Leveraging the Heterogeneity: Teaching Computing Skills to a Multidisciplinary Cohort With a Variable Skill Level

Katerina Michalickova

Early Career Researcher Institute, Imperial College London
k.michalickova@imperial.ac.uk

Jonathan Holt

Mechanical Engineering, Imperial College London
j.holt23@imperial.ac.uk

Chris Cooling

Early Career Researcher Institute, Imperial College London
c.cooling10@imperial.ac.uk

Overarching Themes

To an educator, this could be the stuff of nightmares. A classroom full of bright university students who have nothing in common - discipline, motivation or prior skill level. Their expectations vary, and they all expect you to help them make sense of computing for their future research. Regardless of their differences, all students chose your elective module.

This is precisely the context of a multidisciplinary and multi-skilled audience that our team of educators has committed to addressing since 2019. We designed an undergraduate module and subsequently a framework to navigate this delightful nightmare every spring term for the past five years, gradually refining our approach.

To make this chapter as relevant to programming educators as possible, we avoid fine details of our implementation and define a general framework that allows any educator to facilitate learning for a mixed-interest audience with varying skill levels and motivations. As a secondary effect, this chapter and the framework are relevant to other practitioners interested in student group project work dynamics and in integrating research into teaching practice.

Context: Elective Undergraduate Module Including a Brief History

We describe how we approached the design and delivery of a second-year undergraduate module in *Interdisciplinary Research Computing (IRC)*. The goal of the new module was to introduce our students to programming and related skills relevant to STEMM research and build a better understanding of their application. The plan was to use a variety of pedagogical approaches to motivate and upskill students with variable disciplinary interests and prior programming experience. This included, for example, supporting students' transferable skills and designing an assessment independent of incoming students' programming skill levels.

The IRC module follows the usual UK university format and runs for ten weeks; it is FHEQ level 6 and worth five ECTs. The format comprises 20 hours of face-to-face time evenly distributed over the term, and independent work and study. Our students come from various departments, but mainly from Mathematics, Physics, Life Sciences and Medicine. The module is an elective, and the mark does not appear on transcripts, which may downplay its importance compared to the departmental content. So far, we have been working with a cohort of 28 split into seven groups of four - a fairly standard size for the institutional portfolio of undergraduate elective modules. The teaching team consists of teaching fellows, research fellows, and graduate teaching assistants. Each team member contributes to teaching, marking and group supervision. The module lead coordinates the work, manages logistics, oversees iterative improvements, and supports the GTAs.

So far, we have completed five iterations of the module. This chapter describes the current state of the design, which was initiated in 2019 and underwent yearly improvements based on our observations and student feedback. The overall content and layout have not changed much, the majority of changes concerned supporting group work and introducing a new assessment type.

Contribution: Deep Dive Into the Student Reflections

In this chapter, we go a step beyond a case study. We present a mature design, analyse students' reflections on the module, and conclude with a general framework useful for educators in a similar teaching context.

Our design is the result of collaboration with a large teaching team, informed by systematic analysis of each module iteration and students' feedback. We describe the design in some detail and would like to bring the reader's attention specifically to the Connect 4 individual programming assignment. The module is designed to support individual progress, which makes it difficult to devise a single way to test their programming skills after completing their chosen programming topic. The Connect 4 assessment is our answer to this problem, which has been successfully used for several module iterations.

The last assessment of the IRC module is a reflection in which students discuss their experiences with the module, group work, and how their understanding of research computing has changed. We used reflections from the last two iterations (n=54) to conduct a thematic analysis and systematically identified key themes shaping the module experience. While thematic analysis effectively describes the cohort as a whole, we also examined individual experiences. This was prompted by reflective statements indicating the presence of student experience subgroups. We used a combined approach to explore this topic. This involved comparing self-reported prior skills against final grades and individual grades. We examined an outlier group and cross-referenced it with individual reflections to identify additional factors that shape the overall experience and ultimately determine the quality of learning.

The analysis significantly deepened our understanding of students' experiences. It helped to justify our past improvements and, most importantly, informed our future optimisations. To make this chapter as relevant as possible, we synthesised our experience to date and combined it with the analysis's findings to create a general framework for working with a multidisciplinary, multiskilled audience.

Framing: Rationale and Aims

Through the module, we take students on a learning journey that involves collaborative project work. Student groups design and implement an open-ended project. The activity is supported by a dedicated staff advisor who acts as a coach and an occasional technical expert. The planning stage takes about three weeks, followed by two weeks of writing and editing the project plan based on the advisor's formative feedback. The implementation is done in the second half of the module. Projects are presented at a poster session in the last week of the term.

We give the students considerable freedom to plan their projects, while mitigating the open-ended nature of the work through supervision and structured opportunities for reflection on progress. At the same time, we aim to maximise support for individual learning, making it relevant to the individual's needs. The goal is not simply to teach programming but to create a learning environment that mimics a real research process and fosters students' agency to respond to various learning situations.

A typical example of a project would be a machine learning experiment in which, after researching the topic and data sets, the students would conduct one or more ML experiments on a chosen data set and compare the outcomes with available data in the literature. While the description is short, the reader can appreciate that executing this project involves significant research, planning, programming, coordination and time management.

Our approach explicitly values independence. Many second-year students, however, are still adjusting to university-style learning. We have observed variable readiness for independent work and variable motivation that reflect the high demands of the regular coursework outside the elective. For many, this module is their first opportunity to work in a multidisciplinary setting. Therefore, we also need to address the variable readiness of our students. Over the years, we came to value a coaching approach to group supervision.

The module's design is based on multiple theoretical frameworks. While project-based learning principles (Thomas 2000) are central, interdisciplinary group work, facilitated by collaborative learning rooted in social constructivism (Daniels, Cole, and Wertsch 2007; Piaget 1970; Vygotsky 1978) is the primary dynamic shaping students' experiences. The group work aspect was somewhat underappreciated in the initial design, and consequently, most of the design changes we have introduced so far have aimed at facilitating team dynamics and fostering various transferable collaborative skills.

The design is based on four aims, which address the rationale above:

1. **Enable project-centred group work in interdisciplinary teams:** The central activity of the module is project work done in a multidisciplinary group setting. The group leverages unique student experiences, promotes knowledge sharing, and enables the development of transferable skills.
2. **Simulate the research environment:** We aim to simulate it as closely as possible. In addition to working on a project as part of a multidisciplinary team, students have dedicated advisors, attend guest lectures, discuss their project with visiting scientists, and present the project outcomes in a poster session.

3. **Enable independent learning:** Independent learning happens via online content and group projects (GW label in Table 1). We intentionally design the groups to be as diverse as possible in terms of discipline and prior skills, based on surveys conducted before the module begins.
4. **Carefully scope instruction time:** For the instruction part (LS and HO labels in Table 1), we carefully chose a key topics that provide students with a mental map of the research computing landscape.

The next section describes the implementation of our aims and the design of an appropriate assessment.

Design: Outcomes, Learning Experience, and Assessment

During the module design phase, we defined the following intended learning outcomes to constitute the basis of our students' experience. The outcomes are broad and do not address specific technical learning points. Instead, they create room for every student to have a unique experience connected to their own needs and interests.

1. Reflect on the relevance and explain the importance of research computing and data science in contemporary research.
2. Reformulate a real-world problem as a research question and break it down into achievable steps that can be resolved using programming and research computing tools.
3. Apply programming and research computing tools to solve computational problems and apply principles of collaborative software development.
4. Effectively communicate project progress and results to a multidisciplinary audience.

Table 1 below presents the module content in a nutshell. It is colour-coded based on the type of pedagogical approach used to deliver a particular module part.

Whole-cohort "Hands On" (HO) yellow

"Group Work" (GW) blue

"Lecture Style" (LS) green

"Administrative" (ADM) white.

Each face-to-face session provides a good mix of activities and most of the module involves active learning. The main component is group work on projects of students' own choice (GW). Individual learning takes place via pragmatic instructions (lecture style (LS)) on the fundamental topics and Python self-study (hands on tasks (HO)).

week	**Classroom time:** **5mins-30mins**	**30mins-55mins**	**65mins-90mins**	**90mins-115mins**
0	Skills Survey and Self-Assessment (ADM)			
1	Introduction, Module Information (ADM)	Guest Talk (LS)	Group Introductions and Contract (GW)	Meet your Disciplinary Colleagues (HO)
2	Python and Group Work Q&A (HO)	Final Project Introduction, Guidelines for Project Selection (GW)	Group Work - Identify Commonalities Between Disciplinary Examples (GW)	Guest Talk (LS)
3	Python and Group Work Q&A (HO)	Project Discussion (GW)	Guest Talk (LS)	Introduction to Git (HO)
4	Project Plan Discussion, Group Check-in (GW)	Python and Group Work Q&A (HO)	Introduction to GitHub (HO)	
5	Project Plan Discussion (GW)	Connect 4 Tournament (HO)	Connect 4 Tournament (HO)	Gentle Introduction to Computer Architecture 1 (LS)
6	Gentle Introduction to Computer Architecture 2 (LS)	Visiting Scientists (GW)	Project discussion, Group Check-in (GW)	Finalise Git and GitHub Project Setup (GW)
7	Visualising Data in Python (HO)		Visualising Data Practice (HO)	Group Work with Advisors (GW)
8	Introduction to the Command Line and Command Line Practice (HO)		Group Work with Advisors, Group Check-in (GW)	
9	Introduction to High Performance Computing (LS)		Group Work with Advisors (GW)	
10	Poster Presentations (GW)			Prizes and feedback (LS)

One of the most challenging aspects of the module was creating a fair and effective assessment, considering that students start with varying skill levels. Project work and group assessment can be somewhat unreliable as a measure of students' individual progress (Almond 2009). For this module, we chose a combination of group and individual assessments. Students submit five assessments, three individual and two group ones (45% and 55%, respectively). In the individual assessments, we ask students to reflect and experiment with coding in low-risk environment. The group assessments consist of project plan followed by project presentation via poster and GitHub repo.

Group Work and Group Assessment: From Forming to Storming

We survey students about their coding experience before the module and create interdisciplinary groups with mix of skill levels. Groups are introduced during the first week to maximise time for group forming. Each group has an assigned member of the teaching team with a role of an advisor who is available to them during the face-to-face sessions as well as for any other group meetings.

The initial meeting is structured to facilitate group formation. After introductions and an icebreaker, students are encouraged to collaboratively develop a group contract and establish regular meeting times outside the face-to-face module sessions. We encourage the students to start brainstorming early, so they have ample time during the first half of the module to refine the project plan. We advise them to identify a project that plays to their collective strengths and carefully consider who will take on which task.

A significant portion of time is spent on developing a project proposal with background research, clear milestones including work assignment, defined minimal working product versus extensions, and plans for knowledge sharing. The work does not start until a detailed plan is in place. While we have encouraged individual approach of staff members to group advisory role in the past, we have moved toward stronger coordination across the teaching team and using coaching-like methods.

We group assess an open-ended project proposal and a project presentation via a poster and collaboration documented in a GitHub repository. These assessments emphasise collaborative work, the ability to iteratively plan and improve a project, and deliver a minimal working product. The teaching team reserves the right to mitigate the marks when the group members are not contributing evenly.

The teaching team regularly discusses and monitors the group's progress. These discussions have become a central part of the weekly planning sessions, providing an opportunity for the staff to share experiences with our teaching assistants, who tend to change over the years.

Supporting and Assessing Individual Skills Development: Options and a Low-Risk Environment

Since students join with variable programming skills, we give them a choice from several topics that they commit to study during the first three weeks of the course, with a time commitment of 3-4 hours a week. This enables them to progress their knowledge in the direction that they find useful. Our teaching team has developed a publicly available suite of Python courses with numerous hands-on exercises that are suitable for self-study: *Introduction to Python for Researchers, Intermediate General Python, Object-Oriented Python, Numerical Computing with Python, Plotting with Python, and Data Processing with Python* (ECRI 2026). Self-study is supported by a dedicated channel, weekly Q&A sessions, and quizzes. Students report on their progress in the third week of the module via a short written reflection. Here they demonstrate their progress, and practise for the final course reflection.

Designing a motivating and fair approach to develop and assess individual programming skills in our heterogeneous cohort took us some time. Initially, we avoided individual programming assignments altogether. For the last four module iterations, we have been using a creative design based on the Connect 4 game. Students are given a GitHub repository with a simple, modular implementation of the Connect 4 game. Their task is to study the design and figure out how it works. Next, they edit a function which defines a strategy by generating the next move based on the current state of the board. This function already works, but uses a poor strategy. Students' new code can be immediately tested against a provided random strategy opponent function.

Once the student has developed their code, their mark can be generated repeatedly by automatically playing the student's strategy against this random strategy many times and recording how many games it wins. Once a student figures out how to set up the code, how it works and how to test it, they have already enhanced their learning, especially if they are new to Python. The marking is weighted so students will pass the assignment with a simple strategy. Students who are interested in developing a more sophisticated strategy can spend more time coding and achieving higher marks. Since the assignment counts for 10% of the mark, the advantage of more experienced coders is not disproportionate. We celebrate students' achievements with a Connect 4 tournament between the students' strategies, with an anonymous participation option.

As the last task of the entire module, each student completes a two-page reflection on the module experience and their personal growth. These reflections close the learning loop for individual students and provide a rich source of data for further analysis.

Learning From Students' Reflections

As indicated, the final student reflections on the module serve a two-fold purpose. They help students organise their thoughts on the experience, reason about their contributions, and reflect on what went well and what did not. For the teaching staff, the reflections present a good opportunity to learn about the effectiveness of our design and suggest improvements. For this chapter, we performed a qualitative and quantitative analysis on a set of 54 reflections from the last two iterations of the module (to prioritise the most evolved design). We aimed to 1) draw general conclusions about the module and identify topics that generated the most engagement, and 2) identify if any groups of students do not experience the module as intended and may need a particular adaptation or support.

Thematic Analysis: The Cohort Voice

To frame the reflective assignment, students received instructions on the reflective format and a list of topics they may want to discuss. Some topics were frequently discussed, some hardly mentioned, and some reflections were on topics not present on the list. Students were assured about the use of their reflections to assess the format and not the content of their reflections to encourage honest disclosure. Overall, most reflections subjectively read as broadly neutral, with 9% coming across as mostly negative and 20% as mostly positive. The overall descriptive content of reflections was high, indicating that, for many students, this was the first time they engaged in a reflective exercise. Using a classical thematic analysis approach (Braun and Clarke 2006), we pinpointed three dominant themes that characterise the students' module experience: 1) general opinions on the module as a whole, 2) group work dynamics, and 3) suggestions for future cohorts. Table 2 summarises three main themes and sub-themes.

For the first theme, students appreciated the module. The word "challenge" appeared often, followed by an expression of satisfaction from learning during the process. The module overall is perceived as challenging overall, however students' favourite components were the most complex ones - guest talks, coding assessment, or high performance computing. These components but one are generally not part of the assessment and could become victims of strategic learning. We are happy to report that we found no evidence of this. We were also satisfied to find many unprompted statements indicating the students appreciated the usefulness of the module - either for their respective degree course content, future professional progression or simply as acquiring transferable skills. Interestingly, we observed that only about half of the students chose to elaborate on the evolution of their understanding of computing's place in the research space. There were a few negative comments that indicated a mismatch between expectations from the module and the module content. Most of these students also acknowledged that, after completing the module, they understood the design intent.

For the second theme, the most frequent statements in the reflections concerned group work and dynamics. Group work became a major part of our students' experience, deemed useful and rewarding, yet also difficult g to the point of frustration. We consider this the most important outcome of this analysis. It was revealing that, through the module iterations and reflections, we observed numerous other factors contributing to the heterogeneity of the groups. We have also noted a strong signal from students more experienced and motivated than others: they were frustrated by their peers' inability to contribute on the same level, yet they were also frustrated by not being able to engage them.

For the third theme, students' suggestions for future cohorts were in line with what we recommend and tried to promote during the module. For example, they highlighted a common issue that we have been trying to mitigate - work is often done later than would be optimal. Students know it and wish they did not have to rush at the end. We will use these recommendations directly in our teaching.

Table 2 (next page): Summary of the main themes identified in student reflections. Each theme is divided into several sub-themes, and short explanations and representative quotes are included. Average frequencies per reflection for themes and sub-themes are also included.

Theme / *Subtheme* *(avg. freq.)*	**Representative topics and notes**	**Representative quote**
Opinions on the module as a whole (4)		
General appraisal (1)	Positive or very positive, often using word challenge, a few unmet expectations statements that acknowledge the understanding of intent behind the design.	*"Much appreciation to the absolutely encouraging and skilful teaching staff, and this one-term study is one of my highlights in my university career!" (2023-2024)*
Highlights (1)	Overall, students highlighted challenging topics as most appreciated or interesting - talks by guest scientists, followed by Connect 4 assignment, GitHub and high performance computing	*"One of the most valuable skills that I learnt during the project was learning how to use the collaborative features on GitHub." (2023-2024)*
Usefulness (1.3)	Unprompted (and frequently), students talked about the usefulness of the module - for degree course, future professional progression or for acquiring transferable skills.	*"This experience allowed me to grow both technically and personally, as I navigated the challenges of working in a diverse group and tackled complex problems that required creative problem-solving." (2024-2025)*
Evolution of understanding of research computing (0.7)	Only half of students chose to elaborate. Good understanding - research computing is cross-cutting and requires much more than developing programming skills.	*"This module changed the way I see computing. I used to think of research and computing as two separate things, but now I realise that computing can be a powerful research tool in nearly every field." (2024-2025)*
Group work (4.5)		
General appraisal (2.7)	Group work was commented on most frequently and prompted frequent honest disclosure. A mix of positive statements and expressions of frustration. Multiple factors influencing group work were mentioned: expectation, scheduling and degree course load, motivation, self-efficacy, ethic, health (physical and mental). Challenges: work division causing some students to miss out on learning opportunities, not speaking early enough when group dynamics was suboptimal.	*"I cannot overstate the value of effective communication within our team. Despite our diverse academic backgrounds, we managed to find a common language through the shared goal of the project." (2023-2024)*
Self-appraisal (1.3)	Balanced mix of positive and self-critical. Positive: good progress in technical and professional skills. Negative: lack of confidence (often coupled with less prior experience). Statements that need attention from the teaching team: (1) willing to learn but not finding a way of contributing to the project meaningfully - being an observer, (2) the experienced students completing disproportionate amount of work due to their self-reported inability to involve others effectively.	*"I recognised a need for greater confidence and critical engagement, especially when faced with uncertain methodologies proposed by my team" (2023-2024)*
Appraisal of others (0.5)	Most comments focused on the work ethics of others and came from students who shouldered more work. Very few positive statements about individual colleagues (positive statements generally refer to the whole group)	*"As my group members were inexperienced with coding, whereas I felt more confident, it quickly became clear that I would have to take leadership of the project. I was happy to do this." (2023-2024)*
Message for future cohorts - "I wish I did this..." (1.2)		
Points for group work (0.6)	The statements clustered into the following categories (by order of frequency): select a motivating project for all group members, manage your time better and avoid rush at the module end, be strict about accountability, realistic scoping, divide work fairly and strategically, communicate efficiently and honestly, make an effort to get to know your colleagues.	*"Maintaining strict goals from the beginning and conducting regular meetings so that the project remains on course." (2024-2025)*
Points for personal actions (0.6)	A highly mixed group of statements that show good understanding of what is needed to successfully complete the module: be proactive, don't delay work, learn some Python before the module, ask questions, learn from your peers, realise that everyone has something to contribute, realise the strengths of collaborative work, trust yourself, have open and curious mind, and keep good notes and log of activities.	*"You are studying and working with amazing teammates, please embrace the learning process with an open mind and a curious heart. Don't be afraid to ask questions, seek help from your peers and instructors, and step out of your comfort zone" (2023-2024)*

Student-Centred Analysis: Finding the Essential Details

Qualitative thematic analysis does not give insight into subgroups of students. Thorough analysis of reflections coupled with group membership, prior skill and achievement enables us to identify and focus on supporting groups of students who may find the design not productive.

In order to better understand the diversity of the cohort, we looked at overall module grades grouped by self-appraised programming experience and their background, as shown in Figure 19.1. There is a uniformity of grades that is broadly independent of programming experience and disciplinary background. This was a goal of the assessment design, which prioritises group work and non-technical reflections.

A clearer picture of the students is apparent when we focus on the Connect 4 assignment, which was the only individual programming assessment of the module. Figure 19.2 shows grades grouped by students' self-appraised programming experience and their discipline. Broadly, we see that students from life sciences and medicine came into the course with lower programming experience and overall attained lower marks than their STEM peers. The fact that students with no prior programming experience gained good grades raised concerns regarding the use of artificial intelligence. In an anonymous survey regarding Generative AI use, all students who used AI for this assignment claimed to understand what was being suggested and provided guidance to the AI agent. This confounding factor led us to focus on students who achieved low marks for further analysis.

Figure 19.1: Graph showing the relationship between pre-module self-appraisal and overall module grade.

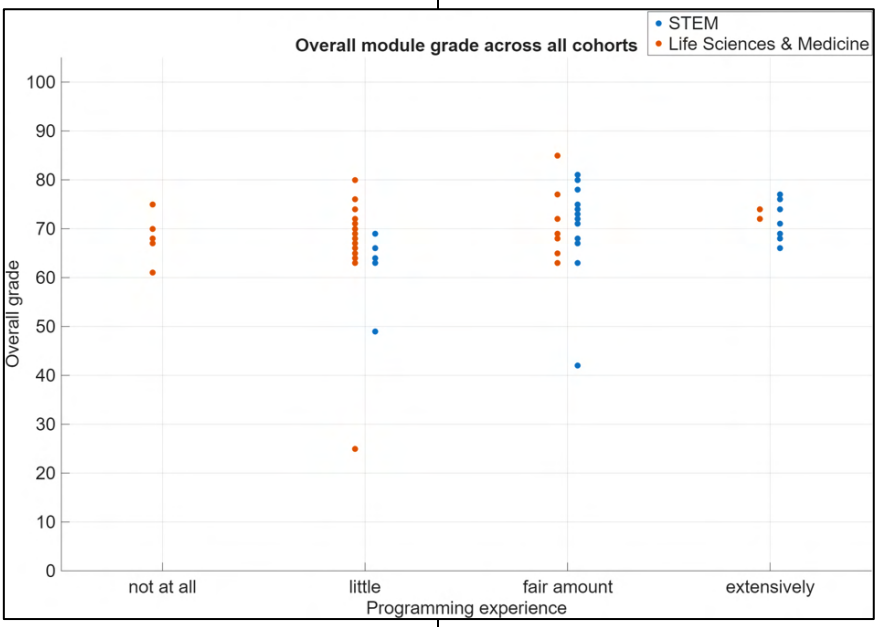
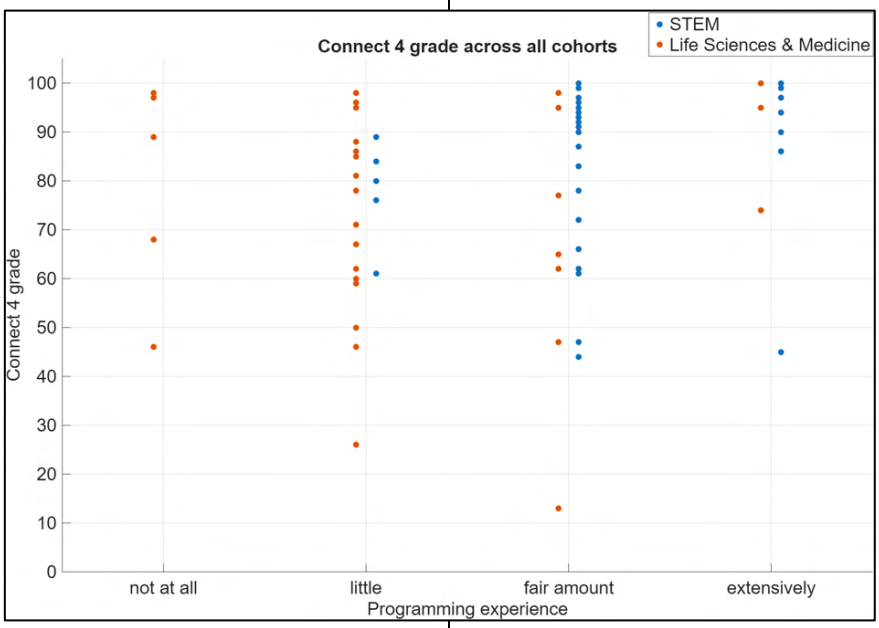


Figure 19.2: Graph showing the relationship between pre-module self-appraisal and Connect 4 programming assignment grades.



We took all students whose Connect 4 grades were lower than 60% and attempted to identify commonalities in their experience in the reflections, focusing on comments about their own performance and the project. The assumption was that this group of students was most likely to run into difficulties and not be able to contribute to the programming aspect of the project.

Project Design Determines Possibilities For Contribution

Out of the 12 students in this group, nine were from Life Sciences or Medicine. Seven mentioned they were not skilled enough to contribute to the programming aspect of the project and ultimately took (or were given) the role of poster designer or researcher. In fact, across the entire cohort, the least experienced students tend to be relegated to the tasks with the lowest friction and, as a result, the lowest learning potential. Even students with self-reported extensive programming experience, but who performed considerably worse than their teammates in the Connect 4 assignment seemed to be given these tasks.

One student, who took a leadership role in their team, reflected that they had been *"very stubborn"* to seek help from their teammates because they knew they *"would not be able to lead as much in the actual coding"*. Stepping back to the most comfortable role led many students to an underexposure to programming, and, ultimately, some regret. The same student continues, *"I wish I had fully engaged with the methods of data analysis (... which would have) allowed me to contribute more to the decision making"*.

Two out of the 12 poorest performing students had little programming experience and no subject-specific knowledge relevant to their project. Both worked on their team's poster and displayed the most dissatisfaction with their progress and experience. One Medicine student noted their ideas were "quickly dismissed", with their team members "believing that working with medical images would be too complex". Their project ultimately became considerably more complex, but leveraged only the interests of the two more experienced programmers.

An advanced project rooted in the discipline of the most experienced group members led to poor learning outcomes for the least experienced members. This was exacerbated by a (self-reported) non-compromising attitude of the expert members, leading us to believe that we must be more proactive in educating our students in group dynamics, especially in finding compromises.

Self-Assessment Mismatch

We also looked for clues in reflections that indicated that students overestimated their programming experience with ultimate negative consequences. Approximately one out of five students described themselves as having a fair amount of experience but reflected that they expected a basic computer science course or that they did not have sufficient time to learn enough to contribute to their project.

We continued this theme by analysing the reflections of the 12 students with a Connect 4 grade below 60%, which also revealed a group that is likely to need additional resources and support. Three of these students from Life Sciences and Medicine severely overestimated self-reported programming experience, which affected their group placement and resulted in a poorer experience and learning outcomes. This has led us to plan improvements to the self-reporting guidelines and the development of a scaffolded project with extra support.

Developing a Generic Framework for a Multidisciplinary and Multi-Skill Context

Based on our experience and analysis with multidisciplinary and multi-skill context, we created an adaptable framework to inspire fellow educators working in similar situations. Our framework is informed by the original module design elements and enriched by our data analysis, which suggested new interventions.

A multidisciplinary and multi-skill learning and teaching context represents an enormous opportunity. Students will be exposed to a wider range of problems, methods and ways of thinking. This context can become a vehicle for better outcomes. For our module, we used a flexible and varied design that was based on group project work. This design created an opportunity for bespoke learning and enabled us to bring research into the classroom.

Before undertaking thematic analysis, we have iteratively improved the model based on student feedback and, sometimes, anecdotal evidence. This chapter describes a deep dive evaluation that benefits any module from time to time. This systematic analysis of student reflections with other module data demonstrates that such an analysis can produce a wealth of new information and perspectives.

Using our case study, we draw generic lessons on key aspects that were uncovered by our analysis: supporting individual skills development, project group facilitation, and the overall narrative presented to students. The list indicates that our context requires a shift in focus and a different approach in data analysis than teaching a single-discipline audience with similar preexisting skills.

Individual Skills Development: Our Suggestions Based On Our Framework

The multi-skill context requires creating conditions for supported individual learning opportunities. Increased support requirement puts extra strain on resources but may be streamlined into a few well-defined learning tracks. For our modules, we use self-study of locally developed Python short courses that are normally taught to a postgraduate audience. Here are our suggestions for other teachers working with a multidisciplinary audience with varied prior skills.

- **Assume a base level of skill and be proactive in communicating this expectation clearly**. If possible, set a prerequisite for learning. For example, before project work starts, all students must have completed a specific introductory Python course.
- **Set individual learning with ample options for help**. Use inventive means to motivate learning, such as tournaments and coding challenges.
- **Use self-reporting to map skill level and monitor the outlier skill groups**, especially the least experienced students.
    - Not all students from the least experienced group will miss learning opportunities, but this group is most at risk. Pay attention to self-efficacy development opportunities, suitable project choice and division of work in the groups.
    - Include an opportunity for students to reflect with peers who closely match a learner profile to generate a self-efficacy-enhancing opportunities.
    - If meaningful participation of students in mixed groups cannot be guaranteed, consider establishing outlier groups, for example, a scaffolded project for coding beginners.

Project Groups Facilitation: Our Suggestions

This is a key area that was identified in our study. Project work facilitation is a large part of implementing group work and largely determines the quality of students' experience. It tends to be neglected compared with technical skills development. Our suggestions are:

- **Prepare students for self-assessment**. Ask for evidence for skills.
- **Assemble mixed groups** and aim for maximum distribution of disciplines and skills.
- Pay **extra attention to beginners' placement**. For example, a single beginner in an advanced group is not a good choice.
- Factor in **resources for group facilitation**. It can be equally demanding on staff time as supporting research aspects of project work.
- Set and **promote culture based on values facilitating effective group work**, and honest reflection with group peers.
- Plan for **systematic and transparent staff and student preparation for group work** and group work facilitation, including written materials; for example, group contract, guidelines for collaboration, project selection criteria, dividing work in meaningful and inclusive ways, reflection tools, appreciating, encouraging and including others, approaching difficult conversations, and a process for group check-ins.
- For open-ended projects, develop a **project choice rating framework** that helps to identify projects that provide appropriate motivation to all and do not lean heavily into the expertise of one or two individuals. Projects requiring advanced discipline-specific knowledge are not recommended. Projects should provide an opportunity to practice cross-cutting computing tools such as machine learning or high performance computing.
- Be extra careful to **manage engagement issues**. If a group member stops participating, reassure the remaining group members that a good outcome is attainable even with a smaller group, as the impact of low or variable participation can be demotivating. Fall back on clear guidelines for intervention.

Holding It All Together: Our Final Suggestions

The learning design in this context is not straightforward, and it is important that students understand why we build it this way. There are several themes running through the module that require students' attention. At the very least, students navigate the individual and group learning journey at the same time. Based on our experience and data analysis, the following points improve student experience and help them organise their thoughts.

- **Clear narrative and signposting of objectives during the module** – do not hesitate to share with your students why you organised the module in a particular way, including examples of what worked and what did not work in the past.
- **Share suggestions from previous cohorts** with your students and any other relevant testimonials.
- Students indicated that they enjoyed the difficult parts of the design the most. Apart from one task, these were not connected to any assessment. It leads us to speculate that it is a good design decision to **remove the burden of assessment and increase accessibility of the most challenging parts** of the design that are not connected with project work.
- Finally, **plan for module data analysis** and consider the type of questions that you need answered. We would not have been able to get the insights described in this chapter from the student feedback alone.

In Closing

In a few words, the multidisciplinary and multi-skilled context presents great opportunity for students to share experiences and learn together while working on a project. When projects are carefully scoped and group have access to dedicated advisors, the design of the project can be made appropriately ambitious while including all group members. It is essential that students' efforts must be matched by ample support and always accompanied by clear and reinforced guidelines for group work. Our experience suggests that when we pay equal attention to all three elements - facilitating individual development, supporting and guiding group work, and presenting a clear narrative - the multidisciplinary and multi skilled module can become a transformative experience.

References

Almond, Richard J. 2009. "Group Assessment: Comparing Group and Individual Undergraduate Module Marks." Journal Article. *Assessment & Evaluation in Higher Education* 34 (2): 141–48. https://doi.org/10.1080/02602930801956083.

Braun, Virginia, and Victoria Clarke. 2006. "Using Thematic Analysis in Psychology." Journal Article. *Qualitative Research in Psychology* 3 (2): 77–101. https://doi.org/10.1191/1478088706qp063oa.

Daniels, Harry, Michael Cole, and James V Wertsch. 2007. *The Cambridge Companion to Vygotsky*. Book. Cambridge University Press. https://doi.org/10.1017/CCOL0521831040.

ECRI, Early Career Researcher Institute. 2026. "Research Computing and Data Science Topics." Web Page. https://www.imperial.ac.uk/early-career-researcher-institute/learning-and-development/courses-by-programme/research-computing-and-data-science/.

Piaget, Jean. 1970. *Science of Education and the Psychology of the Child. Trans. D. Coltman*. Book. New York: Orion Press.

Thomas, JW. 2000. "A Review of Research on Project-Based Learning. San Rafael, CA: Autodesk." Generic.

Vygotsky, Lev Semenovich. 1978. *Mind in Society: Development of Higher Psychological Processes*. Book. Harvard university press.