Part III — Assessment and Feedback: Principles and Practice

Umberto Noè

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

Franziska McManus

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

Brittany Blankinship

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

Assessment is, for many students, the most anxious part of their studies. For many educators, it is the most time-consuming. For institutions, it is one of the most consequential as assessment shapes what students prioritise, how they feel about their learning, and whether they can progress. This part of the book asks: what does good assessment and feedback look like in programming education, and how do we actually build it?

The chapters here approach these questions from two angles. The first theme looks at the *frameworks* we use to think about assessment and feedback, i.e. the underlying schools of thought and structures that shape what we measure and how we respond to student work. The second theme turns to how to do this *practically* by looking at how do educators design, deliver, and scale assessment in programming courses, particularly as generative AI and automated tools are changing the landscape rapidly.

Theme — Frameworks for Assessment and Feedback

Developing a Skills-Based Framework for Assessments (Pankratz 2026)

What if, instead of averaging marks across a course, students simply had to demonstrate each skill and could keep trying until they did? That is the core idea behind the skills-based assessment framework discussed in this chapter, and the author makes a compelling case for it in the context of teaching programming to novices. The author, a lecturer in Psychology at the University of Edinburgh teaching statistics using R, argues that traditional grading harms students who are already anxious, behind the pace, or facing structural disadvantages. The skills-based framework (where students earn skills permanently, are not penalised for re-attempts, and receive feedback on every attempt) shifts the dynamic entirely. The chapter includes a detailed account of how the author's team is redesigning their first-year statistics course around this framework.

This chapter is for you if you teach programming to students who did not come for the programming (e.g., psychologists, social scientists, engineers), and want a principled alternative to marks-based grading. It is also an interesting reading if you care about equity in your classroom as the chapter examines the disadvantages faced by many students and argues that the flexibility of skills-based assessment is not just to be considered pedagogically but also ethically.

The Action Feedback Protocol (MacLaren et al. 2026)

Feedback is only useful if students act on it. That sounds obvious, but many feedback systems are not designed with that goal in mind, they are often designed to justify marks. The Action Feedback Protocol (AFP), introduced in this chapter, is a deliberate attempt to

change that. Developed at Heriot-Watt University and now in use across disciplines and campuses worldwide, the AFP works on three levels: it trains students to receive feedback well ("Tune the Ear"), gives markers a simple and consistent structure for writing useful comments ("Simplify the Message"), and prompts students to do something with what they have read ("Encourage Action"). The chapter is written by the team who built the protocol, including a former student who was in one of the first cohorts to use it.

This chapter is for you if you have ever written detailed feedback that students ignored, or struggled to get a marking team to produce consistent comments. The AFP is refreshingly practical. It is not a theory of feedback but a working protocol, and the chapter shows how it has been implemented in a way that you can adapt to your own course.

Theme — Practical Methods for Assessing Programming

Assessment and Marking in an Introductory Programming Course for Physics Students (Waugh, Chislett, and Dash 2026)

What does a thoughtfully designed sequence of assessments actually look like, in full detail, for an introductory programming course? This chapter provides exactly that: an honest and detailed account of how assessment is implemented in a first-year computing course for physics undergraduates at UCL. The authors describe their full assessment framework (i.e., weekly programming tasks marked in person, online multiple-choice quizzes used as a pre-requisite for submission, take-home assignments, and a final computational physics project), and explain the reasoning behind each choice. They are honest about what has worked, what has not, and how the course has evolved year by year. The in-person marking model is particularly unique as it allows for immediate two-way feedback and, increasingly, offers a practical response to integrity concerns in an age of generative AI.

This chapter is for you if you are designing or revising an introductory programming course and want a detailed model to learn from. The authors do not claim to have found the perfect system. They share a system that has been tested, adapted, and improved. You will find practical insights on scaling feedback, managing marking workloads, and balancing formative and summative assessment in a course where programming is one component among several.

Generative AI in Assessments (Clift and Petrovska 2026)

Generative AI has changed what assessment design, marking, and feedback can look like, and this chapter is a practical guide to using it better. The authors walk through the full lifecycle of an assessment:

- using large language models (LLMs) to draft multiple-choice questions, write assignment briefs, build marking rubrics, and generate model solutions
- supporting students as an exploratory tool during the work itself
- and using AI to speed up the writing of personalised feedback at the end.

The authors' approach is LLM-agnostic and committed to human oversight, making it clear that the chapter sees GenAI as a tool for educators, not a replacement for their expertise.

This chapter is for you if you are curious about how AI can reduce the workload of assessment design and feedback without compromising quality or fairness. Even if you are sceptical about AI in teaching, the chapter's structured, step-by-step prompting examples make it easy to experiment. It is especially relevant for large courses where creating varied assessments and providing timely feedback at scale is a challenge.

Using Automated Marking in Programming Courses to Enhance Learning (Shafti and El Gemayel 2026)

Automated marking has been transforming programming education for years, and this chapter is a practical introduction to what it looks like in practice. Focusing on CodeRunner (a widely used Moodle plugin that runs student code against predefined test cases and returns instant feedback), the authors explain how to set up automated assessments, configure test cases, and customise the system for different languages and course contexts. The chapter draws on experience from the University of Strathclyde, where automated marking has been used across Computer Science and non-Computer Science courses at multiple undergraduate levels, and acknowledges both the benefits and the learning curve involved.

This chapter is for you if you are considering automated marking for the first time and want to hear the reality of implementing it, rather than a sales pitch. It is also useful if you already use automated tools and want to extend what you do, as the chapter covers advanced configurations and points to further resources. If your students are not computer scientists and have little prior experience of programming, the setting described by the authors will feel particularly familiar.

Happy reading!

References

Clift, Lee, and Olga Petrovska. 2026. “Generative AI in Assessments: Before, During, and After.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

MacLaren, Andrew, Frederick Madsen, Antonia Voigt, Alex Buckley, and Tom Farrington. 2026. “Managing the Rubik’s Cube of Assessment: The Action Feedback Protocol.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Pankratz, Elizabeth. 2026. “Developing a Skills-Based Framework for Assessments in Programming Courses.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Shafti, Leila S., and Joseph El Gemayel. 2026. “Using Automated Marking in Programming Courses to Enhance Learning.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.

Waugh, Ben, Rebecca Chislett, and Louise Dash. 2026. “Assessment and Marking in an Introductory Programming Course for Physics Students.” In *Teaching Programming Across Disciplines*. Edinburgh Diamond.