Programming for AMEP
Part 2 – Python

Clélia de Mulatier

November - December 2021 (7 weeks)

Goal of the course. This course is very short with only 1.5EC for the Python part of the course. The overall goal of the course is to get you sufficiently familiar with python, as well as, basic programming concepts in computational physics and data analysis, for you to become independent self-trained learners in python for computational physics.

Contacts. The course is taught by:

– Clélia de Mulatier (lecturer):

– Sergio S. Fernández (teaching assistant):

Don't hesitate to contact us if you have any questions about the course, preferably through the messaging system of Canvas.

Exam: Pass or fail mark based on submitted and reviewed assignments. Assignments will be given in Jupyter notebooks.

Plan of the course (for the python part):

Py 1: Getting familiar with Python
Py 2: Bases of programming in Python
Py 3: NumPy arrays for efficient numerical computing in Python
Py 4: Data visualization with MatplotLib
Py 5: Numerical methods for solving ordinary differential equations
Py 6: Introduction to stochastic simulation
Py 7: An introduction to data manipulation with Pandas

Books. The course is based on the following books:

– **Lectures Py1 and Py2:** *A Whirlwind Tour of Python*, by Jake VanderPlas (O'Reilly). Copyright 2016 O'Reilly Media, Inc., 978-1-491-96465-1 – freely available in its web version.

– **Lectures Py3 and Py4:** *Python Data Science Handbook*, by Jake VanderPlas (O'Reilly) – Book available for free in its web version.

– **Lecture Py6:** (last section) *Statistical Mechanics: Algorithms and Computations*, by Werner Krauth.

Exercises are also inspired from the course (in French) *"Programming in C for physicists"* by François Naulin, and from the book *A student's guide to python for physical modeling* by Jesse M. Kinder and Philip Nelson.

Questions. Don't hesitate to contact us if you have any questions about the course, preferably through the messaging system of Canvas. If the answer to your question can also be useful for others, we encourage you to post it on Canvas in the *Discussions* page of the course. Check Canvas frequently (discussion and other sections of the course), and feel free to help each other by replying to questions for which you know the answer.

Feedback questionnaire for the course of Nov-Dec 2021: Here.

Note about the Course Design and organisation in 2021/22

The course is designed to adapt to all possible students' levels and background in Python. In November 2021, students who considered themselves sufficiently autonomous in Python were offered the possibility to do a computational project instead of following the course. Students were ask to do the first tutorial before they decide if they would like to take the project; none of the students took the project.

To re-fresh students memory in Python, programming tools and notions of algorithmic are progressively introduced, re-starting from a beginner's level in Python. Nevertheless, due to the high speed of the course, we expect students to already have some notions of programming in Python or in another language before taking this course. In each module, in parallel of learning (re-learning) Python tools, students are progressively introduced to new numerical methods. To help students practice and build a long term memory of the concepts and tools they see during the course, exercises build up on the notions seen in all the previous modules.

Course organisation: There are 2h of in-person lecture per week. The course is organised the following way. Each week:

- the course start with 30 min to maximum 40 min of lecture;
- the course is then followed by a tutorial session until the end of the 2h: the students work through the questions of the Jupyter notebook associated with the lecture. The lecturer and the TA go around the classroom to answer students' questions and help them with the exercises.
- each Jupyter notebook is divided in two parts:
  - for modules `Py1` to `Py4`, the first part contains simple questions that helps students understanding how to use the tools seen in the lecture; these questions appear as **Q.xx** in the notebooks.
  - the second part contains more advanced exercises in which students can put in practice the concepts, methods and tools they have seen so far; these exercises appear as **Exo.xx** in the notebooks.
- assignments: each week, students must submit their notebook on canvas, the deadline being at the start of the next lecture;
- each week, students are provided with feedback on their previous assignment by the lecturer and TA directly through their Jupyter notebook. Student must then re-submit their corrected assignment on canvas. Students can ask questions about their assignment directly to the lecturer and TA during the lectures/tutorials.

Lecture 1 (2h) — Getting familiar with Python

Book. *A Whirlwind Tour of Python*, by Jake VanderPlas (O'Reilly). Chapters 1 to 5, 10 and 14.

Goal of the lecture. The main goal of this lecture is for students to have installed Python3 and Jupyter notebook on their laptop, and to know what are the main tools available for writing and running Python codes. At the end of this course, students should be able to write and run simple Python3 codes on their laptop. Students should also have some basic understanding of Python variables, and of number types (int, float) and the numerical limitations that comes with them.

More specifically, after this lecture, all students should:

- have **Python3 and Jupyter notebook installed** on their laptop;
- be familiar with using the **python interpreter**, using **python scripts**, and using **Jupyter notebooks**, and be able to write simple programs in each of these environments; Export scripts from Jupyter notebook.
- know about variables in python and about simple **data types**, and how to use them appropriately to perform operations;
- be aware of the **numerical limitations** of number types (limits and precision);
- start **checking error messages** and **using the `help` function**, as well as the `tab` auto-completion tool in Jupyter notebook; start using relevant online documentation to answer their questions;
- know how install and import **third-party python modules**; have installed `numpy`; import `math` or `numpy` to use common mathematical functions;
- be familiar with **good programming habits**.

Lecture plan.

1. Setting up Python3 (installing Python3, Pip, Other package managers)
2. Working with Python (Python Interpreter, Python Scripts, and Jupyter Notebook)
3. Python variables and objects (Variables, Objects, Attributes and Methods, Simple Built-in Types)
4. Python language syntax
5. Operations (Operations, Illegal operations)
6. Python modules, and help features
7. Good programming habits

Evaluated in the exercises: *using python interpreter, python script and Jupyter notebook; using variables; identity operation; numerical operations and priority rules; types (int, float, complex, bool, string); float limits and precision; scientific notation in python; keeping track of the physical units when solving a computational physics problem; using modules math/cmath/numpy; errors; boolean logic and conditions.*

Lecture 2 (2h) — Bases of programming in Python

Book. *A Whirlwind Tour of Python*, by Jake VanderPlas (O'Reilly). Chapters 6 to 9, 11 and 12.

Goal of the lecture. This lecture introduces students to Python's built-in data structures. This course dives into python's "way of thinking and programming", in particular with the use of loops over iterable objects. The main goal for students is to have understood how such loops work and to practice how to use them through the assignments.

In more details, after this lecture, all students should be able to:

- define any python list and tuple; use lists and tuples appropriately.
- use indexing to access or modify an element of a list; use slicing to extract a part of a list;
- write *for* loops over an iterable python objects; to use *while* loops when appropriate;
- define lists using list comprehensions;
- define and call their own functions;
- compute in very simple terms the complexity of their algorithm.

Lecture plan.

1. Python built-in data structures (Lists, Tuples, Dictionaries, Sets, notion of mutable and immutable objects)
2. Conditional statements (recall from last week)
3. Loops (*For* loop – over any iterable object, *While* loop, List comprehension)
4. Functions

Note: The notion of computational complexity is an important notion, which will be discussed in the exercises. This notion will come back in the next lecture, when comparing Python loops to Numpy vectorized operations.

Evaluated in the exercises: *defining basic lists, tuples, dictionaries, and sets; using* `range` *objects to define lists; writing for loops over iterable objects; writing while loops when appropriate; choosing appropriate stopping criteria for while loops; using list comprehensions; defining and using functions.*

Additional notions seen in the exercises:

- Computing the algorithmic complexity of simple algorithms;
- Simple common programs using recursive definitions: computing $n!$, computing $\binom{n}{k}$;
- Numerical functions computed as the limit of a series (ex. $\sqrt{x}$);
- Sampling, and linear interpolation;
- Numerical differentiation and numerical integration

Comment: In this assignment, students are programming simple python functions, for which they will see the vectorized version or a python function that does it in a future lecture of the course.

Lecture 3 (2h) — NumPy arrays for efficient numerical computing in Python

Book. *Python Data Science Handbook*, by Jake VanderPlas (O'Reilly). Chapter 2.

Goal of the lecture. This third course focuses on Numpy's Arrays and Universal functions. The main goal of the lecture is for students to have a general understanding of why python loops are slower than vectorized operations in NumPy, and to remember that using vectorized operations with NumPy are generally much faster than using python loops. The main goal of the assignment is to make them practice Numpy arrays and vectorized operations.

More specifically, after this lecture, all students should be able to:

- define any Python Arrays (one or multi-dimensional), from a list, from scratch using `arange()`, or from other numpy functions.
- perform array slicing and extract any column or line from a multi-dimensional array;
- be aware that slicing only provides a view on a sub-array, and know how to copy array when needed;
- have a general understanding of why Python loops are very slow compared to numpy's vectorized operations;
- perform operations on NumPy Arrays, including linear algebra operations;
- compute simple statistics on NumPy Arrays using methods;
- sampling random Arrays using functions from the `random` library of NumPy.

Lecture plan.

1. NumPy Arrays
2. Operations on NumPy arrays
3. Simple Data Statistics on arrays
4. Linear Algebra with `np.linalg`
5. Random Arrays with `np.random`

Evaluated in the exercises: *defining Numpy arrays using list, `arange()`, array functions, and stack functions; performing any array slicing and extract columns or lines from multi-dimensional arrays; using Ufunc and vectorized operations; simple data statistics with numpy arrays.*

Additional notions seen in the exercises:

- Computing the norm of a vector using vectorized operations;
- Computing the divergence and the rotational of a vector field

Comment: The next assignment (Lab 4 on MatplotLib) contains two advanced exercises training students on python vectorized operations: one solving the Laplace equation, and one with linear algebra (propagation of a random walker on a graph).

Lecture 4 (2h) — Data visualization with MatplotLib

Book. *Python Data Science Handbook*, by Jake VanderPlas (O'Reilly). Chapter 4.

Goal of the lecture. This lecture gives an overview of the possibilities offered by Matplotlib for plotting data in python: from simple 2D plots, histograms and density functions, to contour plots and 3D surfaces. The goal is for students to get familiar with Matplotlib to plot data in Python, and to become autonomous in searching for information online on how to plot specific graphs and customize their plots.

Lecture plan.

1. General information (Importing Matplotlib, Using Matplotlib in Jupyter notebooks, Plot styles, Simple plots, Save a figure, Exporting figures to file)
2. Customize your plots (Line styles, Adjusting Axes Limits, Labeling plots)
3. Visualizing Errors
4. Bar plots, Histograms, and Density functions
5. Density and Contour Plots
6. 3D-plots (3D scatter plots and line plots, 3D Contour and surface plots)

Evaluated in the exercises: *More advanced computational physics exercises, using numpy arrays for computation and MatplotLib to visualize data:*

- Solving Laplace's equation in a 2-dimensional system (+contour plot and 3d-plot);
- Random walk on graphs: evolution of the density function and entropy of the walker.

Comment: The exercises on solving the Laplace equation introduces notions of numerical integration of differential equations that will be used in the next lecture. The results from the exercises on the random walk can be compared with a stochastic simulation of the walk in Lecture 6.

Lecture 5 (2h) — Numerical methods for solving ordinary differential equations

Goal of the lecture. Getting familiar with fundamental/basic numerical methods for solving differential equations. After this lecture, students should know how to discretise and solve ordinary differential equations (ODEs) and systems of coupled ODEs using the Euler and RK4 methods, as well as more advanced numerical tools implemented in SciPy.

Lecture plan.

1. First order Ordinary Differential Equations (ODEs - Euler method and RK4 method)
2. Systems of coupled first order ODEs
3. Second order ODEs
4. Solving ODEs with SciPy

Evaluated in the exercises: *implementing the Euler method and the RK4 method to solve first order ODE. How to reuse this functions to solve coupled ODEs and second order ODEs. How to solves these ODEs using SciPy functions. Exercises:*

- Second order ODE: driven harmonic oscillator;
- Coupled ODEs: Modeling neuronal activity with the FitzHugh-Nagumo model

Lecture 6 (2h) — Introduction to stochastic simulation

Book. *Statistical Mechanics: Algorithms and Computations*, by Werner Krauth. Chapter 1.

Goal of the lecture. After this course, students must be familiar with how random numbers are generated in by computer; know how to sample from a discrete probability distribution; know how to sample from a chosen probability distribution using Numpy; have a general understanding of what is a Monte Carlo Simulation.

Lecture plan.

1. Random number generators

1. Sampling random variables (Discrete random variables, Continuous random variables, Sampling random numbers with Numpy)

2. Example of application: Brownian motion in continuous space

3. Introduction to Monte Carlo simulations (calculating the area of a circle)

Evaluated in the exercises: *Sampling random number from discrete and continuous probability distribution; Implementing simple stochastic simulations; Implementing simple Monte Carlo simulations.*

- Simulation of a Brownian motion in continuous space; recover Central Limit Theorem;
- Calculating the area of a circle with Monte Carlo;
- (optional) Calculating integrals with Monte Carlo;
- (optional) Simulation of the random walk on a graph of Lecture 4: recovering the probability distribution obtained in lecture 4.

Lecture 7 (2h) — An introduction to data manipulation with Pandas

Book. *Python Data Science Handbook*, by Jake VanderPlas (O'Reilly). Chapter 3.