

Lecture 1

Physics Simulations with Python: prerequisites, tools and basic concepts

Laboratorio di Fisica Computazionale
Computational Physics Laboratory

Antimo Marrazzo (Physics Department, UniTS)
AA 2023/24 I semester









What is Python?

- «Python is an easy to learn, powerful programming language» (source: official Python tutorial)
- Few catches:
 - Easy to *start* coding, difficult to loose accents from other languages
 - You only miss what you know about: several powerful features potentially unexploited
 - Often harder (or not obvious) to produce efficient code for numerical simulations, especially at the HPC level. (Fortran is a Formula TRANslator, it was designed for number crunching; Python is more general purpose, from web design to data analysis)

What is Python? (really)

- Python is an **interpreted**, interactive, **object-oriented** programming language. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes.
- **It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming.**
- Python combines remarkable power with **very clear syntax**.
- It has **interfaces** to many system calls and **libraries**, as well as to various window systems, and is extensible in C or C++. It is also usable as an extension language for applications that need a programmable interface. Finally, Python is portable: it runs on many Unix variants including Linux and macOS, and on Windows.

[Source: docs.python.org]

Why Python?

(in a Computational Physics Laboratory)

- It dramatically reduces **the time to develop codes** (especially true if the programmer time is worth more than CPU time)
- **“Python as a glue”**: ease of integrating C, C++ and Fortran code
- Great for prototyping code
- Great for data analysis, machine learning & visualizing data
- It can be made efficient with extensions and libraries
- It has becoming extremely popular also in computational science (existing projects and available libraries)
- NB: *Pythonic* strategies, tools and style are radically different w.r.t compiled codes...*especially if you were originally trained with C or Fortran!*

A disclaimer

- This course is about **computational physics**, ***not*** a **coding class**.
- We recommend to check out (and, in case, attend) one of these two courses
 - >1 semester course **682SM *Abilità informatiche e telematiche (Computer and Telematic Skills)*** by Milena Valentini and Sara Bertocco
 - >1 semester course **998DF *Strumenti Informatici per la Fisica (Information Technology for Physicists)*** by Daniele Coslovichslides and other material available on the Teams channels, access codes available at <https://www.units.it/en/distance-learning>
- We will not teach you how to code in Python from scratch
 - >check out 682SM or 998DF for a primer
 - >we will revise key concepts through short summaries and code examples

A disclaimer

- We will ***not* require** you to know how to code in Python
- The proven capability to develop a code for numerical simulations in modern Fortran AND Python (i.e. using both!) will be evaluated very positively.
- We will show you that implementing physics simulations sometimes requires different strategies in Python than in C or Fortran 90.
- Statistically speaking, last year many students spontaneously decided to implement the code for their final project both in Fortran 90 and Python:
 - Fortran code embedded in Python via f2py
 - Fortran code for number crunching & Python code for data analysis

To tackle the problem a Python code has been written, to simulate MC evolution of such a system on a square lattice. In second place, in order to make viable the simulation of larger lattices and finer annealing schedules, a Fortran module was developed and was integrated in Python using NumPy's F2Py utility, which had already proven to be very effective in speeding up the computational time of similar problems.

Taken from a final project report


Homework #1

- Make sure you are familiar with these topics
 - Basic Python syntax.
 - Basic built-in datastructures (lists, tuples and dictionaries).
 - Control structures (if-else, while, for).
 - How to write and use functions and modules.
 - Basic File I/O (read and write a formatted text file)
- If you come from modern Fortran, check out this Python-Fortran Rosetta Stone <https://www.fortran90.org/src/rosetta.html> (*Python with NumPy and Fortran are actually rather similar in terms of expressiveness and features*)

The Python interpreter

- Python is an interpreted language: the interpreter runs programs by executing one statement at a time.


```
marrazzo@nb-21-174 ~ %
```



IPython

- **IPython is an enhanced Python interpreter** with tab completion, history and other advanced features, including the support for interactive data visualizations and tools for parallel computing.

```
marrazzo@nb-21-174 ~ %
```



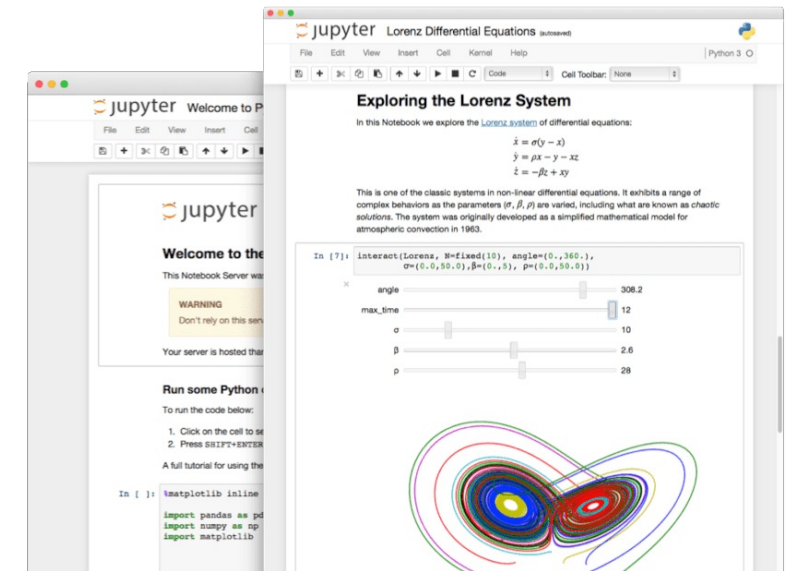
Jupyter Notebooks and Jupyter Lab

- **Jupyter Notebooks**

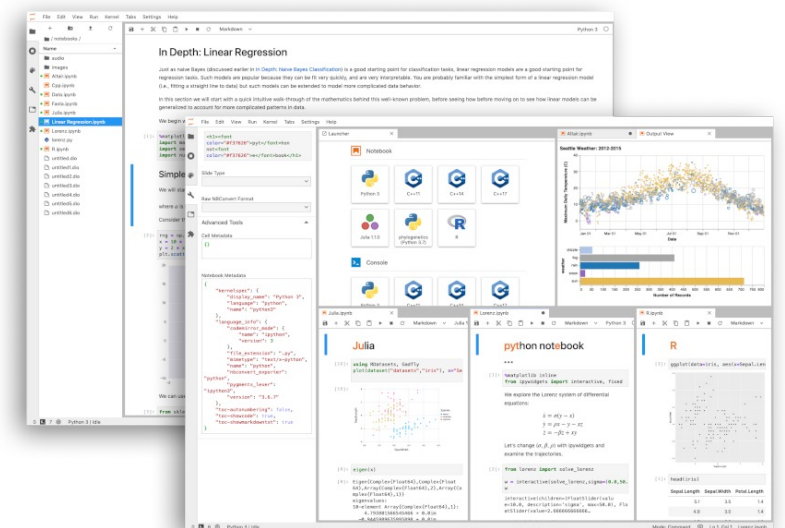
- Spin-off of IPython
- Web-based application for creating & sharing computational documents
- “Web-based” notebooks allow to mix code, text (e.g. Markdown, HTML) and interactive visualization
- They can be used with any programming language (but particularly useful in Python)

- **Jupyter Lab**

- Web-based interactive development environment for notebooks, code, and data



Source: jupyter.org



Integrated Development Environments (a.k.a. IDEs)

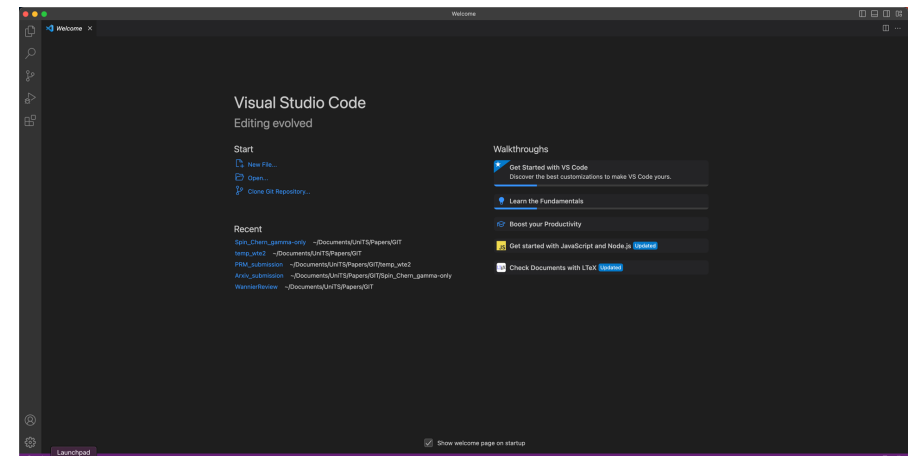
- IDEs are pieces of software which aid computer programmers to write codes
- IDEs are designed to maximise productivity (i.e. save time)
- There exists a large number of them, with a wide range of functionalities:
 - Basic features (code editor): vim, emacs, nano, ...
 - > very useful to use on remote machines (e.g. HPC clusters)
 - More advanced (also GUI, compilers/interpreters, support for version control, ...): Eclipse, Xcode, Visual Studio Code, ...
 - >very powerful to develop code (and write in LaTeX as well!)

```
VIM - Vi IMproved

      version 9.0.270
      by Bram Moolenaar et al.
Vim is open source and freely distributable

      Sponsor Vim development!
type  :help sponsor<Enter>   for information

type  :q<Enter>              to exit
type  :help<Enter> or <F1>   for on-line help
type  :help version9<Enter> for version info
```



Virtual environments

- It is good practice to develop projects in isolated **virtual environments** on top of an existing Python installation, essentially folders which contain all the necessary executables to use the packages that a Python project would need, including their own independent set of Python packages.
- Very easy through the package **virtualenv**
 - pip install virtualenv
 - -> **to create the environment:** virtualenv yourpythonenv
 - -> **to enter the environment:** source yourpythonenv/bin/activate (on Windows: yourpythonenv\Scripts\activate)
 - -> **to exit:** deactivate

...let's use Jupyter
Notebooks!

Some references

- An extensive list at <https://wiki.python.org/moin/PythonBooks>
- For beginners with a Physics background (very recommended!)
Effective Computation in Physics: Field Guide to Research with Python
(A. Scopatz & K. D. Huff, O'Reilly, 2015)
- For advanced Python users:
Fluent Python: Clear, Concise, Effective Programming (L. Ramalho, O'Reilly, 2015)

But ... nobody learns coding on books!

- 1) Learn by doing: practice, practice and practice
- 2) Python official documentation: <https://docs.python.org/3/>
- 3) Stackoverflow: <https://stackoverflow.com>

Installing a Python development environment (useful resources)

**The Ubuntu VMs in *Aula Poropat* are already configured for the course,
you can use those (even from remote!)**

At home

- Option 0 (Linux): for Ubuntu
 - `sudo apt-get install python3 ipython3 python3-pip python3-numpy python3-numexpr python3-matplotlib cython3 python3-cffi python3-scipy`
 - `pip3 install jupyter numba virtualenv`
- Option 1 (Windows, Linux, MacOS): <https://www.python.org/downloads/>
- Option 2 (Windows, Linux, MacOS) [**NOT RECOMMENDED!**]: Anaconda installer (most packages you need for this course are pre-installed)
- If you use Windows, consider also to install an Ubuntu VM with VirtualBox or use Windows Subsystem for Linux (WSL)