An Optimistic Outlook on Teaching, Learning and Assessment for Coding With the Emergence of Generative AI

Steven Watterson

Personalised Medicine Centre, School of Medicine, Ulster University
s.watterson@ulster.ac.uk

Laila Dabab Nahas

Edinburgh Medical School, University of Edinburgh
ldababn@ed.ac.uk

Tuğrulcan Elmas

School of Informatics, University of Edinburgh
tugrulcan@ed.ac.uk

Hebatallah Shoukry

School of Engineering and Physical Sciences, Heriot-Watt University
h.shoukry@hw.ac.uk

Ozan Evkaya

School of Mathematics, University of Edinburgh
Ozan.Evkaya@ed.ac.uk

Introduction

The introduction of generative AI (GenAI) has transformed how staff and students view teaching, learning and assessment. Whilst education is a diverse sector, there has been universal recognition that GenAI has the potential to derail teaching and learning activities (Bente, Randall, and Wäckerle 2024; Brennan and McDermott 2023). The traditional paradigm of communicating ideas, tutoring understanding and assessing learning has become harder to safeguard as authentic when GenAI tools can synthesize text rapidly and easily to a level consistent with passable understanding. Were GenAI tools highly exclusive with limited availability then that risk might be modest, but they have been made widely and freely available, by technology companies that students already trust, as those companies look to build their user bases.

In light of the impact that GenAI will have on teaching, learning and assessment and recognising that it is likely to become part of the graduate's tool set, it is worth considering how we can adapt teaching both to support learners to develop their skills with GenAI tools, but also to defend the authenticity of learning and assessment as far as possible. Here, we consider strategies that can be followed to incorporate GenAI into teaching and to adapt assessment to mitigate the risk from GenAI-based fraudulent completion. To understand the rationale for these strategies, it is important to have some understanding of how GenAI tools work.

How Does Generative AI Work?

GenAI takes a mathematical approach to describing language. Sentences can be broken up into words, partial words, and elements of punctuation. These components are known as tokens and the combination of tokens that make up a line of text is mapped to a data point on a set of axes (in a high number of dimensions). Some processing is applied when creating the data point that weights the relative importance and order of the tokens in the text. Similar text will be mapped to nearby data points, whereas very different text will be mapped to data points that lie far apart (Hua and Yao 2024). The words, partial words, and punctuation that the GenAI recognises as tokens are its vocabulary and the data point describing the text is fed into a neural network. Then calculates the probabilities of each token next following the text, where the neural network itself has been trained with large volumes of literature[44] (Jiang et al. 2019). The GenAI algorithm doesn't always pick the highest probability next token, but one of the most probable and how much it is allowed to deviate from the most probable choice is a model parameter that ensures some variability in the responses generated. Once the next token has been selected, it is added to the text, a new data point is calculated and this is fed into the neural network, with the process repeating until a suitable volume of text has been generated or a stop signal is reached (Ponmalar et al. 2025).

It is worth emphasizing that the text generated is a statistical construction based on the training literature, not a series of quotes. This means that factually correct sentence structures cannot be guaranteed and are only likely to be generated when they are common in the training literature. Factually correct sentences about

[44] https://hatchworks.com/blog/gen-ai/large-language-models-guide/

uncommon topics are harder to produce consistently (Jiang et al. 2019).

This system is known as a Large Language Model (LLM) and it forms the basis of most GenAIs. One quirk of LLMs is that they are trained to consider the input text, which is formed from the prompt, as factual and to reference it with little statistical reinterpretation. Furthermore, prompts can be extremely lengthy (Gan and Mori 2023). We can combine these two properties of LLMs to improve the factual accuracy of the generated text by including factual source materials as part of the prompt. This is usually done in a process called Retrieval Augmented Generation (RAG) and it is a way to dramatically improve the factual recall of the generated text. By using RAG, the LLM can be formulated so that it generates narrative text fluently based on its training data, but draws on factual details from source materials through RAG (see Figure 9.1). It yields a system that can be both conversational and factually accurate (Miao et al. 2024).

The Current State of GenAI in Learning and Teaching

We are still in the early stages of exploring the implications for GenAI in teaching and learning. The literature reveals contradictions in how GenAI tools are employed in education. From one perspective, GenAI offers promising benefits when integrated into educational settings under the guidance of educators using factual frameworks such as RAG. Some systematic studies have shown that GenAI can facilitate student learning, enhancing cognitive, and technical and interpersonal skills, and students who used GenAI performed better than those who did not (Daniel, Msambwa, and Wen 2025; Heung and Chiu 2025). Additionally, students using GenAI outperform those who do not in areas such as learning performance, AI awareness, and managing cognitive load. However, students who do not use GenAI tools demonstrate better critical thinking (Ji et al. 2025). Whilst critical thinking is ordinarily superior amongst non-GenAI-using student groups (Ji et al. 2025), when GenAI is used appropriately improvements in critical thinking have been shown (Daniel, Msambwa, and Wen 2025). It has been recommended as best practice to encourage students to first attempt tasks independently before consulting GenAI tools (Kosmyna et al. 2025).

From the other perspective, questions remain about how to deal with the unguided use of these tools, which can lead to over-reliance and missed learning opportunities for students who over-delegate to GenAI tools (Kosmyna et al. 2025; Pearson 2025). This phenomenon, known as “cognitive offloading”, occurs when individuals choose not to commit details to memory or invest effort in learning skills. Instead, they relying heavily on GenAI tools to retrieve data and complete tasks for them (Pearson 2025; Fan et al. 2024). These challenges are not new. With the advent of search engines, it was documented (Pearson 2025) that there was a tendency to memorise less information, instead counting on search engines to facilitate the retrieval of web-based data as a means of recall. As GenAI grows in quality and capability, there is a risk that cognitive offloading may go beyond recall and result in entrusting entire tasks to AI irresponsibly (Fan et al. 2024). Hallucinations and the biases gained in training GenAI tools both present the risk of unintentionally adopting incorrect information and there is concern over the influence of the companies providing the GenAI tools, where accidental or deliberate training biases could affect public usage (Pearson 2025).

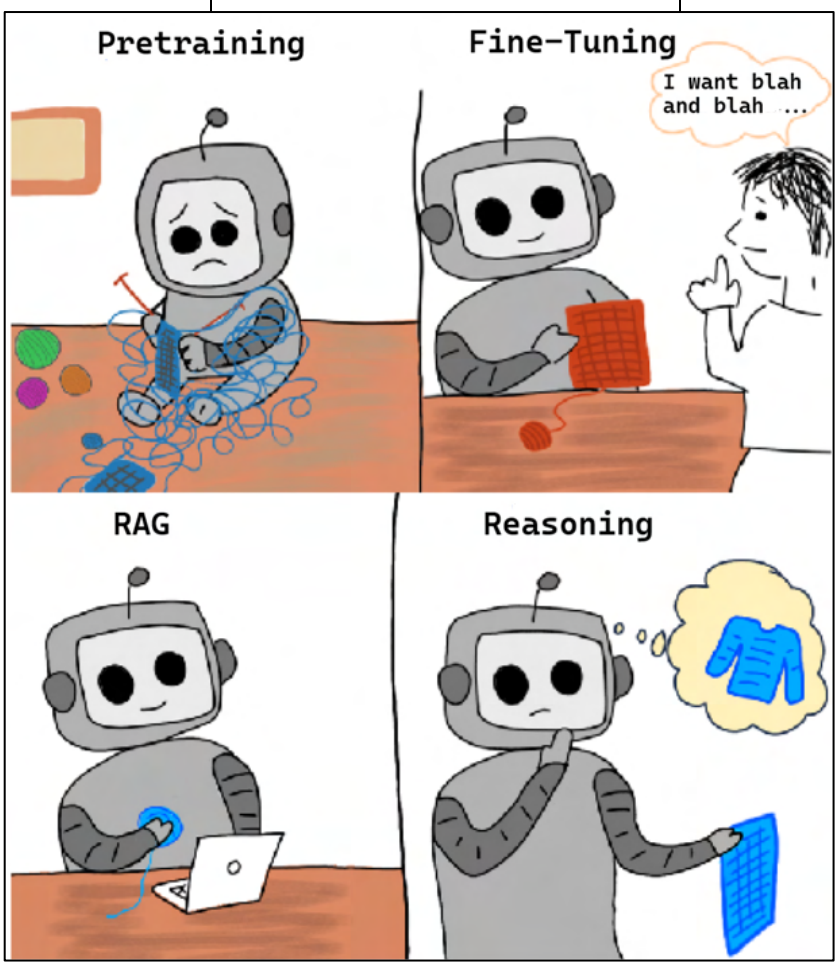

Figure 9.1:

Top:

LLMs development through a two-step process: pretraining & finetuning with question answering in finetuning where the user asking GenAI to do something (e.g., in this example a metaphor for a user asking the GenAI to create a knitted orange square from wool and GenAI did it).

Bottom: enhancement of LLMs through RAG & Reasoning.

The key differentiator in outcomes is how GenAI is employed. Without proper guidance students can over-rely on GenAI tools, substituting the generated output for their own ideas. It is essential to equip students with strategies to harness GenAI's potential whilst avoiding its pitfalls (Boers et al. 2025). Many studies have emphasized the need for longitudinal research to understand the long-term effects of GenAI on recall and cognitive skills, but in the absence of longitudinal research, we must continue to innovate, develop and adapt the guidance and best practices that will support students effectively in integrating GenAI into their academic and professional lives.

GenAI as a Teaching and Learning Partner

GenAI has enormous potential to support students as a teaching and learning partner, offering detailed explanations, engaging in discussions, and interactively responding to queries 24 hours a day, 7 days a week. This makes it a valuable tool that can fulfill a wide range of roles.

GenAI As Topic Expert

By using Retrieval-Augmented Generation (RAG) to integrate a factual corpus of knowledge into GenAI responses, it becomes possible to consider employing GenAI tools as expert systems that can support both staff to teach and students to learn. As a teaching assistant, GenAI has the potential to support staff to create more learning materials across more learning modalities than is currently possible in limited time, which can enhance support for students with different learning preferences (Raju 2025). GenAI has the further potential to act as the first receiver for student questions and queries. By increasing the productivity of teaching staff and by resolving the more straightforward queries, GenAI tools have the potential to help teaching staff to increase the time committed to key pedagogical task (Nyaaba et al. 2024).

The proximity of data points in vector space can represent related ideas. Due to that, GenAIs can connect related ideas well and are adept at providing alternative explanations (as a result of their inherited randomness). GenAIs, therefore, can perform well in tutoring roles and have been well received in studies of student adoption (Kanont et al. 2024; Bouzid and Piron 2024; Watterson et al. 2025). GenAI also has the potential to directly benefit the student by acting as a topic expert available 24/7 that can interact with the student and expand on the areas of greatest need, providing truly personalised support. With access to course documentation, it can also help with management of deadlines and timetables, etc. Examples 1 and 2 below demonstrate use of Google Gemini 2.5 Flash and NotebookLM on student queries, with additional information provided by RAG.

Example 1: Administrative Information Provided by RAG to a LLM

Input:

Module Handbook

- Degree Programme: Programming exercises for coders.
- Module Code: 1234.
- Office Hours: Tuesday and Thursday 2pm-4pm.
- Module Coordinator: Hamish McBeth.
- Room Number: 12.
- Course Duration: 12 weeks.
- Assessments: Examinations are in weeks 6 and 12. Coursework deadlines are at the end of week 5 and the end of week 11.

Prompt: Who is the coordinator and where and when can I find them?

- Response: The Module Coordinator is Hamish McBeth. You can find him in Room Number 12 during his office hours, which are Tuesday and Thursday from 2pm to 4pm.

Example 2: Code Provided by RAG to a LLM

Input:

```
import numpy as np, pandas as pd
from sklearn.datasets import
load_iris
from sklearn.decomposition import
PCA

iris = load_iris()
pca = PCA(n_components=2)
principalComponents =
pca.fit_transform(iris.data)

print(f"Explained variance by PC1
is ",
pca.explained_variance_ratio_[0],
f"Explained variance by PC2 is ",
pca.explained_variance_ratio_[1])
```

Prompt: Concisely, what does this code do and what does it print out?

- Response: The code performs Principal Component Analysis (PCA) on the Iris dataset, reducing its dimensionality to two principal components. It then prints the explained variance ratio for PC1 and the explained variance ratio for PC2.

AI as a Coding Partner

An effective paradigm used to teach coding is that of paired programming where two coders collaborate on a single script. Each takes a separate role, with one acting as the primary code-writer (driver) and the other acting to spot errors, provide structural feedback and manage the resources needed (navigator). By separating the coding activity into writing and reflective roles, paired programming can improve the efficiency of code delivery; by alternating roles, both partners have the opportunity to learn as they work.

Where a coder does not have a coding partner, a GenAI can take on the role. As a writing partner, GenAI tools can create code for a precise prompt. GenAI tools are more accurate and less prone to syntactical errors than humans (i.e., they are less likely to forget a comma or bracket or miss a function specific parameters), but they can lack sufficient context to ensure that the code meets every explicit and implicit requirement of a brief or that the blocks of code generated independently can integrate adequately. These errors can be addressed by the coder. As a reflective partner, GenAI tools can evaluate code, recognise structural errors and suggest refinements to a specific remit to support the coder; see Example 3 below. In both cases, the quality of the role-play will be very sensitive to the prompt and so some trial and error may be required. It may be advantageous to separately specify roles in system-level prompts, which are commonly used to frame the subsequent dialogues.

Example 3: Prompted Critique for the code in Example 2 (Gemini 2.5 Flash/NotebookLM)

Prompt: Please concisely critique this code.

Response: This code snippet from "Iris Dataset PCA Explained Variance" effectively demonstrates the application of Principal Component Analysis (PCA) to the Iris dataset. Here's a concise critique:

Positive Aspects:

- Clarity and Purpose: The code is straightforward, clearly showing how to load data, initialise PCA with two components, and then fit and transform the data to calculate the explained variance ratios.
- Standard Libraries: It uses widely accepted libraries such as sklearn.datasets, sklearn.decomposition, and numpy for data loading, PCA, and numerical operations, respectively.
- Direct Output: The print statement effectively displays the explained variance for the first two principal components (PC1 and PC2).

Areas for Improvement:

- Unused Imports and Variables: The pandas library is imported (import pandas as pd) but not utilised in the provided code snippet. Similarly, the `principalComponents` variable, which stores the transformed data, is created but not used for any subsequent operations within this specific excerpt.
- Scope: While effective for its stated purpose, the snippet could be extended with visualisations (e.g., a scree plot) or further analysis to demonstrate the practical application of the principal components for dimensionality reduction.

A number of Integrated Development Environments (IDEs) already include Ais as a coding partner and this approach is regularly used in industry. Developer surveys and interviews have recorded improvements in code quality and developer satisfaction, but with questions over code reliability, explainability, and trust (Bughin 2024; Chen 2024). Some studies have shown that paired programming with a GenAI tool delivers the best assignment outcomes for students compared to human paired programming or solo programming (Bughin 2024; Lyu et al. 2025). However, other studies have suggested that the improvements in the speed with which code is created are offset by the additional time needed to debug the code (Imai 2022), and that concerns about sensitive parts of code bases mean that GenAI tools cannot support projects consistently, only in a piecemeal manner (Coutinho et al. 2024).

AI as a Puzzle Setter

GenAIs can be prompted to generate coding questions along with worked solutions. This can be invaluable as a way to efficiently produce formative exercises and it becomes very powerful when pre-existing scripts or code-bases are included by RAG to use as a reference document (Pourcel et al. 2024). Typical exercises can include (i) generating incomplete blocks of code for students to complete, (ii) Parsons puzzles where students rearrange instructions into the correct sequence, or (iii) asking students to interpret the behaviour of a generated block of code (see Example 4) (Pourcel et al. 2024; Hou et al. 2024). This approach allows formative coding exercises to be generated with relative ease that align directly to topics of study, such as a particular algorithm or data set.

Example 4: Using the GenAI Tools To Create Coding Puzzles From the Script Provided by RAG in Example 2 (Gemini 2.5 Flash/NotebookLM)

Prompt: Produce a version of this code with key words missing that can be used as a teaching and learning exercise.

Response:

```
import numpy as np, ____ as pd}
from sklearn.datasets import ____
from sklearn.decomposition import
____

iris = load_iris()
pca = PCA(n_components=____)
principalComponents =
pca.____(iris.data)

print(f"Explained variance by PC1
is ", pca.____, f"Explained
variance by PC2 is ", pca.____)
```

Prompt: Please rearrange this code in a parsons puzzle.

Response: Parsons Puzzle: Reconstruct the PCA Code — Please arrange these lines of code in the correct sequence to perform PCA on the Iris dataset and print the explained variance ratio. Pay attention to the order and any necessary indentation.

```
1. from sklearn.datasets import
load_iris
2. pca = PCA(n_components=2)
3. print(f"Explained variance by
PC1 is ",
pca.explained_variance_ratio_,
f"Explained variance by PC2 is ",
pca.explained_variance_ratio_[1])
4. principalComponents =
pca.fit_transform(iris.data)
5. import numpy as np, pandas as
pd
6. iris = load_iris()
7. from sklearn.decomposition
import PCA
```

How to Solve: Your task is to place these lines in the logical order of execution for the script. The original script begins with imports, then loads data, initializes the PCA model, performs the transformation, and finally prints the results

GenAI as a Coding Tool for the Student

As the use of GenAI tools to create code grows, the role of the coder will necessarily change. The coder will no longer directly produce code, but instead supervise the tools generating the code. Nonetheless, the requirement to stand over the code and take responsibility for its correctness and accuracy will remain. This can be achieved either (i) by forensically working through the code produced by the GenAI tools or (ii) modularising the challenge, having the GenAI tools separately produce code for each module and then evaluating the performance of each module's code separately and in combination. The latter is a form of unit testing; a process widely used in industry to manage complex projects across coding teams. Here, it transforms the student into a code manager (Trautsch and Grabowski 2017; Osherove 2009). This way of working is both likely to scale better as projects become increasingly sophisticated, but also has the benefit of starting students to think and work in ways that are closer to software roles in industry (Vorel 2025; Bhatia et al. 2023). In addition, it is very important to notice that a code manager is not for first or second years students but for more senior students after fully understanding and acquiring the foundational skills to spot mistakes.

A unit testing approach means introducing a formal design stage where projects are broken into constituent modules/units with clear specifications (Osherove 2009; Northwood 2018) in which written modules/units are (i) configured with a state and data on which to run (Arrange), (ii) run on the data and state (Act) and (iii) evaluated on the correctness of the output (Assert). A Test Runner is then constructed to evaluate a range of inputs and outputs for each module.

A basic unit testing framework evaluates the behaviour of the code, not the code itself, which is well suited to contexts where coding can be entirely devolved to GenAI tools. Where the use of particular commands or data structures is a requirement, the tests must be adapted to reflect this. Ensuring that the tests are rigorous and appropriate will become a new skill and responsibility for the student.

Unit testing is not only a tool for students to use in code development, but also a tool for staff to use when assessing student submissions and significant work has been undertaken evaluating how well GenAI tools can create unit tests from project briefs (Bhatia et al. 2023; Mock, Melegati, and Russo 2024). There, GenAI such as ChatGPT, Copilot and Gemini were studied in a variety of programming languages such as Python and Java. It was found that such tools perform well, especially in terms of repetitive tests. However, the literature stresses the necessity of human oversight for correctness because of its weaknesses, such as incorrect or hallucinated assertions and weak reasoning about program logic so it requires significant human review.

Assessing Learning

Assessment should always evaluate whether learning has taken place, but, depending on the format, there is significant potential for GenAI to be used fraudulently. The ease of use and high availability of GenAI tools means that conventional take-home, open book, step-wise assignments are highly vulnerable to completion by GenAI tools.

Whilst it is impossible to devise assessments that are invulnerable to GenAI tools, especially when they must remain accessible to students of a wide range of levels of academic ability, there are strategies we can follow that increase the difficulty and therefore reduce the likelihood of students using GenAI in their submissions. Across the range of academic abilities, the most capable students will feel that they have

the least to gain and the most to lose from fraudulently using GenAI. At the other extreme, the least academically able students will feel that they have the most to gain and the least to lose. However, the least able students are also often those who are the least invested in their learning. Our guiding principle to making assessments resistant to GenAIs, is that we must design assignments so that the workload required to complete it by GenAI, ideally exceeds the workload of making an honest submission, but at least exceeds the level of work that less capable students are prepared to commit. Nonetheless, a motivated and capable student will always be able to engineer prompts that support the completion of an assignment. Thus, each of those techniques comes with challenges such as the better start-bias, where students start learning at different levels which we need to take into account and appreciate when developing these strategies. We will consider several strategies here and for a particular exercise, a combination of strategies may be optimal.

Supervision of Assessment

Amongst the simplest ways to maintain the authenticity of assessment is to enforce strict supervision. This is a time- and labour-consuming approach and is not necessarily an efficient use of resources, especially for longer and more complex assignments.

A more time-efficient version of this approach involves replacing complex assignments that evaluate the coordinated use of multiple learning outcomes with a series of shorter competency-based assessments each testing a single atomised component of the learning outcomes. This might be a series of tests in which a student is asked to demonstrate core elemental coding skills whilst being observed one-to-one by the assessor. Each assessment for each student may require only a short time, with a binary pass/fail outcome, and progression from the programme then becomes dependent on a portfolio of demonstrated competencies.

Insufficiency

We can frustrate GenAI by designing assignments with briefs that are individually incomplete and require knowledge from multiple sources to be made complete. Such an assessment must be designed so that a student must bring understanding gained elsewhere to the assessment to fully understand the brief. For a student to use GenAI tools successfully on the assessment, they must both piece together the missing information from the other sources and also formulate it in a way that complements the content of the brief and can be interpreted unambiguously by the GenAI tool. Whilst this will be attainable for the most capable students, returning to the principle described above, we only need to design an assessment so that compiling the brief and the missing information into successful prompts incurs a greater workload than a less capable student is prepared to commit.

Designing assignments that bring together learning from multiple sources, potentially across different modalities (e.g., using videos, written text and external resources), maximises the workload in completing the assignment brief and therefore the challenge of building a useful prompt for a GenAI tool. In practice, incompleteness can take the form of ambiguous phrasing, ambiguous technical detail, missing stages of a pipeline, or newly introduced terminology with resolutions and definitions embedded elsewhere in the programme (for example, having the general in text while more details are in another form such as a video). What inhibits a student from using GenAI tools successfully in this case is that they must not only piece together the missing information needed to complete the assessment brief, but also formulate it in a way that is useful and unambiguous for GenAI completion.

Finally, teachers must be aware that some risks could arise from such approaches for developing such assessments. For example, giving extra attention to material accessibility for some students with special needs and trying to avoid extra unnecessary pressure could be added to students as barriers, such as extra time or effort to put in order to complete the assessment.

Misdirection

We can construct assignments so that they are interpreted differently by students and by GenAI. The simplest example is where hidden text is placed in an assessment brief. Typically, this is text embedded in a paragraph at a small and unobtrusive font size, coloured to blend into the background. This hidden text will not be seen by the student who manually attempts the assessment brief, but when the brief is copied or imported into a GenAI tool, the hidden text is exposed and treated as a part of the brief. The hidden text can negate a statement, add digits to critical numbers or misdirect the GenAI and whilst it will be observed by a diligent student, it may go unobserved when less care is taken, especially in longer briefs where the hidden text can become harder to discern.

Critically, the hidden text should be chosen so as to lead to incorrect responses when the brief is imported or copied into a GenAI prompt without careful attention. This may help in making it more complicated for students who are using GenAI to obtain sufficient marks without extra care and true engagement with the assessment. We can not consider this an effective way for detecting GenAI misuse but we can make it harder for those who misuse GenAI to get undeserved accomplishments.

Reverse Engineering Assessments

The principal strength of GenAI tools lies in their ability to follow a sequence of steps and instructions. Assessments that move away from asking students to complete a prescribed sequence of steps and instead to explore permutations of options to identify an optimal solution are therefore likely to be inherently more challenging for GenAI tools to complete.

One such approach is to ask the student to reverse engineer an analysis or technique. Providing a student with a starting data set, the output from an analytical pipeline and an understanding of the possible steps

that could be combined to derive the output from the starting data set. It is possible to design assessments in which the student must experiment to identify the combination of steps that generate the output from the starting data. If this pipeline has been completely explained elsewhere, this becomes a specialised case of an insufficiency assessment where part of the instructions can be retrieved from elsewhere in the programme. However, if the sequence of steps have not been taught together, but only individually and with alternatives having been explored, the student will have to evaluate multiple permutations of options. Depending on the explicitness of the assessment brief, the number of steps that must be combined and the number of options available at each step, GenAI tools are unlikely to enumerate all possible options to identify the correct answer.

Unsighted Data Sets and Structures

GenAIs are exposed to large volumes of pre-existing scientific literature as part of their training. Assessments that are designed based on widely published data sets or data structures are likely to be easier for GenAI tools to complete than assessments based on rare or bespoke data sets and structures. GenAIs are sensitive to prompts and will attempt to create code for data sets and structures described in the prompt, but if the assessment data sets and structures are sufficiently uncommon then the differences between this and the sets and structures in the related training data may cause the GenAI tools to code imperfectly, with errors for the rare and bespoke examples. Students will then be challenged and need to demonstrate competency to debug these errors. An effective, rare source of data might be that held within databases as this is less likely to have been scraped as part of a training set.

This approach comes with the caveat that experience of working with standardised data sets and structures are often an important learning outcome and so directing students towards the rare and bespoke and away from standard data sets and structures may not be constructive or translational.

Student Self-Assessment

Whilst it is strategically important to frustrate the use of GenAI in assessment, at the same time, we must acknowledge that GenAI tools have a place supporting student learning and we need to encourage students to use AI constructively to prepare them for the workplace. When used effectively, AI can enhance productivity, ensure continuous skill development and provide a more secure future aligned with emerging technologies.

Asking students to outline the logical steps they planned and followed using AI tools as an assistant for completing tasks or assessments introduces a certain degree of transparency around GenAI engagement (Cotelli Kureth, Paliot, and Zink 2025). They should describe the steps taken to validate GenAI outputs, as well as how they addressed biases and disparities in the results. A self-assessment provides students with the additional opportunity to reflect on their experience of working with GenAI tools, identifying when GenAI use was most beneficial, when it was not or was unproductive and how it has improved their learning skills and helped them to achieve their goals (Combrinck and Loubser 2025). Checklists have already been introduced to students to disclose and reflect on their use of GenAI either as part of their assessment or as general guidance. We recommend the example provided by the University of Leeds and published in their generative AI guidance pages[45].

The Digital Divergence

The introduction of any new technology can create inequalities between those who are skilled and able to exploit the technology and those who aren't. However, conventionally, new technologies have been peripheral to the main tasks of coding and so have only offered incremental benefits to productivity. The great potential for GenAI tools lies in their ability to directly create code and therefore accelerate productivity. This is qualitatively different to productivity boosts seen before (e.g., before it used to be manually catching typos and errors, while now it could be easily highlighted and caught with different editors or tools supported by GenAI).

Skilled use of GenAI tools will not only affect the pace at which code is created, but also the scale, so that code can be created quickly and more easily in parallel. These developments will necessarily place an emphasis on code management and prompt engineering, which will have to be taught alongside the fundamentals of coding. These new paradigms raise the level at which a student will be expected to operate and create new standards of excellence for student learners. However, adding new and more layered skills to a teaching programme is likely to exacerbate the academic divide between weaker and stronger learners (Suárez and García-Mariñoso 2025; Rottner et al. 2025).

At the same time, because GenAIs have weakened the legitimacy of the simplest structures of assessment, not only is the syllabus likely to develop in sophistication, but so too will the structures and formats of assessment, potentially compounding confusion and further exacerbating the academic gap between weaker and stronger learners.

[45] https://generative-ai.leeds.ac.uk/ai-and-assessments/gen-ai-quick-checklist/

Conclusion

Here we have introduced GenAI as both an asset to teaching and learning and a threat to the authenticity of assessment. We have highlighted evidence that GenAI tools can improve productivity, but not necessarily across all the skills associated with coding and that there maybe risks associated with students becoming dependent on GenAI tools. In particular, we highlight the roles that GenAI tools can play that support teaching and learning, namely as a topic expert, coding partner, and puzzle setter. We also explore how to preserve authenticity in assessment in the age of GenAI, where educators must combine several deliberate strategies suggested in this chapter. Strengthened supervision can help ensure genuine student performance, while designing assignments that integrate learning from multiple sources and modalities — such as text, videos, and external materials — increases the cognitive demand and reduces the ease of GenAI misuse. Techniques such as embedding hidden text, using reverse-engineering-based tasks, and working with unsighted or bespoke datasets make it harder for GenAI tools to generate correct answers without genuine student engagement. Finally, incorporating structured student self-assessment encourages transparency, reflection, and responsible use of GenAI, supporting more authentic demonstration of learning.

Ultimately, we believe that the introduction of GenAI will transform coding so that the coder's role becomes that of a code manager. This may change the approach taken to the coding exercises in courses. However, these transformations are so central to how coding is taught, learned, and applied that not all students may be able to adapt equally well across all ability levels. Hence, there is a significant risk that their introduction may exacerbate the gap in abilities rather than close it.

References

Bente, Stefan, Natasha Randall, and Dennis Wäckerle. 2024. "A Conceptual Framework to Transform Coding Education in Times of Generative AI." In *Software Engineering Im Unterricht Der Hochschulen 2024*, 93–104. Gesellschaft für Informatik eV. https://doi.org/10.18420/seuh2024_08.

Bhatia, Shreya, Tarushi Gandhi, Dhruv Kumar, and Pankaj Jalote. 2023. "Unit Test Generation Using Generative AI: A Comparative Performance Analysis of Autogeneration Tools." In *Proceedings of the 2024 IEEE/ACM International Workshop on Large Language Models for Code (LLM4Code)*, 54–61. https://doi.org/10.1145/3643795.3648396.

Boers, Jelle, Terra Etty, Martine Baars, and Kim van Broekhoven. 2025. "Exploring Cognitive Strategies in Human-AI Interaction: ChatGPT's Role in Creative Tasks." *Journal of Creativity* 35 (1): 100095. https://doi.org/10.1016/j.yjoc.2025.100095.

Bouzid, Sara, and Loïs Piron. 2024. "Leveraging Generative AI in Short Document Indexing." *Electronics* 13 (17): 3563. https://doi.org/10.3390/electronics13173563.

Brennan, Robert W, and Brenda McDermott. 2023. "Coding Literacy in the Age of Generative Ai." In *International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*, 445–56. Springer. https://doi.org/10.1007/978-3-031-53445-4_37.

Bughin, Jacques. 2024. "The Role of Firm AI Capabilities in Generative AI-Pair Coding." *Journal of Decision Systems*, 1–22. https://doi.org/10.1080/12460125.2024.2428187.

Chen, Tianyi. 2024. "The Impact of Ai-Pair Programmers on Code Quality and Developer Satisfaction: Evidence from Timi Studio." In *Proceedings of the 2024 International Conference on Generative Artificial Intelligence and Information Security*, 201–5. ACM. https://doi.org/10.1145/3665348.3665383.

Combrinck, Celeste, and Nelé Loubser. 2025. "Student Self-Reflection as a Tool for Managing GenAI Use in Large Class Assessment." *Discover Education* 4 (1): 72. https://doi.org/10.1007/s44217-025-00461-2.

Cotelli Kureth, Sara, Elisabeth Paliot, and Suzana Zink. 2025. "Fostering Transparency: A Critical Introduction of Generative AI in Students' Assignments." *Language Learning in Higher Education* 15 (1): 63–85. https://doi.org/10.1515/cercles-2024-0105.

Coutinho, Mariana, Lorena Marques, Anderson Santos, Marcio Dahia, Cesar França, and Ronnie de Souza Santos. 2024. "The Role of Generative Ai in Software Development Productivity: A Pilot Case Study." In *Proceedings of the 1st ACM International Conference on AI-Powered Software*, 131–38. ACM. https://doi.org/10.1145/3664646.3664773.

Daniel, Kangwa, Msafiri Mgambi Msambwa, and Zhang Wen. 2025. "Can Generative AI Revolutionise Academic Skills Development in Higher Education? A Systematic Literature Review." *European Journal of Education* 60 (1): e70036. https://doi.org/10.1111/ejed.70036.

Fan, Yizhou, Luzhen Tang, Huixiao Le, Kejie Shen, Shufang Tan, Yueying Zhao, Yuan Shen, Xinyu Li, and Dragan Gašević. 2024. "Beware of Metacognitive Laziness: Effects of Generative Artificial Intelligence on Learning Motivation, Processes, and Performance." *British Journal of Educational Technology* 56 (2): 489–530. https://doi.org/10.1111/bjet.13544.

Gan, Chengguang, and Tatsunori Mori. 2023. "Sensitivity and Robustness of Large Language Models to Prompt Template in Japanese Text Classification Tasks." In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, 1–11.

Heung, Yuk Mui Elly, and Thomas KF Chiu. 2025. “How ChatGPT Impacts Student Engagement from a Systematic Review and Meta-Analysis Study.” *Computers and Education: Artificial Intelligence* 8: 100361. https://doi.org/10.1016/j.caeai.2025.100361.

Hou, Xinying, Zihan Wu, Xu Wang, and Barbara J Ericson. 2024. “Codetailor: Llm-Powered Personalized Parsons Puzzles for Engaging Support While Learning Programming.” In *Proceedings of the Eleventh ACM Conference on Learning Scale*, 51–62. ACM. https://doi.org/10.1145/3657604.3662032.

Hua, Haowei, and Co-Jiayu Yao. 2024. “Investigating Generative AI Models and Detection Techniques: Impacts of Tokenization and Dataset Size on Identification of AI-Generated Text.” *Frontiers in Artificial Intelligence* 7: 1469197. https://doi.org/10.3389/frai.2024.1469197.

Imai, Saki. 2022. “Is Github Copilot a Substitute for Human Pair-Programming? An Empirical Study.” In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, 319–21. IEEE. https://doi.org/10.1145/3510454.3522684.

Ji, Yu, Zehui Zhan, Tingting Li, Xuanxuan Zou, and Siyuan Lyu. 2025. “Human-Machine Co-Creation: The Effects of ChatGPT on Students’ Learning Performance, AI Awareness, Critical Thinking, and Cognitive Load in a STEM Course Towards Entrepreneurship.” *IEEE Transactions on Learning Technologies* 18: 402–15. https://doi.org/10.1109/TLT.2025.3554584.

Jiang, Zhengbao, Frank F Xu, Jun Araki, and Graham Neubig. 2019. “How Can We Know What Language Models Know?” *Transactions of the Association for Computational Linguistics* 8: 423–38. https://doi.org/10.1162/tacl_a_00324.

Kanont, Kraisila, Pawarit Pingmuang, Thewawuth Simasathien, Suchaya Wisnuwong, Benz Wiwatsiripong, Kanitta Poonpirome, Noawanit Songkram, and Jintavee Khlaisang. 2024. “Generative-AI, a Learning Assistant? Factors Influencing Higher-Ed Students’ Technology Acceptance.” *Electronic Journal of e-Learning* 22 (6): 18–33. https://doi.org/10.34190/ejel.22.6.3196.

Kosmyna, Nataliya, Eugene Hauptmann, Ye Tong Yuan, Jessica Situ, Xian-Hao Liao, Ashly Vivian Beresnitzky, Iris Braunstein, and Pattie Maes. 2025. “Your Brain on ChatGPT: Accumulation of Cognitive Debt When Using an AI Assistant for Essay Writing Task.” *arXiv.org* 4.

Lyu, Wenhan, Yimeng Wang, Yifan Sun, and Yixuan Zhang. 2025. “Will Your Next Pair Programming Partner Be Human? An Empirical Evaluation of Generative Ai as a Collaborative Teammate in a Semester-Long Classroom Setting.” In *Proceedings of the Twelfth ACM Conference on Learning Scale*, 83–94. ACM. https://doi.org/10.1145/3698205.3729544.

Miao, Jing, Charat Thongprayoon, Supawadee Suppadungsuk, Oscar A Garcia Valencia, and Wisit Cheungpasitporn. 2024. “Integrating Retrieval-Augmented Generation with Large Language Models in Nephrology: Advancing Practical Applications.” *Medicina* 60 (3): 445. https://doi.org/10.3390/medicina60030445.

Mock, Moritz, Jorge Melegati, and Barbara Russo. 2024. “Generative AI for Test Driven Development: Preliminary Results.” In *International Conference on Agile Software Development*, 24–32. Springer. https://doi.org/10.1007/978-3-031-72781-8_3.

Northwood, Chris. 2018. *The Full Stack Developer: Your Essential Guide to the Everyday Skills Expected of a Modern Full Stack Web Developer*. Springer.

Nyaaba, Matthew, Lehong Shi, Macharious Nabang, Xiaoming Zhai, Patrick Kyeremeh, Samuel Arthur Ayoberd, and Bismark Nyaaba Akanzire. 2024. “Generative AI as a Learning Buddy and Teaching Assistant: Pre-Service Teachers’ Uses and Attitudes.” *arXiv.org*. https://doi.org/10.48550/arXiv.2407.11983.

Osherove, Roy. 2009. “The Art of Unit Testing: With Examples In.” *NET Manning*.

Pearson, Helen. 2025. “Are the Internet and AI Affecting Our Memory.” *Nature* 638 (8049): 26–28. https://doi.org/10.1038/d41586-025-00292-z.

Ponmalar, P Sridevi et al. 2025. “Evaluation Framework for Large Language Models: Assessing Factual Consistency and Answer Relevancy in Question-Answering Tasks.” In *2025 2nd International Conference on Research Methodologies in Knowledge Management, Artificial Intelligence and Telecommunication Engineering (RMKMATE)*, 1–6. IEEE; IEEE. https://doi.org/10.1109/RMKMATE64874.2025.11042605.

Pourcel, Julien, Cédric Colas, Gaia Molinaro, Pierre-Yves Oudeyer, and Laetitia Teodorescu. 2024. “Aces: Generating a Diversity of Challenging Programming Puzzles with Autotelic Generative Models.” *Neural Information Processing Systems* 37: 67627–62. https://doi.org/10.52202/079017-2160.

Raju, Bondu. 2025. “Generative AI as a Teaching Assistant: Opportunities and Challenges.” *International Journal of Integrated Research and Practice* 1: 28–36. https://doi.org/10.25215/31075037.031.

Rottner, Renee, Lenore Porter, Jason Bock, Jordan Jannone, Rory Walsh Senerchia, Janet Ward, and Joshuah Whittinghill. 2025. “AI and the Digital Divide.” In *Teaching and Learning in the Age of Generative AI*, 309–31. Routledge. https://doi.org/10.4324/9781032688602-19.

Suárez, David, and Begoña García-Mariñoso. 2025. “On the Verge of a Digital Divide in the Use of Generative AI?” *Telecommunications Policy* 49 (7): 102997. https://doi.org/10.1016/j.telpol.2025.102997.

Trautsch, Fabian, and Jens Grabowski. 2017. “Are There Any Unit Tests? An Empirical Study on Unit Testing in Open Source Python Projects.” In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 207–18. IEEE; IEEE. https://doi.org/10.1109/ICST.2017.26.

Vorel, Roman. 2025. “Generative AI for Coding and Unit Testing.” In *NoOps: How AI Agents Are Reinventing DevOps and Software*, 105–18. Apress. https://doi.org/10.1007/979-8-8688-1694-9_6.

Watterson, Steven, Sarah Atkinson, Elaine Murray, and Andrew McDowell. 2025. “AI as a Teaching Tool and Learning Partner.” *arXiv.org*. https://doi.org/10.48550/arXiv.2509.13899.