Part I — Theories and Applications of Teaching Programming

Brittany Blankinship

Edinburgh Medical School, University of Edinburgh
b.blankinship@ed.ac.uk

Franziska McManus

Edinburgh Medical School, University of Edinburgh
z.mcmanus@ed.ac.uk

Umberto Noè

School of Philosophy, Psychology and Language Sciences, University of Edinburgh
Umberto.Noe@ed.ac.uk

As a community, our philosophy towards teaching is grounded in the practice and framework of pair programming (if you haven't heard of this yet, read on!). This framework for structured group work, taken from industry, was the first aspect of our teaching which we shared amongst ourselves at the University of Edinburgh and became a uniting feature of our teaching. Today, pair programming features in all of our teaching as a standard. Our teaching and working life centres around the philosophy of working together to create a shared product and vision – that is, working openly, kindly, and together.

In the first theme, you will get acquainted with what pair programming is and how it can be implemented across courses, levels of study, and disciplines. By the end of it, we hope it will convince you that pair programming is a framework worth getting excited about and perhaps something to implement in your own teaching.

The second theme focuses on a selection of core concepts in programming that are essential for building healthy programming habits. While learning a programming language's vocabulary and syntax is a fundamental part of learning to code, these chapters propose broader approaches that nurture more general skills required to be successful coders and learners (hello transferrable skills).

The third theme acknowledges that being a good learner necessarily means thinking critically and making connections across ideas and topics. As educators, we expect our students not only to produce code that executes, but also to interact meaningfully with it and the wider context. It discusses what it means to teach and learn programming in a world of Generative AI, addressing the good, the bad, and the ugly. You will not find another debate but a range of viewpoints as they exist in our teaching community. These chapters will resonate with educators, address concepts that have always been fundamental to programming, and hopefully reveal a few surprises along the way.

Finally, as programming educators, you may have an idea of what concepts and mindsets you would like to teach your students. But how do you actually implement them in a pragmatic, engaging, and effective way? In the fourth and final theme of Part I, our authors tackle core pedagogical issues from the perspective of teaching programming:

- How do we get total beginners learning a tool they never expected to use in the first place?
- How do you teach multiple complex concepts at the same time?
- How do you build confidence and create an inclusive and welcoming atmosphere for your students?

Some of the concepts and ideas presented in Part I may be familiar to you already, but we hope that you will gain either a framework to describe what you are already doing or a new way to approach teaching.

Theme — Pair Programming: What? How? Why?

Where Do I Even Start With Pair Programming? (Orzechowski, Blankinship, and Banas 2026)

An introduction to pair programming in the form of a (written) podcast interview where you can read or listen. This chapter is a joyful and pragmatic approach to answering some of the most frequently asked questions around introducing pair programming in the classroom.

This chapter is for you if you are interested in pair programming as a teaching practice or you would like ideas for how to explain it to your students or colleagues.

Lessons From Pair Programming Across Disciplines (Desvages et al. 2026)

A walkthrough of the origins and practice of pair programming and what this has looked like for the authors in their disciplines. A highlight of this chapter is the fun illustrations that bring to life explanations of pair programming supported by student feedback.

This chapter is for you if you want to learn more about pair programming or are looking for a resource to explain pair programming to students, colleagues, and management.

A PhD Tutor's Reflections (Elliot 2026)

A PhD students' experience of teaching pair programming in an introductory R course. The author reflects on when pair programming works and when it doesn't, as well as the tutor's role in facilitating pair programming.

This chapter is for you if you are a graduate student or tutor on a course that uses pair programming, or are in a position to implement it in a tutorial setting. Course organisers will also benefit from the insightful reflections from the front line of teaching.

Theme — Building Healthy Programming Habits

A Practical Debugging Framework for Learners (Doig 2026)

An accessible and joyful introduction to debugging via the 5 B's framework. The sections of Brain, Book, Buddy, Boss, Break are expertly scaffolded and explained, turning this chapter into an adaptable teaching resource.

This chapter is for you if you teach any programming language and want your students to approach errors and debugging in a critical and independent fashion.

Empowering Students to Develop a Coding Mindset (Bray et al. 2026)

The most important skillset you can learn when coding is not the syntax of the specific programming language, but rather the "coding mindset". This chapter outlines practical suggestions and approaches for ensuring your teaching goes beyond syntax and focuses on developing students' mindset to allow them to effectively code in other languages in the future.

This chapter is for you if you are embarking on teaching your first programming course and want to develop the course beyond the syntax you might be teaching. Experienced programmers will also find the chapter interesting as a source for reflection on their own "coding mindset" and how it has developed.

It Depends: How to Develop Judgement in Programming (Wilson and Blankinship 2026)

Extending the idea of "best practice" in coding, this chapter asks the programmer to question the needs of their context rather than seeing "good" and "bad" coding practices in black and white. The central concept, that there are "forces" of coding, will be useful as

a teaching resource to help students write good code and consider why they are coding in the first place.

This chapter is for anyone who codes and wants to expand their approach to coding independently, consciously, and/or collaboratively.

Theme — Programming in a World of Generative AI

Hype and The Need for Responsible Compute (Ahern, Skipsey, and Riviera 2026)

Every new computational tool or AI model comes with exciting promises, but it also brings challenges. These often leave behind questions about fairness, who benefits from this technology, and who pays the hidden costs. As educators, we are placed in the middle of this hype, trying to help our students make sense of these tools.

This chapter will take you through the history of responsible computing through the lens of the United Nations (UN) Sustainable Development Goals. You will learn how issues like inequality, environmental harm, workforce exploitation, and bias can be part of new technologies. Along this journey, the authors share classroom activities, teaching ideas, and recommended readings you can assign in your own courses. This chapter is for you if you want your students not only to learn about fancy new algorithms, but also to be responsible and sustainable in their usage.

Teaching With the Emergence of Generative AI (Evkaya et al. 2026)

You are likely familiar with the fact that students can now generate working code in a matter of seconds with AI. However, you may also worry about what this means for assessments and question the value of your own teaching. This chapter invites you to rethink your approach to teaching and focus on what matters most when teaching students how to code. The authors talk about how Gen AI tools can help with exploration and feedback, while keeping some space for struggle and genuine understanding. They share examples of how to adapt your teaching and assessment without giving up on rigour.

If you are wondering about how to rethink your teaching or assessment approach for students coding with AI, this will be an interesting read.

AI, Voice, and Style in Programming Education (Wilson 2026)

The author puts into writing many facets of the impact of Generative AI on academic skills development that educators will have been uncomfortable about but may not have been able to put into words themselves.

This chapter is for educators who want to be able to communicate their concerns around the erosion of academic skills with the existence and accessibility of Generative AI. It will be a useful teaching resource to give to students as is.

How Teaching Programming Across Disciplines Can Instil Systems Thinking (Riviera et al. 2026)

Starting with the purpose and function of the university as an institution, this chapter takes you on a journey through the benefits of learning to program from a systems thinking perspective. If systems thinking is a new concept for you, you will find an intriguing introduction and a new way of looking at programming and the wider world (i.e., systems).

For many readers, this will answer the question of why it is a worthwhile endeavour to teach students programming skills (that they may not use again). As a teaching resource, this chapter would be excellent compulsory reading for students and a starting point for discussion-based tutorials.

Theme — Teaching Programming Courses: What Worked for Us

Reducing Technological Barriers in Programming Courses (Wadsworth et al. 2026)

Learning to code for the first time is fraught with challenges ranging from the technical, cultural, and practical. As an educator, how do you help your students when “the computer says no”? This chapter covers implementable steps for reducing barriers in programming courses, to ensure students don’t lose motivation to learn programming when faced with such issues, while ensuring instructors are not overworked.

This chapter is for you if you teach novice programmers and want to reduce friction in getting set up and working through the programming curriculum. With clear suggestions and a summary of pros and cons, it will allow you to pick the best setup for your own course!

Inclusion and Accessibility in Programming Teaching (Colquhoun et al. 2026)

Moving beyond standard prescriptions for inclusion and accessibility, this chapter takes a broader approach to pedagogical practice that fosters learning, allows vulnerability, and creates a space for all students to succeed. It provides a structured overview of the impact of different methods of delivery and a range of approaches that you can use dynamically depending on your student cohorts.

For programming educators who would like to make their classroom more welcoming, this chapter will offer practical approaches tailored to the experience of teaching programming across disciplines and being an adaptive teacher.

Overcoming Coding Anxiety (Oldnall et al. 2026)

For many students, anxiety is a central and paralysing part of their coding experience. This is particularly true for students in non-programming disciplines where educators may need to make a targeted effort to approach this head on. The authors offer practical suggestions and tips provided from their experiences that you can easily pick up and embed out of the box in your own teaching.

This chapter is for you if you have ever taught coding to anxious students or would like ideas on ways to make your programming course more accessible and fun for all students, regardless of background.

3 Stars and 1 Wish (Orzechowski et al. 2026)

With its origins in primary school teaching, this reflective practice framework will be a surprising and adaptable tool for use in higher education teaching, both in-person and online. This chapter introduces the idea of ‘3 stars and a wish’, where reflection is a flexible way of creating community that may replace existing discussion boards and feedback surveys. The authors present a case study and practical tips for implementing the framework.

This chapter is for you if you are an educator who would like to integrate continuous bite-sized reflection into your teaching or for anyone who is looking for ways to pulse-check student learning and progress.

Sequential vs. Simultaneous (Young et al. 2026)

Within many statistics modules, programming is presented as a tool which students need to learn quickly, engage with briefly, and then move on from. Yet, both statistics and programming come with a heavy cognitive load and are sources of angst, anxiety, and confusion for students. This chapter compares three approaches to teaching statistics and programming for students who need to know both. This chapter is for anyone designing (or re-designing) a course teaching both programming and statistics. With concrete examples from the author’s diverse experiences, this chapter presents a practical guide as well as an opportunity to learn from others who have been there before.

References

Ahern, Samantha, Sam Skipsey, and Leo Riviera. 2026. “Hype and the Need for Responsible Compute.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Bray, Neil, Samantha Alvarez Madrazo, Lucia Michielin, Nick Jayanth, and Sam Skipsey. 2026. “A Guide on Empowering Students to Develop a Coding Mindset.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Colquhoun, Rebecca, Samantha Ahern, Brittany Blankinship, and Patricia Loto. 2026. “Practices to Foster Inclusion and Accessibility in Programming Teaching.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Desvages, Charlotte, Brittany Blankinship, Umberto Noè, and Pawel Orzechowski. 2026. “Structured Group Work with Assigned Asymmetrical Roles and Switching: Lessons from Pair Programming Across Disciplines.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Doig, Mar. 2026. “Brain Book Buddy Boss Break: A Practical Debugging Framework for Learners.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Elliot, Sarah. 2026. “Pair Programming in Practice: A PhD Tutor’s Reflections.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Evkaya, Ozan, Hebatallah Shoukry, TJ Elmas, Laila Dabab Nahas, and Steven Watterson. 2026. “An Optimistic Outlook on Teaching, Learning and Assessment for Coding with the Emergence of Generative AI.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Oldnall, Chris, William Kay, Chris Sutherland, Tiago A. Marques, and Robert Young. 2026. “Overcoming Coding Anxiety: Lowering the Stakes and Making It Fun.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Orzechowski, Pawel, Brittany Blankinship, and Kasia Banas. 2026. “Where Do i Even Start with Pair Programming in My Classroom? A Conversation with Seasoned Practitioners.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Orzechowski, Pawel, Elaine Mowat, Karim Rivera Lares, Rebecca Sewell, Clare Llewellyn-MacRae, Bea Alex, and Kasia Banas. 2026. “3 Stars and 1 Wish: Small and Frequent Student Reflections Promote a Sense of Wonder and a Community of Vulnerability.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Riviera, Leo, Maeve Murphy Quinlan, Rebecca Colquhoun, and Devanjan Bhattacharya. 2026. “How Teaching Programming Across Disciplines Can Instil Systems Thinking.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Wadsworth, Lizzie, Lucia Michielin, Chris Cooling, Ozan Evkaya, Sam Skipsey, and David Cutting. 2026. “Computer Says No: Reducing Technical Barriers to Help Novice Programmers.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Wilson, John. 2026. “AI, Voice, and Style in Programming Education.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Wilson, John, and Brittany Blankinship. 2026. “It Depends: How to Develop Judgement in Programming.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Young, Robert S., Rebecca L. Colquhoun, Tiago A. Marques, Brittany Blankinship, Ozan Evkaya, and William P. Kay. 2026. “Sequential Vs. Simultaneous: Approaches to Learning Programming and Statistics.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.