Presentation

Improving students' programming skills through Collaborative Scientific Python

Julien-Pooya Weihs[1,2] og Daniel Oddmund Lid[1]

[1] Geophysical Institute, University of Bergen, Norway
[2] Bjerknes Centre for Climate Research, Bergen, Norway

Abstract:
Scientific programming is an increasingly important part of the scientific curriculum and careers overall. However, students often struggle to see its uses and benefits, and where and how to begin learning it. At the same time, teachers face challenges in defining clear programming goals and creating new teaching materials that integrate the complexities of scientific programming.
To address these issues, a pilot course of five lectures over two weeks was created at the Geophysical Institute of the University of Bergen in Spring 2024. This course aimed to help students understand and practice the basics of scientific collaborative Python, focusing on common Python packages, standard scientific procedures, basic Git commands, and collaborative tools. Designed with methods supported by didactic literature, the course emphasized active participation and group work, using real-world geophysical data. Pre- and post-course data were collected via surveys from 12 of the attending students.
We here present the main structure of the course, and the resources developed for it. We also share the reported positive impact on students' motivation, self-efficacy, and sense of belonging. Additionally, we highlight the students' reflections on the integrated Python-Git framework at the centre of the course.

Keywords:
Programming education, Python, Git, Collaborative learning, Co-creation

1 Introduction

1.1 Context

1.1.1 Programming teaching and learning

Programming education poses challenges to both students and teaching staff. Discussions with students have revealed that they can lack experience with the programming routines required by their disciplinary courses, or lack the motivation to start learning programming, are not sure where to begin, and do not see the benefits of it overall. The too few assessments of programming tasks and sometimes absence of programming during the exams render the motivation to learn low. Lecturers in the natural sciences reported that they often use a programming language in their research that does not coincidence with the programming language in their teaching, leading to a possible mismatch between research and instruction, and inducing teaching difficulties. In addition, programming is a complex topic to teach, with specific semantics, syntax and use cases, and programming goals are not always well defined in the curriculum. This results in teaching and learning programming in higher education to be one of the cruxes of a modern degree in natural sciences, despite programming being a skill of very high societal relevance and necessity in scientific careers.

A sustainable solution is hence needed, amongst others to introduce programming to the students in an incremental way, to help students with their related struggles, and to provide students with the associated professional skills (coding + data management).

1.1.2 The Git-VSC framework

Git (https://git-scm.com) is a cloud-based version control system that allows to track any history of changes to a file or project. Recording edits and updates, it also permits to revert a file or project to an earlier saved version of it. This makes it a transparent system for users and developers, showing throughout the development of a project which changes have been made to what by who, when and even why, as contributors can comment on each of their submitted changes. Every contributor to a project has a local copy of both the project and project history, and the changes to a group project are tracked between the local copies and the collaborative cloud-based version of the project.

A popular online platform for working on project development using Git is Github (https://github.com), and it allows multiple people to work simultaneously by tracking and organising their changes to all files of a same project. It is the most popular worldwide platform for developers (100 millions developers and over 386 millions repositories in 3/23), and is commonly used for open-source projects (40 millions publically available repositories in 3/23). In the geosciences, a search across public repositories gave 1700 hits for ‘climate models’, 1000 hits for ‘oceanography’ and 2600 hits for ‘meteorology’ on the Github platform in 3/23.

Visual Studio Code (https://code.visualstudio.com) is a free and open-source code editor/interpreter/compiler with a wide range of features developed and maintained by a large community. It has been the most popular coding environment for users (74% of

71000 respondents in 2022) and allows a full integration and synchronisation of projects with Github.

By introducing and using a combined Git-VSC environment to the students in our course, we provided them with a tool that is valued as a professional skill to master, that offers great collaborative possibilities, and supports integrating programming into any possible course. The use of Git also allows for full transparency on individual contributions in the workflow, and assures that shared documents are always up to date for everyone.

Research on using Git in programming (Zagalsky et al., 2015; Kertész, 2015; Feliciano et al., 2016; Glazunova, 2021) has shown that it improves collaboration, allows for effective version control, facilitates a platform for discussion between users, holds users accountable for their contributions, and allows students to develop technical skills at an industry-standard. However, it is also acknowledged that there is a steep learning curve for beginners unfamiliar with these environments, that the framework may not be suitable for grading assignments, and that some students may be uncomfortable sharing their work.

1.2 Meaningful learning

Weurlander et al. (2009) present a theory of learning built upon 5 main pillars: context, motivation, connections, wholeness, and collaboration. Figure 1 presents a graphical representation of these intertwining constructs.

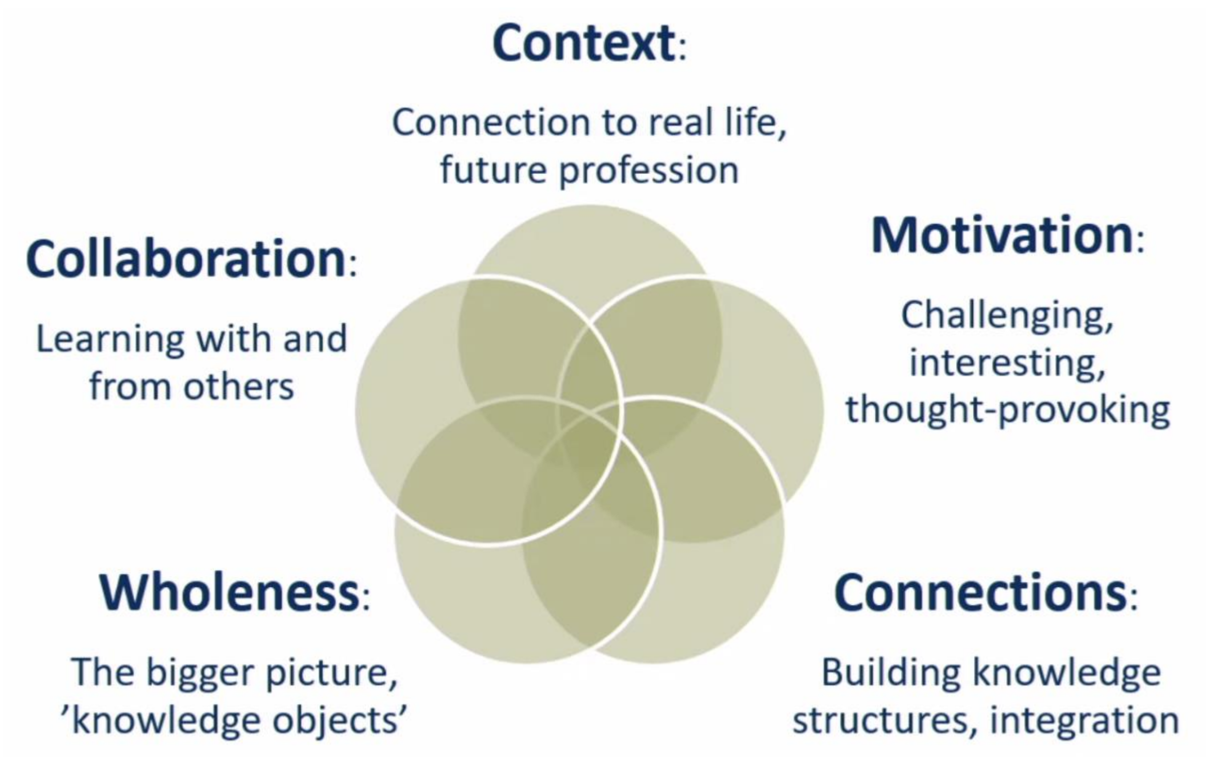
Context:
Connection to real life, future profession
Motivation:
Challenging, interesting, thought-provoking
Connections:
Building knowledge structures, integration
Wholeness:
The bigger picture, 'knowledge objects'
Collaboration:
Learning with and from others

Figure 1: Representation of the 5 main constructs underlying the theory of meaningful learning.

We value this theory for its multifaceted structure that overlaps with the main constructs from other popular theories of learning: collaboration integrates the socio-constructivist view that learning is the negotiation of norms and knowledge amongst peers, and can hence influence our sense of belonging to a group. Context and connections are closely tied with self-efficacy, when a learner realises the relevance and applicability of their knowledge and skills to their discipline. Motivation is commonly split into 2 mains axes: extrinsic, relating to external drivers, and intrinsic, relating to genuine interest. We used this framework when developing our course, being mindful to address each of these 5 pillars in the learning outcomes for the students.

Particularly, we put emphasis on students *learning to learn* (Blaschke, 2012), where the final goal is not the content but their development as a learner: capable of finding new approaches and information, and applying these in an effective manner. This can include utilising online resources, actively communicating with peers, and developing creative skills within the workflow. The learners undergoing such a process would become self-determined, gaining abilities to be effective and seeking out knowledge.

We lastly followed recommendations on teaching from discipline-based educational research (Singer et al., 2012; Deslauriers et al., 2019), namely, to encourage students' active participation, to involve students in collaborative activities, and to supplement instruction with tutorials.

1.3 Research Question

Considering the challenges of programming education, and designing a research-informed course on collaborative scientific programming, this project was driven by the following research question:

What are the students' perception of the course on their programming learning?

2 Methods

2.1 The Collaborative Scientific Python Course

2.1.1 Course design

This first edition of our course was given as a pilot at the Geophysical Institute of the University of Bergen. The course was co-created by the authors, respectively a PhD student in geoscience education, and a master student in oceanography, taking advantage of two different disciplinary perspectives, with the following objectives:

1. allowing the students to reach a certain fluency in working on collaborative projects using Git: download files, *fork/branch* projects, submit *pull/merge* requests, *commit-push* tracked changes, *fetch-pull* new versions, and run and manage projects with peers and collaborators.

2. providing the students with an understanding and practice of basic scientific Python: discover and use *Pandas*, *Matplotlib*, *Numpy*, *Cartopy*, *Scipy*, be able to read-in, pre-process, analyse and visualise geophysical data, and groom and utilise these skills all along their studies and future careers.

To this end, the course featured activities using real-life data and focusing on scientific case-studies from meteorology and oceanography.

We iteratively designed a course-program that would focus 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 day, we updated the content for the next day, either repeating previous elements that were deemed complex, or swapping modules to increase the coherence of the teaching flow. Every day, we would summarise the content of the previous day to help with the information retention.

The course was open to all students from all programs at the Geophysical Institute, and listed 15 attendees (course capped at 16 students, with 1 last-minute cancellation). The instruction was provided over 5 days during 2 weeks in the Spring 2024 semester, from 17:00 to 20:00, and offered free pizza and snacks to the students. No exam was used to assess the learning of this course, but the students prepared a final group presentation given to an invited climate expert from the Geophysical Institute.

2.1.2 Teaching structure

The curriculum was as follows:

Day 1 – **Introduction** (+ pre-course survey)
- theory and motivation behind meaningful learning & Git for programming.
- installation of Python, Visual Studio Code and Git infrastructure.

Day 2 – **Scientific Python**
- demonstration of use cases by invited researcher (in oceanography).
- practice and exercises.
- getting everyone past the minimal required programming level.

Days 3 & 4 – **Git & Collaboration**
- basic Git commands.
- scientific group project.
- collaborate in teams and between different teams.

Day 5 – **Reflections** (+ post-course survey)
- present and discuss results with a climate expert.

2.2 Student surveys

In order to capture the self-reported impact of the course on the students' learning, we created pre- and post-course surveys inspired by the works of Boljat et al. (2019) and Van Staveren (2022), and answerable on a 5-points Likert-scale. Both surveys had in common a set of 10 questions targeting intrinsic, extrinsic and general motivation, self-efficacy, and belonging, and the post-course survey had an additional set of 10 questions to reflect on the course directly. An open-ended final question allowed the students to provide additional thoughts and comments for improvements of future editions of the course. Participation to the surveys was done on a voluntary basis, and 12 of the total 15 students responded to both the pre- and post-course surveys.

3 Results and Discussion

3.1 Course resources

An important result of this project was the creation of various teaching and educational resources, amongst which:

- Course slides for 5 lecture days.
- Python exercises and possible solutions.
- Pre- and post-course survey instruments.
- Link to GitHub repository with all resources.

All these resources are available upon request to the corresponding author. We would be glad to inspire fellow teaching practitioners to use and build up on these resources, both to improve the teaching material, but also to collect more educational data through the surveys, and hence contribute to a larger body of research and instruction in programming education.

3.2 Students' responses

At the time of writing, the students survey data is being processed and analysed, and we here present only an excerpt of our preliminary results. Figure 2 displays the evolution of the pre- and post-course survey responses (first set of 10 questions).

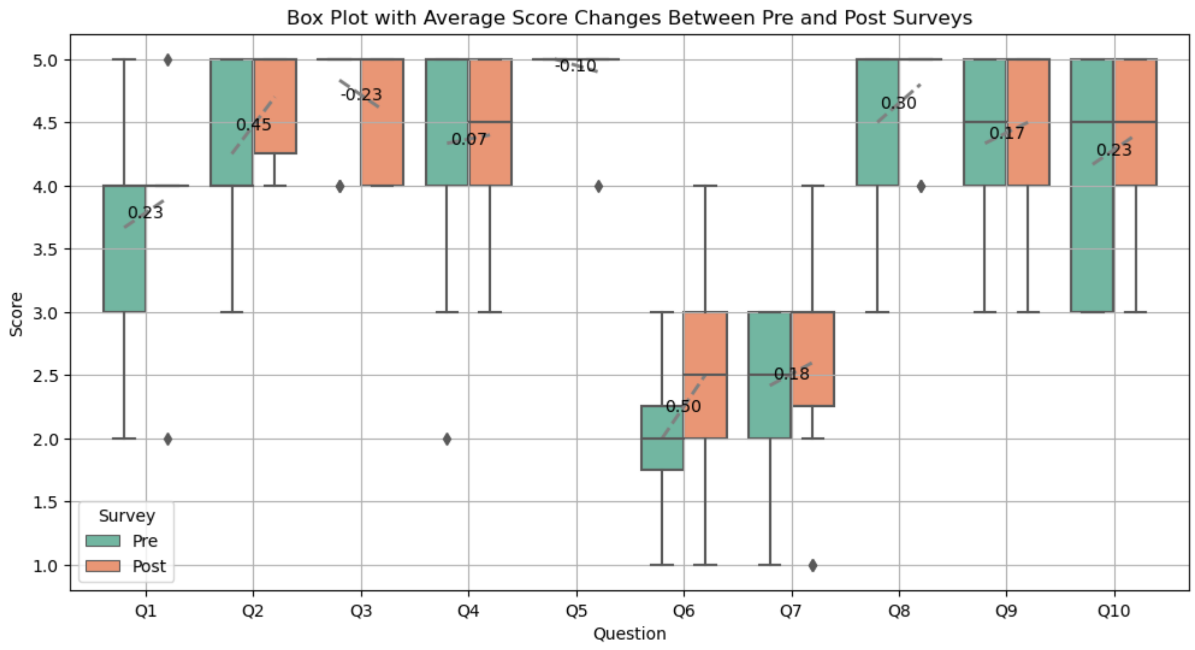

Figure 2: Boxplots of the evolution of scores from pre- to post-course surveys. The targeted constructs are intrinsic motivation (Q1 - Q2), general motivation (Q3), extrinsic motivation (Q4 - Q5), self-efficacy (Q6 - Q7 - Q8), and belonging (Q9 - Q10).

Despite the very limited dataset of 12 students, we observe the following trends: the intrinsic movitation (+0.34 points on average) has the largest increase. Theories of learning often attribute one of the largest roles to this contruct in a learner's drive and efficacy to learn.

General motivation (-0.23) and extinsic motivation (-0.02) are the two only constructs not gaining in score after the course. A explanation could reside in a slight transfer from extinsic ('we need to study to pass the exam') to intrinsic ('programming is an interesting skill and activity') motivation for the learners. Q5 in particular went from a score of 5.0 (pre) to 4.9 (post) and read 'I believe that programming skills will be beneficial for my future career.' Only 1 student answered '4' instead of '5' after the course, perhaps indicating a slight reconsideration of future professional prospects.

Self-efficacy (+0.33) and belonging (+0.20) show clear trends, which is very positive despite the short format of this pilot, and motivates repetitions of the course in at least similar length formats.

4 Conclusion

This pilot course on collaborative scientific programming was successful and resulted in great satisfaction for the students taking it, as well as for us leading it. There is large educational potential for repetition and expansion of this project in various departments of natural sciences.

Acknowledgements

This project has been run within the HKDir-funded *CoCreatingGFI* project (https://cocreatinggfi.w.uib.no) at the Geophysical Institute of the University of Bergen.
The authors warmly thank all participating students for their great energy and enthusiasm throughout the course.

References

Blaschke, L. M. (2012). Heutagogy and lifelong learning: A review of heutagogical practice and self-determined learning. *The International Review of Research in Open and Distributed Learning*, *13*(1), 56-71.

Boljat, I., Mladenović, M., & Jogun, N. M. (2019). Students'attitudes towards programming after the first year of implementing a new informatics curriculum in the elementary schools. In *ICERI2019 Proceedings* (pp. 9486-9495). IATED.

Deslauriers, L., McCarty, L. S., Miller, K., Callaghan, K., & Kestin, G. (2019). Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom. *Proceedings of the National Academy of Sciences*, *116*(39), 19251-19257.

Feliciano, J., Storey, M. A., & Zagalsky, A. (2016, May). Student experiences using GitHub in software engineering courses: a case study. In *Proceedings of the 38th International Conference on Software Engineering Companion* (pp. 422-431).

Glazunova, O. G., Parhomenko, O. V., Korolchuk, V. I., & Voloshyna, T. V. (2021, March). The effectiveness of GitHub cloud services for implementing a programming training project: students' point of view. In *Journal of physics: Conference series*(Vol. 1840, No. 1, p. 012030). IOP Publishing.

Kertész, C. Z. (2015, October). Using GitHub in the classroom-a collaborative learning experience. In *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)* (pp. 381-386). IEEE.

Singer, S. R., Nielsen, N. R., & Schweingruber, H. A. (2012). Discipline-based education research. *Washington, DC: The National Academies*.

Van Staveren, M. (2022). Integrating python into a physical chemistry lab. *Journal of Chemical Education*, *99*(7), 2604-2609.

Weurlander, M., Masiello, I., Söderberg, M., & Wernerson, A. (2009). Meaningful learning: students' perceptions of a new form of case seminar in pathology. *Medical teacher*, *31*(6), e248-e253.

Zagalsky, A., Feliciano, J., Storey, M. A., Zhao, Y., & Wang, W. (2015, February). The emergence of github as a collaborative platform for education. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing* (pp. 1906-1917).