absolute reference. In other instances, it can start from where you are located (your current working directory) and start without a “/”. This is the case for the parent directory “../” for example.

In Windows, an absolute reference starts with the disk letter followed by a colon (e.g., C:, D:, etc). For example, “C:\Users\jane\Documents\somefile.txt”.

You can lookup a file’s full path and file name through its “properties”:

* On Windows, right click on the file name in the file explorer and click “Properties” (alternatively, Alt-Enter after highlighting the filename). For a file called “myfile.py” in directory “C:\Users\jane\Documents\scripts”, the full path in Python would need to be “C:\Users\jane\Documents\scripts\myfile.py”.
* On Mac, right-click on the file and select "Get Info". Alternatively, "Command + i"

summons up the Get Info panel. Using the same example, the full path in Python would need to be entered as “/Users/jane/Documents/scripts/myfile.py”.

Python, however, allows the usage of ‘/’ on Windows, for portability reasons. Therefore, the path in the following example is perfectly valid. We recommend that you use forward slashes “/” over backslashes “\” when using Python on Windows. Another reason to use a forward slash is that the backslash character is an escape character when used inside a string (a dialect inhirited from the C programming language).

