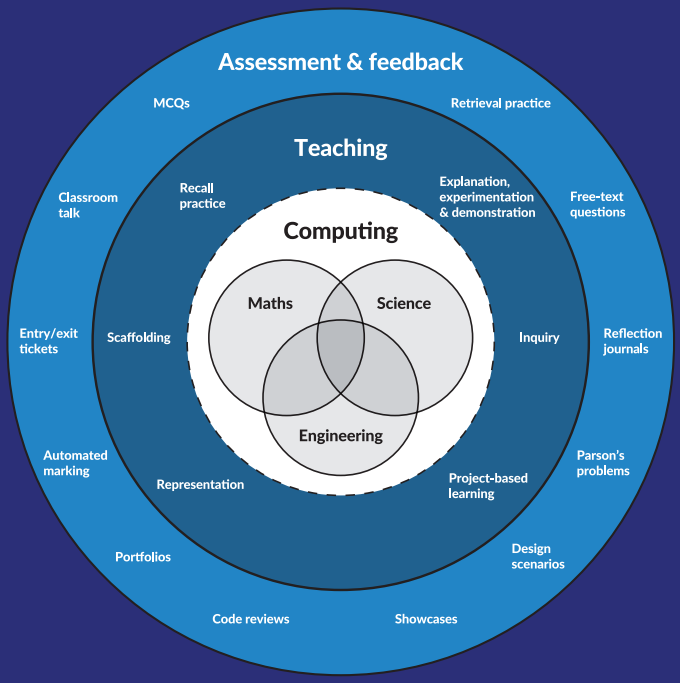
Raspberry Pi Foundation

Pedagogy Quick Read

Variety in teaching and assessment of programming activities

Computing is a broad discipline with deep roots in engineering, mathematics, and science,[1] as well as applications and connections to many other subjects. Appreciating these varied perspectives and the variety of teaching approaches and assessment strategies available allows educators to tailor their teaching to suit their learners' experiences, needs, and the subject matter in question.



Summary

Perspectives on computing:

Computing is a broad discipline rooted in three main traditions:

- Computing as **engineering** is concerned with design and development
- **Maths** is integral to computing systems, software, and how we describe them
- Most **scientific** fields apply computing to model and explore the physical world

Classroom strategies:

A holistic approach to teaching computing reflects each of these perspectives, their priorities, and practices:

- An **engineering** perspective leads to more project-based learning with scaffolding to support individual learners
- A **maths** perspective focuses on acquisition and construction of knowledge, using representations to explore abstract concepts, and regular recall and practice of facts and processes
- A **science** perspective leads to a more inquiry-based approach where explanations, demonstrations, and practical activities develop learners' understanding and they predict and experiment

Computing as a broad discipline

According to the work of Tedre,[1] computing is a broad discipline built principally on three traditions, each bringing its own perspectives. Those involved in the field of computing tend to see it as being concerned with either engineering and design, as a branch of mathematics and logic, or as a science. Each tradition has a different focus, prioritises different knowledge and skills, and invites different teaching approaches. However, all form part of computing as a whole:

- Computing is engineering: This view prioritises the design and development of artefacts including software and systems. It incorporates user research, prototyping, testing, and evaluation.
- Computing is maths: Logic and mathematics are present throughout computing. Our software and systems are built on mathematical principles and we use mathematical techniques to describe and reason about programs.
- Computing is science: Computing is pervasive across almost every field of science. We use computers to explore and model the physical world, and to make predictions and discoveries.

Beyond these three traditions, computing is connected to other areas such as the arts, where computing is applied as a medium, or philosophy and ethics where the application of computing provides rich materials for discussion.

Depending on our experience, we're each likely to favour one or more of these perspectives, which may impact how we present computing to our learners. By understanding these traditions and the wider connections, educators can provide their learners with a complete and holistic experience of computing. This enables them to provide a variety of meaningful entry points to the discipline supported by appropriate pedagogy.

Expanding your strategies toolkit

To expand the range of strategies you are able to employ in the classroom, reflect on your perspective of computing. Does your perspective impact the approaches you favour? What new practices could you try that could increase entry points for your students and enhance their experience? How else could you capture and assess your learners' understanding?

Variety within teaching approaches

Whether during a single lesson or an entire course, computing educators need to be able to apply a variety of pedagogical strategies. These will vary depending on the subject matter, the learners, and the aims of each learning experience.

Maths concerns understanding, applying, and connecting abstract concepts. The same is true for areas of computing with links to maths, where learners need to understand abstract concepts, recall facts, and practise calculations and processes. In these situations, educators adopt approaches that focus on the acquisition and construction of understanding. For example:

- Representation is a key part of a mastery approach to maths, which uses different modes of representation, including physical objects, pictorial representations, and eventually symbols and language. This approach may be a successful way to teach learners about binary number systems, for example.
- Varied and regular recall of concepts and processes is used to secure existing understanding, challenge misconceptions, and form connections and develop a coherent understanding.

Engineering concerns making an artefact that solves a problem or addresses a user need and can link computing to other areas of the curriculum. In computing, an artefact could be a program, system, or digital media. Physical computing, in particular, is an obvious way to learn about computing through an engineering lens. For example:

- Project-based learning is closely associated with this perspective. Learners apply their prior knowledge to a problem, focusing on one or more aspects of the design process.
- Specifically when programming, you can adjust the scaffolding and support you provide to learners depending on their needs and your focus.[2]

A science-based view of computing involves more inquiry-based practices where understanding is constructed through prediction, exploration, and observation. Simulations, practical demonstrations, and experiments are also used to develop skills and understanding.[3] For example:

- Learners develop their inquiry skills when programming with the PRIMM methodology; they predict and validate their predictions, as well as investigating and asking questions of the code.
- Topics like computer systems or networks[4] contain plenty of substantive ideas or facts that can be explored through a combination of explanation, demonstration, or experimentation. Educators have to select the best balance of approaches to suit each new concept and their learners' needs.

Another lens through which to understand computing is its role in society and the ethical and personal implications of the use of technology. Offering learners the opportunity to discuss ideas and engage in meaningful classroom talk, whether with the teacher or their peers, can support a rich understanding of concepts.[5] Discussion and debate are particularly relevant to computing as the reach and impact of technology is fertile ground for legal, moral, and ethical discussions. The social and cultural connections educators draw upon have an impact on how learners engage with a topic. Rooting your practice in the lived experiences, cultural knowledge, and background of your students makes their learning more relevant and accessible.

Variety within assessment

There is still much work needed to create reliable assessment approaches for all teachers and for all students. However, there is general agreement amongst researchers that using a variety of assessment approaches helps give teachers a much better picture — a holistic view — of student progress.

Classroom talk is an important assessment tool, providing teachers with an opportunity to assess student understanding in depth and provide feedback. Using design scenarios where students can discuss and adapt example programs highlights their skills as well as knowledge. Code reviews and showcases for students to talk about their work can provide peer and teacher assessment opportunities.[6]

Incorporating assessment activities into lessons embeds assessment. For example, portfolio creation and analysis or reflection journals, which require students to answer key questions during project development, encourage continuous self-assessment. Using entry and exit tickets, where students quickly record knowledge or confidence about current learning topics before and after lessons, can be a quick and regular assessment approach.

More traditional assessment tasks, such as multiple-choice questions (MCQs) and free-text questions, provide formative and summative assessment opportunities. Although effective MCQs can be challenging to create and offer limited feedback, they can be a quick and low effort way to discover student understanding.

References

1. Papert, Tedre, M., (2014). *The science of computing: shaping a discipline*. CRC Press.

2. Waite, J., Liebe, C., (2021, March). Computer Science Student-Centered Instructional Continuum. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, online (p. 1246). Association for Computing Machinery.

3. GOV.UK (2021). *Research review series: science*. [online] Available at: <https://www.gov.uk/government/publications/research-review-series-science/research-review-series-science#practical-work> [Accessed 23 August 2021].

4. National Centre for Computing Education (2021). *Computer Systems and Networking Within the Computing Curriculum. Teaching and Learning Reports*. [online] Available at: <https://blog.teachcomputing.org/computer-systems-and-networking-within-the-computing-curriculum/> [Accessed 24 August 2021].

5. Sentance, S., Waite, J. (2021, August). Teachers' Perspectives on Talk in the Programming Classroom: Language as a Mediator. In *Proceedings of the 17th ACM Conference on International Computing Education Research*, online (pp. 266–280). Association for Computing Machinery.

6. Grover, S., Sedgwick, V., Powers, K. (2020). Feedback through formative check-ins. In S. Grover (Ed.), *Computer Science in K-12: An A to Z Handbook on Teaching Programming*. Edfinity.

Raspberry Pi Foundation