Part II — Teaching Programming in Context: Design and Practice

Serveh Sharifi Far

School of Mathematics and Maxwell Institute for Mathematical Sciences, University of Edinburgh
Serveh.Sharifi@ed.ac.uk

Christopher Aldous Oldnall

Edinburgh Medical School, University of Edinburgh
chris.oldnall@ed.ac.uk

Effective programming education requires thoughtful design that responds to learners and their context. Decisions about technologies, learning activities and collaboration are shaped not only by programming concepts, but also by the characteristics of learners themselves. This is particularly important in contemporary programming education, where learners come from different disciplinary backgrounds, possessing a variety of prior experience. The chapters in this section explore three interconnected aspects of programming education. First, they consider how learning can be strengthened through collaboration, peer interaction and the dynamics of learning within groups. Second, they examine how educators can support learners as they begin their programming journey and develop confidence with programming languages and tools. Finally, they explore how creative and unconventional teaching approaches can broaden participation, increase engagement and encourage learners to think differently about programming and computational problem-solving.

Together, these perspectives demonstrate that programming education extends beyond the choice of programming language or curriculum content, highlighting the interplay between collaboration, learner development, and experimental teaching approaches. The chapters illustrate how active choices of programming tools and environments support the development of confidence and technical competence. From group-based learning and hybrid classrooms to visual, musical, and unplugged approaches to computational thinking, these contributions illustrate the wide range of pedagogical strategies available to programming educators. Collectively, they offer practical insights into designing learning experiences that are responsive to learners, contexts, and educational goals.

Theme — Drawing on Group Dynamics for Programming Education

This theme explores how collaboration, peer interaction, and cohort diversity can be leveraged to enhance programming education. Together, the chapters examine approaches that promote inclusive participation, strengthen engagement, and support learning across a range of educational contexts.

Peer Programming in Action: Pair Programming in Larger Groups (Noè, McManus, and Xu 2026)

This chapter expands on the concept of pair programming by introducing “Peer Programming,” applying collaborative principles to larger groups. With five case studies from a 20-week course, it offers a practical look at the benefits and challenges of implementing peer programming.

This insight is helpful for teachers who aim to boost student engagement through collaborative methods.

Leveraging the Heterogeneity (Michalickova, Holt, and Cooling 2026)

This chapter tackles the complexities of teaching a diverse cohort with varied skill levels and motivations. The authors share a refined framework for addressing these differences, combining quantitative and qualitative analyses to improve teaching techniques.

Educators facing similarly diverse classrooms will find this chapter valuable for its insights and adaptable strategies, particularly those interested in integrating research into teaching practice.

Learning Together Across Modes (Alex, Llewellyn-MacRae, and Orzechowski 2026)

This chapter presents an ambitious approach for integrating onsite and remote students in hybrid teaching. By addressing technological, social, and pedagogical challenges, the authors craft a successful model for fusion teaching, where onsite and remote learners collaborate seamlessly.

This chapter is helpful for educators seeking to push the boundaries of online learning and explore new avenues in hybrid education.

Theme — Teaching Programming to Novices: Python and Beyond

This theme focuses on supporting learners as they develop programming knowledge and confidence. The chapters explore the use of Python for data analysis, the role of tools such as Jupyter in programming education, and the challenges of transitioning to additional programming languages. Together, they provide practical insights into designing learning experiences that support both immediate application and longer-term development.

Teaching Python as a Computational Tool (Sharifi Far, Qu, and King 2026)

This chapter provides a practical guide for integrating Python into an introductory data science course. The chapter addresses the challenge of teaching statistical and programming skills together, offering a step-by-step approach that helps students connect technical coding skills with data science concepts.

Educators looking to balance theory with practical application in data science education will find this chapter useful.

Notebooks for Novices? Pros and Cons of Jupyter (El Gemayel, Budiarto, and Bell 2026)

This chapter dives into the role of Jupyter Notebooks in programming education. The chapter explores the benefits of using Jupyter for teaching core programming concepts without overwhelming students with technical setup issues. It also discusses potential pitfalls and presents ways instructors can optimise the learning experience.

This chapter is a valuable read for those teaching in Python tech-heavy courses.

Bridging Languages: Teaching C to Python Novices (Skipsey et al. 2026)

This chapter tackles the challenges of introducing a second programming language to students already familiar with Python and Jupyter. It focuses on the transition to learning C, highlighting typical difficulties and offering solutions to ease the process.

This chapter is helpful for educators who aim to guide students through diverse programming paradigms, teaching them to adapt and expand their programming toolkit.

Theme — Thinking Outside the Box: Designing Engaging Learning Experiences

This theme focuses on supporting learners as they develop programming knowledge and confidence. The chapters explore the use of Python for data analysis, the role of tools such as Jupyter in programming education, and the challenges of transitioning to additional programming languages. Together, they provide practical insights into designing learning experiences that support both immediate application and longer-term development.

Co-Designing Programming Teaching Experiences (Quinlan and Queiroz 2026)

This chapter draws inspiration from role-playing games and anarchist pedagogies to create dynamic and collaborative learning environments.

By organizing learning activities without traditional hierarchies, it helps educators design courses that foster student agency and teamwork.

Removing Barriers by Programming Without Computers (Cutting 2026)

This chapter tackles programming education barriers by teaching computational thinking without computers. By using games and tangible resources, it provides a low-pressure environment that emphasizes problem-solving skills.

This approach is valuable for diverse groups, including non-traditional learners.

Sound and Music in Programming Pedagogy (Mudd et al. 2026)

This chapter connects sound, music, and programming and how these can be combined in programming teaching. The authors provide a rich set of resources in numerous programming languages that educators can adapt and use in their teaching.

This chapter is for you if you are interested in sound, music, or programming and would be excited to bring these domains together.

Ways to Teach Online (Cooling et al. 2026)

This chapter explores practical approaches to online and blended programming education, drawing on experiences from MOOCs, self-paced learning, and professional education programmes.

It offers insights into course design, learner engagement, assessment, and feedback in digital learning environments.

Seeing Before Coding, Doodling Before Doing (Goopy 2026)

This chapter focuses on a data visualization course that encourages students to sketch and doodle. The approach described in this chapter nurtures critical thinking and creativity, urging students to conceptualize data visually before coding - a crucial skill in an AI-driven world.

This chapter is for educators who are interested in teaching data visualisation as a challenging and creative activity that requires engagement and real-world thinking rather than code output.

References

Alex, Beatrice, Clare Llewellyn-MacRae, and Pawel Orzechowski. 2026. “Learning Together Across Modes: Online and on-Site Pair Programming in a Fusion Course.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Cooling, Chris, Nick Jayanth, Samantha Alvarez-Madrazo, Leila S. Shafti, Joseph El Gemayel, and Lucia Michielin. 2026. “Ways to Teach Online: Lessons Learned Through Experience.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Cutting, David. 2026. “Removing Barriers by Programming Without Computers.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

El Gemayel, Joseph, Arif Budiarto, and William Bell. 2026. “Notebook for Novices? Pros and Cons of Jupyter.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Goopy, Suzanne. 2026. “Seeing Before Coding, Doodling Before Doing: How Teaching Data Visualisation Transforms the Way Students Think.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Michalickova, Katerina, Jonathan Holt, and Chris Cooling. 2026. “Leveraging the Heterogeneity: Teaching Computing Skills to a Multidisciplinary Cohort with a Variable Skill Level.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Mudd, Tom, Charlotte Desvages, Mike Taverne, Matthew Hamilton, and Yashique Chalil. 2026. “Practical Approaches to Using Sound and Music in Programming Pedagogy.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Noè, Umberto, Franziska McManus, and Eileen Y. Xu. 2026. “Peer Programming in Action: Pair Programming in Larger Groups.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Quinlan, Maeve Murphy, and Francisco Oliveira de Queiroz. 2026. “Dungeon

Crawlers and Anarchists: Co-Designing Programming Teaching Experiences." In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Sharifi Far, Serveh, Ruini Qu, and Stuart King. 2026. "A Practical Guide to Teaching Python as a Computational Tool in an Introductory Data Analysis Course." In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Skipsey, Sam, Gordon Stewart, Jeremy Singer, and David Cutting. 2026. "Bridging Languages: Teaching c to Python Novices." In *Teaching Programming Across Disciplines*. Edinburgh Diamond.