Installation_notes

February 28, 2022

For any question please contact us: - Matteo Bonetti (matteo.bonetti@unimib.it, matteo.bonetti@uninsubria.it) - Elisa Bortolas (elisa.bortolas@unimib.it, elisa.bortolas@uninsubria.it)

1 INSTALLATION NOTES

2 Operating System (OS)

2.1 Linux OS and its filesystem

In order to work proficiently with Python and much other scientific software, a Linux (or at least Unix-like) operating system is generally recommended. You can choose one among several distributions: Ubuntu, Fedora, Centos, OpenSuse... (but if you still want to use Windows see next section). Among Linux distributions, Ubuntu is probably the most user-friendly, and we recommend it. See here installation guides: - https://askubuntu.com/questions/1031993/how-to-install-ubuntu-18-04-alongside-windows-10 - https://hackernoon.com/installing-ubuntu-18-04-along-with-windows-10-dual-boot-installation-for-deep-learning-f4cd91b58557 - https://wiki.ubuntu-it.org/Installazione/InstallareUbuntu - https://itsfoss.com/install-ubuntu/

If you are using MAC, a Unix shell should already be present. You can install a terminal to access it, such as Aquaterm or xterm.

- in Linux the whole filesytem starts with the root directory, represented by character / . Everything is contained inside the root. / contains several important folders for the operating system. Among these folders a particular one is your home directory /home/username (see: the absolute path begins with /), where username is your name. This directory is special because here you can create, modify and delete everything you want. Instead, any action (e.g. installation of libraries, compilers, "programming languages" etc.) that involves any other folder in / requires the sudo keyword (super user do) and the user password specified before the command.

IMPORTANT: ANY ACTION PERFORMED ON DIRECTORIES DIFFERENT THAN THE HOME CAN POTENTIALLY DESTROY YOUR LINUX INSTALLATION, SO BE CAREFUL!

- here a complete list of linux commands https://maker.pro/linux/tutorial/basic-linux-commands-for-beginners , the most used are:
  - ls – to list file in the current directory
  - pwd – the absolute path of the current folder
  - cp – to copy files
  - mv – to move or rename files

  - mkdir – to create a folder
  - rm – to remove files (be careful: a file removed with rm is lost, no way to revover it!)
  - less – to display file content
  - tar – to create/extract compressed archives (see here some basic examples https://www.howtogeek.com/248780/how-to-compress-and-extract-files-using-the-tar-command-on-linux/)
  - zip – to create/extract compressed zip archives (e.g. zip -r my_zip.zip files)

- Ubuntu packages can be installed through the following command (similar package managers are often present on other linux distributions):

  ```
  sudo apt install package_name
  ```

  Recommended tools (some could be already installed):

  - C (gcc) and C++ (g++) compilers
  - text editors
    - graphical editors: gedit, geany (it also has other features), atom, sublime text, vscode and many others
    - terminal editors: vi, vim, nano, emacs
  - latex: you can choose between a minimal OR a complete installtion (https://www.tug.org/texlive/debian.html)

    ```
    sudo apt install texlive
    sudo apt install texlive-full
    ```

    alternatively if you don't want to install on your PC, you can use Overleaf (https://www.overleaf.com/), an online latex editor

2.2 Linux subsystem on Windows

If you would like to have a linux terminal on Windows OS, you can install the Linux subsystem on Windows (LSW). Visit https://docs.microsoft.com/en-us/windows/wsl/install-win10 for details. With the following steps you should be able to install it:

- Open PowerShell (right click on the start button and select Windows PowerShell(Administrator)) and insert following code:

  ```
  Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
  ```

- Restart your PC
- Open the Microsoft Store and choose your favorite Linux distribution (Ubuntu is the most user-friendly)
- Once installed, lauch the app (a terminal will appear) and complete the Linux installtion (choose user name, password etc.)
- Now you have a full Linux filesystem installed on Windows! You can install linux packages through the standard command:

  ```
  sudo apt install package_name
  ```

before any new installtion we suggest to run the following command to update the ubuntu repositories (it can take a while):

```
sudo apt update && sudo apt-get upgrade
```

- To access all directory under the Windows filesystem you have to go to /mnt/ where you will find the standard C drive and inside it the whole Windows system, i.e. **/mnt/c/Users**
- OPTIONAL: We will usually work on a Jupyter notebook (see next, don't worry), but if you want to use an independent graphical interterface you need to follow the next steps (see also https://stackoverflow.com/questions/43397162/show-matplotlib-plots-in-ubuntu-windows-subsystem-for-linux):
  - Google "xming" (x11 server) and download (from sourceforge) / install / run
  - sudo apt-get update && sudo apt-get install x11-apps
  - sudo apt-get install gnome-calculator (to get GTK)
  - sudo apt-get install qtbase5-dev (for qt backend of pyplot)
  - sudo apt-get install python3-tk
  - finally add the following line at the end of the file ~/.bashrc (this will make this instruction permanent. .bashrc is a hidden file in your home directory and contains some standard instructions, dont't delete them when adding the new line!):

    ```
    export DISPLAY=localhost:0.0
    ```

  REMEMBER TO RUN THE "xming" app WHEN USING A TERMINAL THAT PRODUCES GRAPHICAL OUTPUTS (you may choose to activate xmingat PC boot)

2.3 Virtualbox on Windows

VirtualBox is a tool for the creation and management of guest virtual machines running Windows, Linux, BSD, OS/2, Solaris, etc. Practically you can emulate a linux distribution on you Windows PC (but beware, it can be slow!). You can download from https://www.virtualbox.org/

See here https://www.virtualbox.org/manual/UserManual.html for the complete manual and here https://www.nakivo.com/blog/use-virtualbox-quick-overview/

If you still want to use only Python feature on Windows see next.

3 Python: installation

3.1 Ubuntu:

Python is a programming language. Usually any Linux distribution comes with a version of Python already installed. The version of Python we will employ is **Python3**. On Ubuntu, to check that Python3 is actually installed on your system open a terminal and type

```
python3
```

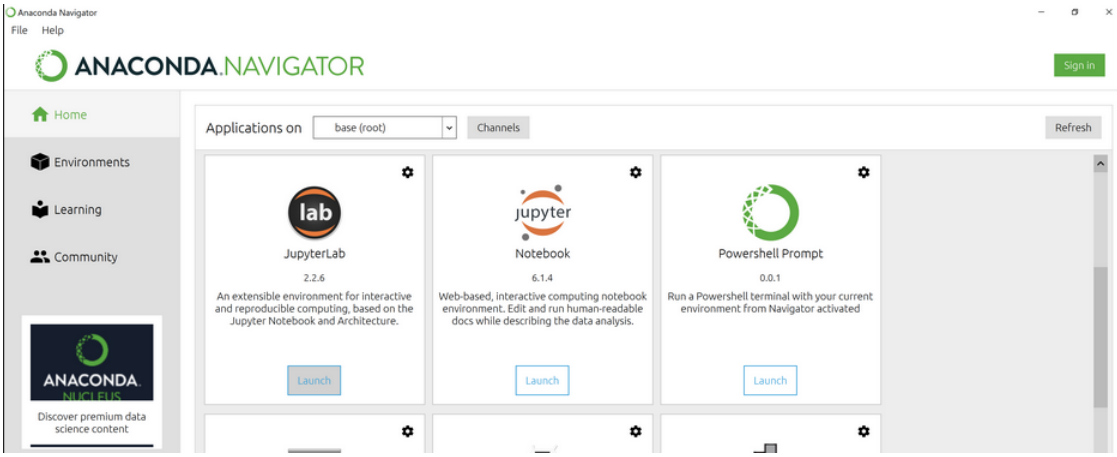
if you get an error, Python3 may not be installed. You can install it through the standard apt-get command

```
sudo apt install python3
```

3.2 Windows:

There are few ways in which you can have a running Python on your Windows PC. The most straightforward and most self-contained one is Anaconda Navigator (https://www.anaconda.com/), which already contains all the software you need. You can download the installer from here https://www.anaconda.com/products/individual and you can follow the instructions here https://docs.anaconda.com/anaconda/install/windows/

Here https://docs.anaconda.com/anaconda/user-guide/ you can find a quickstart guide and here https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf a cheat sheet.



4 Python: modules and packages

4.1 Python package manager

Python has many packages, i.e. a lot of tools programmed by many code developers that make our life simpler. The easiest way to install packages is through a package manager (same concept of `sudo apt install` of Linux). The official package manager of Python is `pip`, which collects packages from the Python Package Index (PyPI, https://pypi.org/).

- How to install pip (if not already present):
  - visit https://pip.pypa.io/en/stable/installation/
  - click on the get-pip.py link to download get-pip.py or copy and paste on a terminal

    ```
    curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
    ```
  - type on the terminal (be sure to be in the same directory in which get-pip.py has been saved):

    ```
    python3 get-pip.py
    ```

    (if you get an error about distutils run the command: sudo apt install python3-distutils)
  - upgrade to lastest version (if needed):

```
pip3 install -U pip3
```

(if pip3 cannot be found try to use `pip` instead of `pip3`, this holds also for the following instructions)

- Now pip can be used to install any python packages:

```
pip3 install package_name
```

`Anaconda` is another way of dealing with Python packages (see here https://docs.anaconda.com/anaconda/ to install it for a Linux distribution and learn how to use). `Anaconda` has its own repositories but it works also together with `pip`. To install packages from anaconda just type

```
conda install package_name
```

and if the package is available it will be installed (see `conda` documentation https://conda.io/en/latest/). A powerful aspect of `Ananconda` is the environment creation. You can think about an environment as a closed box in which a specific Python installation lies. Different environments can contain different packages or even different versions of Python! You can create environments typing (see here for complete documentation https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html?highlight=environment):

```
conda create -n myenv python=3.6 scipy=0.15.0 numpy
```

For instance here we are creating an environment in which the Python version is the 3.6 and we are using the scipy package (version 0.15.0) and the numpy one (latest stable version). To activate the environment type in your terminal the command:

```
conda activate myenv
```

and deactivate it with

```
conda deactivate
```

4.1.1 List of frequently used packages

The basic and most useful packages are (see next for other details):

- jupyter (handy and easy-to-use tool to write and execute python code using a browser as interface)
- numpy (optimised math, arrays and other stuff)
- matplotlib (graphical library for plots etc.)
- scipy (scientific library)

Some additional, useful packages: - astropy (astronomical tools) - pandas (table and database manipulation) - sympy (symbolic math)

To employ the tools that constitute the above packages you then need a way to tell your program how to access the code.

5 Jupyter

The Jupyter Notebook (http://jupyter.org/) is an opensource application that supports several programming languages. With Jupyter you can create and share documents that contain live code, equations, visualizations and text (see here for a nice tutorial https://www.dataquest.io/blog/jupyter-notebook-tutorial/ and also here https://jupyter-notebook.readthedocs.io/en/stable/notebook.html).

- Installation:

  ```
  pip3 install jupyter
  ```

- How to launch:

  open a terminal and type

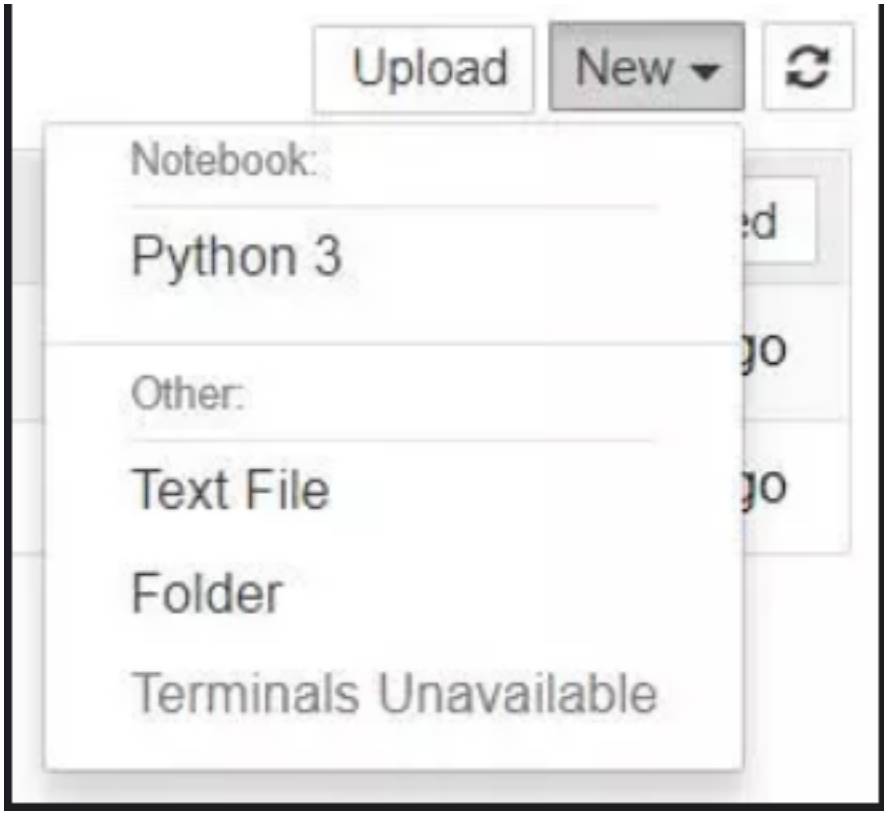
  ```
  jupyter notebook
  ```

  (if you are using linux subsystem use: jupyter notebook –no-browser)

  and a browser window opens (if not, follow the instructions on the terminal and copy and paste the url in your browser).

  With Jupyter Notebook open in your browser, you may have noticed that the URL for the dashboard is something like https://localhost:8888/tree. Localhost is not a website, but indicates that the content is being served from your local machine: your own computer.

  Next you can start a new project (choose Python3 at top right corner among other choices).



- Core ingredients: the two fundamental ingredients of jupyter notebook are *kernels* and *cells*
  - A *kernel* is a "computational engine" that executes the code contained in a notebook document (it depends on the chosen programming language).
  - A *cell* is a container for text to be displayed in the notebook or code to be executed by the notebook's kernel. Cells in turn can be of two types, **code** and **markdown** cells. Cells can be executed with Shift+Enter:
    * **code** cells contain code to be executed in the kernel and display the output just below (the cell is labelled by `In [ ]` at its creation, while after excution change to `In [i]`, where `i` is equal to the order of execution through the kernel).
    * **markdown** cells contain text formatted using Markdown and simply display the output in-place when they are run (see https://www.markdownguide.org/basic-syntax/ for more information).

Below, you'll find a list of some of Jupyter's keyboard shortcuts. You don't need to memorize them all immediately, but this list should give you a good idea of what's possible.

- Toggle between edit and command mode with Esc and Enter, respectively. You can also click on the left part of the cell, i.e. on the `[ ]` symbol.
- Once in command mode:
  - Scroll up and down your cells with your Up and Down keys.
  - Press A or B to insert a new cell above or below the active cell.
  - M will transform the active cell to a Markdown cell.
  - Y will set the active cell to a code cell.
  - D + D (D twice) will delete the active cell.
  - Z will undo cell deletion.
  - Hold Shift and press Up or Down to select multiple cells at once. With multiple cells selected, Shift + M will merge your selection.
    * Ctrl + Shift + -, in edit mode, will split the active cell at the cursor. You can also click and Shift + Click in the margin to the left of your cells to select them

Direct pdf production (for LINUX): you can convert the jupyter notebook into a pdf file by following the below steps:

- be sure that `nbconvert` is installed (otherwise install with pip);
- check if `pandoc` is installed (type `which pandoc` in the terminal, if a path appears then it is installed). If not, install it via `sudo apt install pandoc`;
- to convert from jupyter to pdf format type in the terminal `jupyter nbconvert --to pdf test.ipynb`, where *test* is the file name of the notebook (be sure to be in the same folder that contains the ipynb file);
  - NOTE1: latex equations written within dollars in markdown mode do not admit spaces between the dollar and the equation, i.e. for the expression $\ddot{r} = -\dfrac{GM}{r^2}$ be sure to write `$\ddot{r} = -\dfrac{G M}{r^2}$` and not `$ \ddot{r} = -\dfrac{G M}{r^2} $`;
  - NOTE2: if you wish to add an image in a markdown cell, type `![description](figure.png)`. The figure must be in the same folder of the notebook;
  - NOTE3: pay attention to the width of the text in code cells, if e.g. a comment is too long it could not be seen in the pdf. A quick fix is to avoid too long lines. For more systematic fixes (but less user-friendly) see link below.

* check https://towardsdatascience.com/how-to-convert-jupyter-notebooks-into-pdf-5accaef3758 for additional details and customisation.

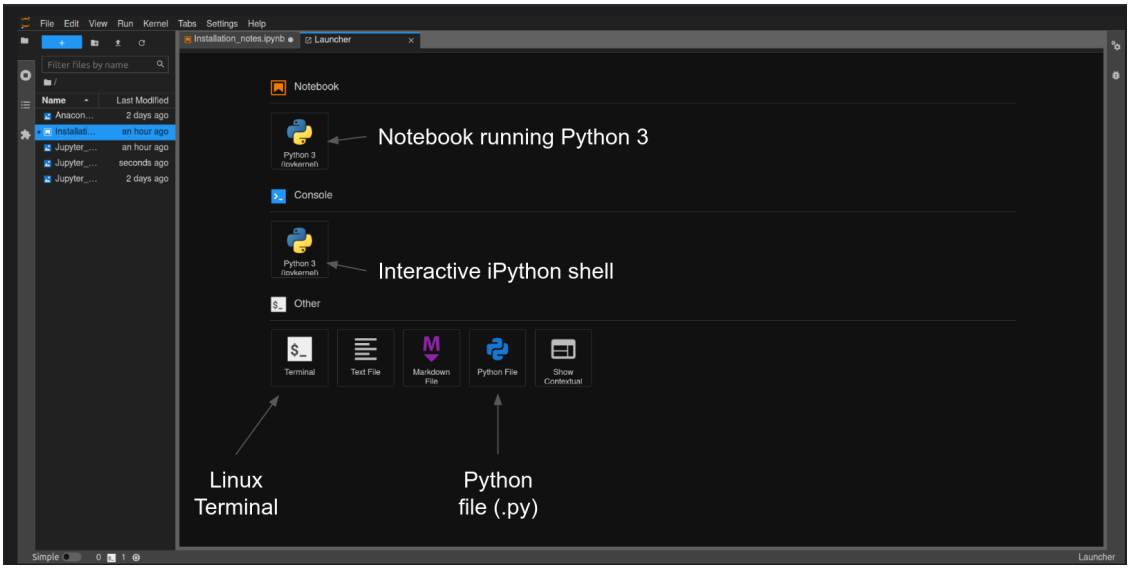
In addition to Jupyter, you may want to install Jupyterlab, which is essentially Jupyter, but more powerful, e.g. you can also open shell within it. As usual you can install it thorugh pip

```
pip3 install jupyterlab
```

and you lauch with

```
jupyter lab
```

Here an example of what you get



As you can see from the jupyter lab launcher you can start: - an interactive notebook (.ipynb) - an ipython interactive shell - a Linux terminal - various kind of files (plain txt, markdown, .py) - an interactive window that grants you access to the documentation of functions invoked in the notebook (we'll see this)

A note on `.py` files: we will usually employ interactive notebooks, but this is definitely not the only way to deal with Python scripting. An alternative way of using Python, that resembles what you do with with older programming languages like *C/C++* or *Fortran*, involves the use of a text editor (there are seveal, jupyter can be used as a simple text editor) and a terminal that executes the code contained in a specific file.

The Python code has to written in a file, e.g. `example.py`, then the code is simply executed in a terminal with the command

```
python3 example.py
```

We can also pass command line arguments like

```
python3 example.py arg1 arg2 arg3
```

that can be used by the code (we will see how this works).

[ ]: