Python for (open) Neuroscience. Final report

The course **Python for (Open) Neuroscience** took place in the second semester of the academic year 2022/2023. It was delivered in 18 frontal lectures (two two-hours lectures per week), organised in 4 different modules.

Given the experimental nature of the course, and the challenges of teaching coding to a very diverse audience, I have been seeking constant feedback from the students to optimally titrate the content and the structure of the lectures. Here I share some observations that could be useful in future iterations of the same or different coding classes, based on the interim and final student feedbacks.

Summary points

- There seems to be a high demand for a Python course in a neuroscience graduate school;
- The length of the course (32+4h) has been appreciated and even deemed insufficient by some students;
- Frontal lectures with in-class exercises were effective;
- It is important to teach it in a way that can be useful for students; this means:
  - not stopping at the basics
  - start with plug-and-play tools (like Colab) but bring students to the level of using local Python installations by the end of the course;
- Is it beneficial to include final projects related to research, submitted via GitHub

---

Timeliness and need

The course had a very good participation (20 sign-ups; 3 dropped out of the course (some would like to do it next year); 14 students completed assignments for credits; occasional joining by 2 postdocs & 1 undergraduate student). Several students were already running Python code for their analysis, and appreciated the opportunity of understanding the language better; several others learned to implement code they could immediately apply to their research work (for experiment scripting or data analysis).

As the neuroscience/cognitive neuroscience field is increasingly adopting Python-based tools, the overall feeling shared by the participants is that offering a Python course is fundamental in a modern graduate school.

A lateral aim of the course was to teach some basis on version control and code sharing using GitHub. Those topics also struck a chord, as many students were happy to learn about this aspect of coding that is often disregarded when learning a specific programming language.

Overall organization

The course aim was ambitious: it was structured around the idea that to learn to code, a student needs to immediately apply what they learn in their day-to-day research work. Therefore, it started from the bases but it was set to get to the point at which students would have been capable of using Python in the real world.

I feel this was a good approach, as several students already had brief introductions to programming that never stuck with them being too introductory and general. I would strongly recommend keeping this in mind when planning future programming courses.

The cost for this was a **long time and strong commitment on the students' side**. The challenge was **very positively taken by the students**: the initially scheduled 24 hours got extended with their unanimous approval to 32 hours, and even with this extension many students suggested that the course would have benefit from a longer schedule. Some of the students who dropped the course complained about it being too fast; more time could have probably reduced the drop out rate. The

course started late and it was not possible to reduce the pace.

For the future, I would recommend to space out the lectures of the first module, to make sure everyone could have the time to assimilate all parts of the lectures.

Structure of the lectures

The course was structured around **Jupyter Notebooks/Slides** that were shared with the students before the course. The frontal lecture was carried out with material that the students could run on their machine following along and experimenting during the lecture. During the lecture, **20-30 minutes blocks of explanations were alternated to 20-30 minutes blocks of hands-on exercises**. The exercises ranged from drills on the syntax and basic functions to more elaborate tasks for the most skilled students. This formulation was tuned with inputs from the students and got quite satisfactory by the end.

Some **office hours (5 x 2h sessions) were scheduled for the students to come and ask questions** on the lectures and the assignments. All office hours had at least 4 students attending and where overall productive: students where less afraid of ask questions in this more private setting. A **teaching assistant** (Jean-Charles Mariani, IIT) was helping out for the good part of the course, assisting students during the practical part of the lecture (some students suggested that a second TA would have been beneficial with this number of participants).

A request by the students has been more verbose teaching slides; another common feedback was the request for more verbose and clear instructions on the exercises. Both could be addressed and improved with more iterations of the course. Some students reported some challenges in following the explanations, that sometimes were assuming too much knowledge; a student reported being a bit afraid of asking questions, even if the atmosphere was nice. A lot of effort by the teacher should be put in making sure everything is delivered as slowly and clearly as possible.

Lectures content

The syllabus of the course was followed quite thoroughly; with the supplementary hours, no topic had to be dropped.

- The **first module (Colab notebooks-based) went through all the basics** of Python: variable types, flow control, functions, classes and objects. The content of the module was generally well assimilated. Some students suggested to avoid introducing concepts related to bits/bytes/memory too early, and focus more on usage, which I think is a good idea. Several students found hard the objects/classes class, with some suggesting to skip or keep for later in the course. This I do not agree with; however, more time should be allocated to assimilate this important part of the language, and more effort put in finding the optimal way to deliver it.
- **The second module (Colab notebooks-based) focused on `numpy`, `pandas`, and `matplotlib`**. This was considered in general the most useful module, and I would not change anything about how much time it was allocated for it.
- **The third module started with an intro to Python in the real world**: students had instructions to configure a local Python installation, and the first lecture was dedicated to understanding a bit how it worked. We talked about environments and pip installing third-part modules. This lecture was very appreciated and I feel it is crucial in a Python course that want to make students self-sufficient. The second and third lectures were more focused on working with local files, how notebook work, and image analysis and statistics in Python. The image analysis was overall not considered very useful and could be dropped or made optional in the future, could be swapped with one of the final lectures.
- **The final module was covering topics very close to every-day work***, and had 5 lectures of which students could choose 3: GitHub/version control, experiment scripting/hardware control, functional imaging analysis, advanced statistics, data visualization. The lectures that were most appreciated were GitHub version control (which I would recommend keeping part of main course in the future) and data visualization. Some students noted that the approach of single lectures for

specific topics might not be the best structure for absorbing the content. Maybe it would be better to have the students picking two themes and focus on them.

By the end of the course, the students have asked for: overall **more time on the fundamentals**, by extending the duration of the course; **more `pandas` and data visualization**; more systematic lectures on experiment scripting (`psychopy` etc) and imaging analysis; publishing code and developing custom packages; however, those last three would probably take too much time with topics that are not of general interest.

Evaluation

The first two blocks of lectures (Basic Python and Scientific Python) were followed by a set of Assignments that the students had to complete on their own in order to get their final pass.

In addition to the assignments, students had to develop a Python project. Most students opted to do it on work-related coding issues they were addressing, proving that the course could bring some actual value to the research they do in the lab. Projects had to be packaged in a GitHub repository, with an explicative `README` containing the instructions to run it and replicate results. In this way, it was not only a Python programming exercise, but also a way for students to practice sharing code related to their research - something that will be increasingly important in the future.