Collaborative Scientific Programming: an evening pilot course at GFI-UiB

Despite having naturally become a cornerstone of scientific studies and careers, programming skills remain neglected in the teaching environment at most departments of the University of Bergen. At the Geophysical Institute, almost all courses list programming skills as one of the prerequisites for their students, yet very little training is offered to the students taking their degrees. As a results, students are left to rely on their own autonomy to start with the learning, struggle to see the uses and benefits of programming, and seldom experience consistent repetition and continuity with programming throughout their studies.
This background inspired us to together create a course targeting the needs of the students; instead of preparing another standard programming course, we designed it around certain particularities:

- the course is about Python, an open-source, polyvalent and state-of-the-art programming language in the natural sciences,
- the programming is tailored to the needs of the students at the Geophysical Institute, namely the processing, visualising, and analysing of data from ocean/atmosphere datasets,
- the course project mimics the workflow of contemporary scientific projects, with individual tasks, shared responsibilities and communication all along,
- the communication inside the groups and across different groups is verifying the latest academic and industrial standards: a Git-based framework to retrieve and share data, and collaborate throughout the coding processes.

Our course therefore focused on collaborative scientific programming and was given across a period of 2 weeks during the Spring 2024 semester. 15 students from the 2nd to the 4th year at the Geophysical Institute joined for 5 evenings (17:00-20:00), with a break for collective dinner in the middle of each session.
We iteratively designed a course-program that would focuses on each element of the curriculum separately, before combining them in a group-project. We laid out a structure-skeleton, discussed what modules were essential, and in what order to best present them. By doing this weeks ahead of the course start, we had enough time to talk through each module, add activities, and revise the order and contents, if needed. During the course, and based on the progress of each evening, we updated the content for the next evening, either repeating previous elements that were deemed complex, or swapping modules to increase the coherence of the teaching flow. Every evening, we would summarising the content of the previous evening in order to help with the information retention. Our final curriculum became as follows:

- day 1: introduction (theory on Git for programming, theory of teaching and learning, installation and setup of the necessary softwares and libraries)
- day 2: scientific python (presentation of coding work by invited researcher, practice with tasks and tutorials, making sure everyone reaches the same minimum level of skill and knowledge)
- day 3/4: collaboration (basic Git commands, setup of Gitlab accounts, scientific group project)
- day 5: conclusion (compilation and presentation of project-results to an invited climate science expert, scientific discussion around plots and analysis)

Our teaching was based on literature from didactics and discipline-based educational research, to maximum the learning and overall stimulation for the students. From a discussion on the theory of meaningful learning, we highlighted the importance of active participation, collaboration amongst peers, and the use of resources and tutorials. Doing so, we encouraged the students to develop as self-determined learners in order to gain efficiency and seek out knowledge on their own. In accompaniment of the course, we also measured the evolution of certain constructs for the students through a survey completed before and after the course, and obtained positive results regarding their motivation, self-efficacy and sense-of-belonging to the discipline. These results and their analysis will be presented in a future academic publication that we are now preparing.
This course on collaborative scientific programming was very successful and resulted in great satisfaction for the students taking it, as well as for us leading it. The students learned about the importance of programming in ocean, atmosphere and climate sciences, the advantages and hurdles of using Git-tools in scientific projects, insights about meaningful learning, the common packages used for scientific routines, the basic Git commands, and how to work on a collaborative scientific project. On our side, we particularly enjoyed preparing and curating this course together, taking advantage of our different fields (oceanography and educational research) and our perspectives from two different academic levels (master student and junior researcher).

Daniel Oddmund Lid & Julien-Pooya Weihs