ISSN: 2473-4901
v12 n6931

Updated Comparison of Generative AI Solutions and Textbook Solutions in an Introductory Programming Course

Teko Jan Ernst Bekkering
ernst.bekkering@gmail.com
Northeastern State University
Tahlequah, OK, USA

Abstract

Three years ago, Bekkering and Harrington (2023) reported on the ability of generative AI (GenAI) to generate computer code. As the gold standard, the textbook solutions were used to compare the solutions generated with ChatGPT-4 plugged into the Bing search engine. The results showed that while the book solutions were not perfect at meeting 90.8% of exercise requirements, generative AI still fell short with meeting 78.9% of requirements. Since that time, generative AI has undergone significant changes and advancements. This study repeats the Spring 2023 research with Copilot, the successor in the Edge browser. The Spring 2026 results show that generative AI now surpasses solutions provided by the textbook authors (90.8%) with 99.3%. Implications and recommendations are discussed.

Keywords: artificial intelligence, generative AI, ChatGPT, C++, introductory programming courses.

Updated Comparison of Generative AI Solutions and Textbook Solutions in an Introductory Programming Course

Teko Jan Ernst Bekkering

1. INTRODUCTION

Generative AI has been used in programming courses in higher education for only a few years. Early on, one of the major concerns about using AI was its ability to generate good computer code. Bekkering and Harrington (2023) compared computer programming exercise solutions provided by textbook authors with solutions provided by ChatGPT. Since this study from Spring 2023, the perception of using generative AI in programming education has changed significantly. Using GenAI is no longer experimental or controversial. It is now institutionalized; widely researched; and actively reshapes assessment, curriculum design, and student performance (Gordon et al., 2026). Universities now teach explicit AI literacy, redesign programming assessments to account for AI assistance, and observe measurable shifts in student performance and skill distribution. These changes are documented in educator surveys, longitudinal syllabus analyses, and empirical studies of student outcomes.

1.1 AI literacy and responsible use

Institutions no longer treat AI as an optional tool. AI literacy and responsible use are now formal curriculum components. Programming courses increasingly include (Gordon et al., 2026):

- Explicit instruction on using AI coding tools appropriately
- Ethical guidelines and academic integrity policies
- Assignments requiring students to explain or critique AI-generated code
- Training in prompt design and debugging AI output

This is a direct response to widespread educator concerns about student overreliance on AI and the need to preserve conceptual understanding (Kohen-Vacs et al., 2025; Kucera et al., 2025; Terroso & Pinto, 2025).

1.2 Redesigned programming exercises

Instructors worldwide have shifted toward code comprehension exams rather than pure code writing, in-person or oral assessments to verify individual understanding, AI-aware rubrics that distinguish human reasoning from AI-generated syntax, and assignments requiring students to justify design choices or annotate AI output (Barari et al., 2026; Gordon et al., 2026; Prather et al., 2025; Terroso & Pinto, 2025).

1.3 Student performance

Generative AI acts as a "procedural equalizer" in programming course work. Scores increased across the board after AI became widely available, and lower-performing students improved the most. Score distributions compressed, with many students clustering near the top which created a ceiling effect (Barari et al., 2026; Viswanathan et al., 2026).

1.4 Formal AI policies

The field has now recognized the "importance of establishing clear institutional policies" (Yalwa et al., 2026). A longitudinal analysis from 2024 to 2026 showed that institutions now publish formal AI policies in syllabi (Bui & Dong, 2026). Explicit GenAI policies are rapidly adopted, the use of AI has moved from blanket bans in 2023 to regulated use in 2026, disclosure of AI use is required, and AI tools are integrated into assignments, labs, and debugging workflows. This marks a shift from ad-hoc instructor decisions in the early days to institutional governance at present (Giugliano, 2026).

1.5 Standard use of AI coding tools

In 2023, typical CS1 courses focused on novices writing many small programs to master basic constructs like loops and selection structures (Becker et al., 2023). Assignments emphasized the manual "translation of logical intent into rigid syntactic structures" (Lee et al., 2026). Writing many small programs was the primary approach to learning (Hahnel et al., 2026). By 2026, universities increasingly treat using AI coding tools like Copilot, CodeCompose, and Gemini as essential workplace skills. The ACM task force notes that courses now teach how to evaluate AI-generated code, how to integrate AI into professional workflows, using AI for scaffolding, debugging, and test generation (Gordon et al., 2026). This is a major shift from 2023, when faculty were still debating whether the use of AI should be allowed at all.

1.6 From syntax to reasoning

Before the use of GenAI, introductory programming courses focused on writing simple programs to teach basic coding principles. Becker et al. (2023) note that LLMs demonstrate the ability to produce "syntactically and logically correct code from natural language prompts that rival the performance of high-performing introductory programming students". Now that GenAI can generate this low-level code, programming courses now emphasize problem decomposition, algorithmic thinking, testing and verification, understanding constraints and specifications, and explaining *why* code works – not just producing code.

1.7 New challenges and opportunities

Educators now face issues that were not visible in 2023. GenAI can still hallucinate in code explanations. One manual analysis of buggy student solutions showed that while 93% of AI-generated explanations identified at least one error, 50% contained at least one incorrect statement (Franklin et al., 2025). Students bypass conceptual struggles when AI generates working code in moments and students skip the "desirable difficulties" for deep learning (Fowles et al., 2026; Hahnel et al., 2026). Assignments and tests have to be developed to measure reasoning rather than syntax, leading to a move to oral exams, in-person mastery checks, and "explain in plain English" tasks to verify true understanding (Franklin et al., 2025; Gordon et al., 2026; Lee et al., 2026). Access to the most capable models often requires a paid subscription, creating equity concerns (Prather et al., 2025). However, these concerns are balanced by opportunities like focusing on deeper conceptual learning by offloading syntax, shifting from computer programming to software engineering (Hahnel et al., 2026). Students have earlier exposure to refactoring and testing, using personalized adaptive learning, and working with more complex programs early on (Becker et al., 2023; Deriba et al., 2024; Hahnel et al., 2026)

The structure of this paper is as follows. The next section covers the literature review, followed by a description of how the original study was updated, an analysis of the results, and finally conclusions and recommendations.

2. LITERATURE REVIEW

GenAI can now do more than just generate code, and we see it increasingly used in different ways to help students learn.

2.1 Prompt engineering for code generation

As noted by Bekkering and Harrington (2023), writing specific inputs for GenAI is important. Major software companies like Google, Microsoft, and OpenAI publish online materials to help users generate better prompts (Google Career Certificates, 2024; Microsoft, 2023/2026; OpenAI, 2026). Students need to learn to construct natural language prompts precise enough to generate code that meets strict specifications. This helps them to develop requirement specification, edge-case identification, and technical communication (Denny et al., 2023; Pădurean et al., 2025). Prompt engineering also requires a conceptual understanding of coding principles in general, so that these can be used in the specifications. Prather et al. (2025) cite developers who advise learners to "learn enough domain knowledge that lets you adequately explain your needs to an AI".

2.2 Tutoring and conceptual scaffolding

When students learn abstract CS concepts like recursion, pointers, or array indexing, GenAI can provide tailored analogies based on student interests (Becker et al., 2023; Franklin et al., 2025). This can include the traditional explanation of stack frames using a stack of cafeteria trays, but also non-traditional analogies like video game save states. Once simple code examples are used to illustrate the concepts, line-by-line code walkthroughs explain existing book code at customizable experience levels, helping students to develop reading skills before writing skills (Deriba et al., 2024).

Progressing to creating code, GenAI can be used as a Socratic tutor ("Teach me how to write a program that xxxx. Do not write any code but tell me step by step what to write and give me feedback after each step") (Gordon et al., 2026; Groher et al., 2026; Viswanathan et al., 2026). This forces students to apply basic programming steps in sequence and helps them develop planning of larger structures. When they make mistakes, compiler error statements like "IndexError: list index out of range" can be translated into plain, approachable English with explanations of why the code does not compile.

Finally, GenAI feedback of code reviews and style acts as an automated code reviewer that evaluates student work against instructor requirements (Crandall et al., 2023) or style guides (Woodrow et al., 2024), such as PEP 8 in Python , variable naming best practices, modularity, choice of structures, and comment clarity. Feedback is increasingly accurate, with Franklin et al. (2025) reporting correctly assessing syntax for 89% and conceptual understanding for 92%.

2.3 Instructor-facing design and assessment

On one hand, GenAI creates the problem of directly copying solutions based on existing exercises associated with educational materials. In the past, students might find or purchase instructor solutions online. With GenAI, basic programming tasks may be trained on "countless examples of solutions to these tasks" that are "widely available on the internet" (Franklin et al., 2025). However, faculty can now use the same tool to generate multiple variations of core programming tasks (e.g. nested loops) set in different real-world contexts (sports statistics, digital music playlists, climate data) to increase student engagement and deter copying (Becker et al., 2023; Franklin et al., 2025).

Beyond mere variation of contexts, faculty can easily create realistic code snippets containing intentional syntax, semantic, or logic errors. Students can use these to trace, debug, and write unit tests (Kohen-Vacs et al., 2025; Viswanathan et al., 2026). In addition to GenAI tools, faculty can use dedicated tools like Edugator (Diaz et al., 2025) to create customized problems in multiple languages, validating them against auto-generated solutions before releasing them to students.

Finally, faculty can automatically create comprehensive lists of tests (including boundary conditions like empty lists, negative numbers, or non-standard input) for use in grading rubrics and classroom demonstrations. According to Alkafaween et al. (2025), LLM-generated test suites for introductory programming are not only accurate but are "in many cases, as comprehensive as instructor-created test suites".

2.4 Novel pedagogical paradigms

Teaching students to use GenAI properly also includes exploring the limits and pitfalls of using these tools (Becker et al., 2023; Prather et al., 2025). Sometimes GenAI creates hallucinations, and students need to review AI-generated code that contains logic errors or inefficient algorithms. This is especially problematic in complex or unfamiliar contexts (Hilario et al., 2024) or in advanced programming courses (Yalwa et al., 2026). The focus shifts from writing code to critical evaluation using the basic coding principles at scale to audit, test, and correct the output (Mahon et al., 2024).

One of the mainstays of programming courses is program tracing (Hertz & Jump, 2013; Pozzan et al., 2025). Students step through a program line by line, traditionally with paper and pencil, later by using breakpoints and stepping through or over, and now using automated tools like C++ Tutor (Python Tutor, 2026). Faculty can now easily create "mystery code" snippets where students must predict the output or draw execution trace tables, which helps to develop strong mental models of programming language runtime.

2.5 GenAI tools for introductory programming classes

Since the original study, GenAI tools suitable for use in programming classes have proliferated. Students have a plethora of choices between free and paid models. These currently include, but are not limited to:

- General-purpose web models
  - ChatGPT (OpenAI): the baseline tool for code generation, step-by-step logic tracing, and custom explanations.
  - Claude (Anthropic): highly favored for code auditing and detailed explanations due to its strong performance in technical writing, code reading, and error analysis.
  - Google Gemini: widely used for explaining concepts, generating visual flowcharts/ markdown, and analyzing code.
  - Copilot (Microsoft): high accessibility due to integration with Microsoft systems, including the Microsoft Office suite.
- Integrated with code editors
  - Github Copilot: the most widely cited IDE assistant in CS education.
  - Cursor: an AI-native code editor
  - Replit AI: browser-based IDE popular in high school and introductory courses
- Interactive learning and textbook platforms
  - Zybooks AI/ Codecademy AI: provides real-time hint generation when students get stuck
- Custom and guardrailed "Socratic" Educational AI
  - CS50.AI: built specifically for Harvard's introductory computer science course.
  - Khanmigo (Khan Academy): educational assistant built on GPT models that challenges students without completing tasks

2.6 Accuracy, reliability, and security of AI-generated code

Beyond higher education, there exists a significant distrust of AI-generated code. This is supported both by industry publications and peer-reviewed research. Ninety-six percent of developers do not fully trust that it is functionally correct, and 61% agree that AI often produces code that "looks correct but isn't reliable" (Sonar, 2026). Song et al. (2023) classify errors as semantic (logical misunderstanding of instructions) and syntactic (structural misconceptions), and find that most semantic errors are "garbage code" where models lose track of long-term dependencies and generate meaningless snippets or comments-only output. Results do depend on the language used. For instance, Python has higher vulnerability rates than Rust with its memory-safe design patterns (Schreiber & Tippe, 2026). Results also depend on the model used. In a study of the exploitability of AI code, seven models had a mean vulnerability rate of 55.8% ranging from 48.4% for Gemini 2.5 to 62.4% for GPT-4o (Blain & Noiseux, 2026). After integration into real-world software projects, 15% of commits introduce at least one quality issue and 22.7% of these issues survive long-term, creating a "hidden maintenance tax" (Liu et al., 2026). Even when code looks like high quality code, production incidents occur in 78% of organizations because the code fails in the real world (*Introducing the State of AI Coding 2026*, 2026). Finally, AI coding assistants may increase the number of pull requests filled, but organizational limitations limit the implementation and improvement in business outcomes. This is the AI productivity paradox in software development (Faros, 2025)

3. METHODOLOGY, SAMPLE, AND DATA COLLECTION

Consistent with the previous study of Bekkering and Harrington (2023), the textbook instructions for the then-current CS1 textbook were used. The original book instructions and the C++ adapted prompts are available at the original link AppendixA.docx (Bekkering & Harrington, 2023). As noted before, some textbook solutions were not available and those exercises were omitted.

In the original study, the free version of ChatGPT-4 integrated with the Bing search engine was used because it was freely available and was trained on all current information from the internet. The current version of Microsoft's AI tool is Copilot, which is both available in the Edge browser (Figure 1) and as a stand-alone tool (Figure 2).

ISSN: 2473-4901
v12 n6931

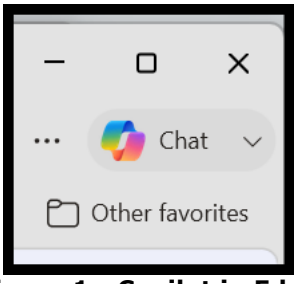

Figure 1 - Copilot in Edge

Figure 2 - Copilot stand-alone

Even though Claude is generally considered to be the best tool for coding and software development (Abid, 2026) and Copilot in VS Code for integrated development (FindSkill Team, 2026), Copilot in the Edge browser was used to maximize strength of comparison. The same 152 prompts as for the original study were copied in the Copilot interface, and the code copied and pasted to VS Code to check whether (yes/no) all requirements in the instructions were met. Since the textbook solutions had already been checked, this only needed to be done for the GenAI solutions. The book data for 2023 were simply carried forward.

4. ANALYSIS AND FINDINGS

In contrast to the original study, GenAI performed better than the textbook solutions. Originally, the success percentage of ChatGPT was 78.9% compared with 90.8% of textbook solutions. The success rate of GenAI has now risen to 99.3%. The only exercise where ChatGPT failed, was one exercise with the instruction "Demonstrate the function in a program that lets the user experiment with different values for the formula's terms" and the program did not include a while loop. Since the book solutions were statistically significantly better than the GenAI solutions in the past with a p-value of 0.00198, the percentages were again compared with a Z-test in Excel. The null hypothesis for the one-tailed test with proportion for ChatGPT (p_c) higher than the proportion for the book (p_b) was

$$H_0: p_c <= pb$$

and the alternative hypothesis

$$H_a: p_c > p_b$$

The p-value was calculated as follows: the book proportion p_b is 0.908; the ChatGPT proportion p_c is 0.993. Both sample sizes s_b and s_c are 152. The pooled sample proportion p_s

$$p_s = (p_c * s_c + p_b * s_b)/ (s_b + s_c) =$$
$$(0.993 * 152 + 0.908 * 152) / (152 + 152) = 0.95066$$

The test statistic z:

$$z = (p_c - p_b) / \sqrt{p_s * (1- p_s) * [(1/ s_b) + (1/ s_c)]}$$
$$= (0.993 - 0.908) / \sqrt{0. 95066* (1- 0.95066) * [(1/ 152) + (1/ 152)]}$$
$$= 3.44259.$$

The p-value can be found on a Z table or calculated with the Excel formula

$$p = 1 – NORM.S.DIST(z,true) = 0.00029$$

At a value of 0.00029, the null hypothesis is rejected and the alternative hypothesis of the ChatGPT success percentage being greater than textbook solutions, is supported. Appendix A provides a detailed comparison for the 2023 and 2026 data.

5. CONCLUSIONS AND RECOMMENDATIONS

GenAI, as represented by Copilot, has progressed to being more accurate than the old gold standard, solutions by the textbook authors. This has significant implications for programming courses, especially introductory programming courses. If AI can solve 99.3% of textbook problems on the first try, the "human bottleneck" needs to shift to higher-level content like requirement specification, testing, and documentation. No longer should faculty rely on using author-provided solutions but work on developing

course materials and assessments with frequent variations using GenAI tools. In fact, considering the availability of tools like NotebookLM which can integrate multiple sources, we should anticipate using reference sources like C++ references at https://cppreference.com/ , Python documentation at https://docs.python.org/3/, and the C# language references at https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/ to generate course materials. Using selected course content from these reference sources will enable faculty to more seamlessly integrate higher-level content such as prompt engineering, testing, and documentation, which now take a backseat to coding syntax, into programming courses. Important basic skills like reading "mystery code" snippets and program tracing can be streamlined by generating more and more varied practice materials with GenAI. Since good specification is so important, instruction in prompt engineering should be included in an early stage.

This study has several limitations. The results are only valid for one textbook, and other textbooks could conceivably have perfect instructor resources. Results for GenAI are generated with one-shot prompting analogous to one-time exercise instructions in textbooks. More advanced prompting techniques like role-based prompting, few-shot examples, chain-of-thought prompting, and tree-of-thought prompting may produce different results.

The measurement of performance is narrow, as in the original study. Since GenAI generates code based on exercise instructions in programming textbooks, and is compared with the author solutions, the only success measure is whether the requirements in the exercise are met or not (yes/no). This leads to a potential artificially high completion rate of 99.3%, and does not address quality, security, and maintainability. A single rater was used, but again, due to the simplicity of the task, this should not be a major issue.

Furthermore, even though I compared two Microsoft products, the comparison is not completely apples to apples due to the progress in technology. In 2023, Bing's integration of ChatGPT-4 provided users with a conversational interface built largely around the raw GPT-4 model, supplemented by Bing search, and protected by early-stage safety mechanisms. The system lacked awareness of user context and user workflow, resulting in code generation that relied primarily on model memory. In 2026, Copilot in Edge is still powered by GPT-4 models but connects to open applications and the user's established coding patterns. Better tools like Claude exist now too, but this makes the better performance of Copilot even more relevant.

Most importantly for limitations, the level of programming is introductory, and results will currently not reflect higher levels of software development. The concerns about accuracy, reliability, and security of AI-generated code have been discussed earlier. Based on these sources, generative AI should still be seen as a teaching tool in higher education, and a tool for providing a starting point for professional software development. Further studies could test complex algorithms and investigate how GenAI performs on non-deterministic problems, as well as address the concept of explainability.

In conclusion, using GenAI in introductory programming classes has great potential to improve education in Computer Science, as long as we shift from basic coding exercises to higher-level skills.

6. REFERENCES

Abid, A. (2026, June 17). Best AI Models 2026 | Top Ai Models Compared. *Aicomparison.Ai*. https://aicomparison.ai/best-ai-models/

Alkafaween, U., Albluwi, I., & Denny, P. (2025). Automating Autograding: Large Language Models as Test Suite Generators for Introductory Programming. *Journal of Computer Assisted Learning*, *41*(1). https://doi.org/10.1111/jcal.13100

Barari, G., Ortega-Moody, J., Jenab, K., Ward, T., & Siebold, K. (2026). AI as a Procedural Equalizer: Performance Comparison in Programming-Based Engineering Coursework Following the Emergence of Generative AI. *Applied Sciences*, *16*(10), 4884. https://doi.org/10.3390/app16104884

Becker, B. A., Craig, M., Denny, P., Keuning, H., Kiesler, N., Leinonen, J., Luxton-Reilly, A., Malmi, L., Prather, J., & Quille, K. (2023). Generative AI in Introductory Programming. *Computer Science Curricula*, 438–439.

Bekkering, E., & Harrington, P. (2023). If You Want Something Specific, Ask for it Specifically: A Comparison of Generative AI Solutions and Textbook Solutions in an Introductory Programming Course. *Proceedings of the ISCAP Conference*, *9*(n5951). https://doi.org/10.62273/YQWP1758

Blain, D., & Noiseux, M. (2026). *Broken by Default: A Formal Verification Study of Security Vulnerabilities in AI-Generated Code* (arXiv:2604.05292). arXiv. https://doi.org/10.48550/arXiv.2604.05292

Bui, N. T., & Dong, A. (2026). Adoption of Generative AI Policies in Computing Education: A Longitudinal Syllabus Analysis. *2026 International Conference on Semantic Computing (ICSC)*, 336–342. https://doi.org/10.1109/ICSC67292.2026.00055

Crandall, A. S., Sprint, G., & Fischer, B. (2023). Generative pre-trained transformer (GPT) models as a code review feedback tool in computer science programs. *Journal of Computing Sciences in Colleges*, *39*(1), 38–47.

Denny, P., Kumar, V., & Giacaman, N. (2023). Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2023*, 1136–1142. https://doi.org/10.1145/3545945.3569823

Deriba, F., Sanusi, I. T., O Campbell, O., & Oyelere, S. S. (2024). *Computer programming education in the age of generative AI: Insights from empirical research*. https://doi.org/10.2139/ssrn.4891302

Diaz, M., Karp, D., Tuli, P., & Kapoor, A. (2025). Edugator: An AI-enabled Tool for Creating and Delivering Interactive Computing Content. *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2, SIGCSETS 2025*, 1732. https://doi.org/10.1145/3641555.3705025

Faros. (2025). *The AI Productivity Paradox Research Report*. https://www.faros.ai/blog/ai-software-engineering

FindSkill Team. (2026, January 22). *ChatGPT vs Claude vs Gemini 2026: Best AI for Writing, Coding & Research*. FindSkill.Ai — Learn AI for Your Job. https://findskill.ai/blog/which-ai-should-you-use/

Fowles, P., Falor, E., Bhattarai, S., Edwards, J., & Poulsen, S. (2026). *Combating Harms of Generative AI in CS1 with Code Review Interviews and a Flipped Classroom* (arXiv:2605.21374). arXiv. https://doi.org/10.48550/arXiv.2605.21374

Franklin, D., Denny, P., Gonzalez-Maldonado, D. A., & Tran, M. (2025). *Generative AI in computer science education: Challenges and opportunities*. Cambridge University Press. https://www.cambridge.org/core/elements/generative-ai-in-computer-science-education/0A22106CBD7FCB391FD120C56E21420F

Giugliano, A. (2026, June 25). *Generative AI Policies at the World's Top Universities: 2026 Update*. https://www.thesify.ai/blog/generative-ai-policies-top-universities-2026

Google Career Certificates. (2024, May 13). *Discover the Art of Prompting | Google AI Essentials* [Video recording]. https://www.youtube.com/watch?v=jNNatjruXx8

Gordon, S., Denny, P., Keuning, H., Kiesler, N., Kumar, A. N., Kumar, V., Leinonen, J., & Prather, J. (2026). *ACM Task Force on Generative AI and Programming Assessment*. https://acm-education-genai-task-force.github.io/ACM_Taskforce_GenAI_Report_16Feb26.pdf

Groher, I., Heissenberger, P., & Vierhauser, M. (2026). *Design and Deployment of a Course-Aware AI Tutor in an Introductory Programming Course*. https://doi.org/10.5220/0015021900004021

Hahnel, C., Benario, J. G., Keuning, H., Kiesler, N., Kohn, T., Komm, D., Lewis, C., Lohr, D., Reeves, B., & Šavelka, J. (2026). *4.2 GenAI in Programming Education: Hypes, Hoaxes, and Hopes* (p. 275). Dagstuhl Seminar.

Hertz, M., & Jump, M. (2013). Trace-based teaching in early programming courses. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, 561–566. https://doi.org/10.1145/2445196.2445364

Hilario, E., Azam, S., Sundaram, J., Imran Mohammed, K., & Shanmugam, B. (2024). Generative AI for pentesting: The good, the bad, the ugly. *International Journal of Information Security*, *23*(3), 2075–2097. https://doi.org/10.1007/s10207-024-00835-x

Introducing the State of AI Coding 2026. (2026, June 10). https://newrelic.com/blog/ai/state-of-ai-coding-2026

Kohen-Vacs, D., Usher, M., & Jansen, M. (2025). Integrating Generative AI into Programming Education: Student Perceptions and the Challenge of Correcting AI Errors. *International Journal of Artificial Intelligence in Education*, *35*(5), 3166–3184. https://doi.org/10.1007/s40593-025-00496-4

Kucera, L., Akhmetov, I., Izu, C., & Omojokun, O. (2025). Adapting Computing Curricula in the Era of Automated Software Development. *Proceedings of the ACM Global Computing Education Conference 2025 - Volume 2*, 353–355. https://doi.org/10.1145/3736251.3754332

Lee, Y.-C., Boonprakong, N., Tan, Y., Soh, H., Potanin, A., Kumar, V., Sinha, A. K., Qian, C., Denny, P., El-Assady, M., Oakley, I., Renzella, J., Zhang, A., Singh, J., Lee, W. S., Lin, H.-T., E, J. L., Tang, A., Burnett, M. M., … Cristea, A. I. (2026). *Reshaping Undergraduate Computer Science Education in the Generative AI Era* (Version 1). https://doi.org/10.48550/arXiv.2606.07545

Liu, Y., Widyasari, R., Zhao, Y., Irsan, I. C., Chen, J., & Lo, D. (2026). *Debt Behind the AI Boom: A Large-Scale Empirical Study of AI-Generated Code in the Wild*. https://doi.org/10.48550/arXiv.2603.28592

Mahon, J., Mac Namee, B., & Becker, B. A. (2024). Guidelines for the Evolving Role of Generative AI in Introductory Programming Based on Emerging Practice. *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, 10–16. https://doi.org/10.1145/3649217.3653602

Microsoft. (2026). *Microsoft/generative-ai-for-beginners* [Jupyter Notebook]. https://github.com/microsoft/generative-ai-for-beginners (Original work published 2023)

OpenAI. (2026). *Prompt engineering | OpenAI API*. OpenAI Developers. https://developers.openai.com/api/docs/guides/prompt-engineering

Pădurean, V.-A., Denny, P., Gotovos, A., & Singla, A. (2025). Prompt Programming: A Platform for Dialogue-based Computational Problem Solving with Generative AI Models. *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, 458–464. https://doi.org/10.1145/3724363.3729094

Pozzan, G., Padova, C., Montuori, C., Arfé, B., & Vardanega, T. (2025). On the Use of Tracing to Diagnose Misconceptions About Iteration. *Informatics in Schools. Fostering Problem-Solving, Creativity, and Critical Thinking Through Computer Science Education: 18th International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, ISSEP 2025, Trier, Germany, September 8–10, 2025, Proceedings*, 139–152. https://doi.org/10.1007/978-3-032-01222-7_11

Prather, J., Leinonen, J., Kiesler, N., Gorson Benario, J., Lau, S., MacNeil, S., Norouzi, N., Opel, S., Pettit, V., Porter, L., Reeves, B. N., Savelka, J., Smith, D. H., Strickroth, S., & Zingaro, D. (2025). Beyond the Hype: A Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools. *2024 Working Group Reports on Innovation and Technology in Computer Science Education*, 300–338. https://doi.org/10.1145/3689187.3709614

Python Tutor. (2026). *C++ Visualizer, Visual Debugger, and AI Tutor*. C++ Visualizer, Visual Debugger, and AI Tutor. https://pythontutor.com/cpp.html

Schreiber, M., & Tippe, P. (2026). *Security Vulnerabilities in AI-Generated Code: A Large-Scale Analysis of Public GitHub Repositories* (Vol. 16219, pp. 153–172). https://doi.org/10.1007/978-981-95-3537-8_9

Sonar. (2026). *Developer Survey report*. https://www.sonarsource.com/the-state-of-code/developer-survey-report/

Song, D., Zhou, Z., & Wang, Z. (2023). *An Empirical Study of Code Generation Errors made by Large Language Models*. https://mapsworkshop.github.io/assets/LLM_Code_Error_Analysis_MAPS2023_camera-ready.pdf

Terroso, T., & Pinto, M. (2025). The Influence of GenAI on the Evaluation of Computer Programming Students in Higher Education. *6th International Computer Programming Education Conference (ICPEC 2025)*, 18–1. https://drops.dagstuhl.de/storage/01oasics/oasics-vol133-icpec2025/OASIcs.ICPEC.2025.18/OASIcs.ICPEC.2025.18.pdf

Viswanathan, V., Weitz, R., & Rosenthal, D. (2026). Teaching Tip: Generative AI in Education: Building Intelligent Tutors With Large Language Models. *Journal of Information Systems Education*, *37*(2), 167–183. https://doi.org/10.62273/COCJ8767

Woodrow, J., Malik, A., & Piech, C. (2024). AI Teaches the Art of Elegant Coding: Timely, Fair, and Helpful Style Feedback in a Global Course. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 1442–1448. https://doi.org/10.1145/3626252.3630773

Yalwa, A. S., Othman, M. S., Yusuf, L. M., & Almarshadi, M. S. (2026). Generative AI in Higher Education: A Systematic Review with Emphasis on Programming and Computer Science Education. *International Journal of Advanced Computer Science and Applications*, *17*(5). https://doi.org/10.14569/IJACSA.2026.0170531

ISSN: 2473-4901
v12 n6931

APPENDIX A

Comparison book solutions and generative AI solutions

(original data available upon request)

2023

Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book
2-1	yes	no	3-1	yes	yes	4-1	yes	yes	5-1	yes	yes	6-1	yes	yes	7-1	no	yes	8-1	yes	yes
2-2	yes	no	3-2	yes	yes	4-2	yes	yes	5-2	no	yes	6-2	yes	yes	7-2	yes	yes	8-2	yes	yes
2-3	yes	no	3-3	yes	yes	4-3	yes	yes	5-3	yes	yes	6-3	yes	yes	7-3	yes	yes	8-3	yes	yes
2-4	yes	yes	3-4	yes	yes	4-4	yes	yes	5-4	yes	yes	6-4	yes	yes	7-4	yes	yes	8-4	yes	yes
2-5	yes	yes	3-5	yes	yes	4-5	yes	yes	5-5	yes	no	6-5	yes	yes	7-5	no	yes	8-5	yes	no
2-6	yes	yes	3-6	yes	yes	4-6	yes	yes	5-6	yes	yes	6-6	yes	yes	7-6	yes	yes	8-6	no	yes
2-7	yes	yes	3-7	yes	yes	4-7	no	yes	5-7	yes	yes	6-7	yes	yes	7-7	no	yes	8-7	no	yes
2-8	yes	yes	3-8	yes	yes	4-8	yes	yes	5-8	no	yes	6-8	no	yes	7-8	yes	yes	8-8	yes	yes
2-9	yes	yes	3-9	no	yes	4-9	yes	yes	5-9	yes	yes	6-9	no	yes	7-9	yes	no	8-9	yes	yes
2-10	yes	yes	3-10	yes	yes	4-10	yes	yes	5-10	yes	yes	6-10	yes	yes	7-10	no	yes	8-10	yes	yes
2-11	yes	yes	3-11	yes	yes	4-11	no	yes	5-11	yes	yes	6-11	no	yes	7-11	yes	no	8-11	yes	yes
2-12	yes	no	3-12	yes	yes	4-12	yes	no	5-12	yes	yes	6-12	no	yes	7-12	yes	yes	8-12	*	*
2-13	yes	yes	3-13	yes	yes	4-13	yes	yes	5-13	yes	yes	6-13	yes	yes	7-13	yes	no			
2-14	yes	yes	3-14	yes	yes	4-14	yes	yes	5-14	yes	yes	6-14	no	yes	7-14	yes	yes			
2-15	yes	yes	3-15	yes	yes	4-15	yes	yes	5-15	yes	yes	6-15	no	yes	7-15	yes	yes			
2-16	yes	yes	3-16	yes	yes	4-16	yes	yes	5-16	no	yes	6-16	yes	yes	7-16	no	yes			
2-17	yes	yes	3-17	no	yes	4-17	yes	yes	5-17	yes	yes	6-17	yes	no	7-17	yes	yes			
2-18	yes	yes	3-18	yes	yes	4-18	yes	yes	5-18	no	yes	6-18	no	yes	7-18	yes	yes			
2-19	yes	yes	3-19	yes	yes	4-19	yes	yes	5-19	yes	yes	6-19	no	yes	7-19	*	*			
2-20	no	no	3-20	yes	yes	4-20	yes	no	5-20	yes	yes	6-20	no	yes	7-20	*	*			
			3-21	yes	yes	4-21	yes	yes	5-21	yes	yes	6-21	yes	yes	7-21	yes	yes			
			3-22	yes	yes	4-22	yes	yes	5-22	yes	yes	6-22	no	yes						
			3-23	yes	yes	4-23	yes	yes	5-23	yes	yes	6-23	yes	yes						
			3-24	yes	yes	4-24	no	yes	5-24	yes	yes	6-24	no	yes						
			3-25	no	yes	4-25	yes	no	5-25	no	yes									
						4-26	yes	yes	5-26	*	*									
						4-27	no	yes	5-27	*	*									
						4-28	no	yes												

ch02			ch03			ch04			ch05			ch06			ch07			ch08		
ChatGPT	19	95.0%	ChatGPT	22	88.0%	ChatGPT	23	82.1%	ChatGPT	20	80.0%	ChatGPT	13	54.2%	ChatGPT	14	73.7%	ChatGPT	9	81.8%
book	15	75.0%	book	25	100.0%	book	25	89.3%	book	24	96.0%	book	23	95.8%	book	16	84.2%	book	10	90.9%
count	20		count	25		count	28		count	25		count	24		count	19		count	11	

*	no solution in the book solution bank	ChatGPT total:	120	ChatGPT %:	78.9%
		book total:	138	book %:	90.8%
		count total:	152		

Legend	
yes	requirements met
no	requirements not met

Two sample -test (one-tailed)	
book proportion	0.908
book sample size	152
ChatGPT proportion	0.789
ChatGPT sample size	152

Pooled sample proportion	0.84868
Test statistic	2.88085
p-value	0.00198

ISSN: 2473-4901
v12 n6931

2026

Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book	Exercise	ChatGPT	book
2-1	yes	no	3-1	yes	yes	4-1	yes	yes	5-1	yes	yes	6-1	yes	yes	7-1	yes	yes	8-1	yes	yes
2-2	yes	no	3-2	yes	yes	4-2	yes	yes	5-2	yes	yes	6-2	yes	yes	7-2	yes	yes	8-2	yes	yes
2-3	yes	no	3-3	yes	yes	4-3	yes	yes	5-3	yes	yes	6-3	yes	yes	7-3	yes	yes	8-3	yes	yes
2-4	yes	yes	3-4	yes	yes	4-4	yes	yes	5-4	yes	yes	6-4	yes	yes	7-4	yes	yes	8-4	yes	yes
2-5	yes	yes	3-5	yes	yes	4-5	yes	yes	5-5	yes	no	6-5	yes	yes	7-5	yes	yes	8-5	yes	no
2-6	yes	yes	3-6	yes	yes	4-6	yes	yes	5-6	yes	yes	6-6	yes	yes	7-6	yes	yes	8-6	yes	yes
2-7	yes	yes	3-7	yes	yes	4-7	yes	yes	5-7	yes	yes	6-7	yes	yes	7-7	yes	yes	8-7	yes	yes
2-8	yes	yes	3-8	yes	yes	4-8	yes	yes	5-8	yes	yes	6-8	yes	yes	7-8	yes	yes	8-8	yes	yes
2-9	yes	yes	3-9	yes	yes	4-9	yes	yes	5-9	yes	yes	6-9	no	yes	7-9	yes	no	8-9	yes	yes
2-10	yes	yes	3-10	yes	yes	4-10	yes	yes	5-10	yes	yes	6-10	yes	yes	7-10	yes	yes	8-10	yes	yes
2-11	yes	yes	3-11	yes	yes	4-11	yes	yes	5-11	yes	yes	6-11	yes	yes	7-11	yes	no	8-11	yes	yes
2-12	yes	no	3-12	yes	yes	4-12	yes	no	5-12	yes	yes	6-12	yes	yes	7-12	yes	yes	8-12	*	*
2-13	yes	yes	3-13	yes	yes	4-13	yes	yes	5-13	yes	yes	6-13	yes	yes	7-13	yes	no			
2-14	yes	yes	3-14	yes	yes	4-14	yes	yes	5-14	yes	yes	6-14	yes	yes	7-14	yes	yes			
2-15	yes	yes	3-15	yes	yes	4-15	yes	yes	5-15	yes	yes	6-15	yes	yes	7-15	yes	yes			
2-16	yes	yes	3-16	yes	yes	4-16	yes	yes	5-16	yes	yes	6-16	yes	yes	7-16	yes	yes			
2-17	yes	yes	3-17	yes	yes	4-17	yes	yes	5-17	yes	yes	6-17	yes	no	7-17	yes	yes			
2-18	yes	yes	3-18	yes	yes	4-18	yes	yes	5-18	yes	yes	6-18	yes	yes	7-18	yes	yes			
2-19	yes	yes	3-19	yes	yes	4-19	yes	yes	5-19	yes	yes	6-19	yes	yes	7-19	*	*			
2-20	yes	no	3-20	yes	yes	4-20	yes	no	5-20	yes	yes	6-20	yes	yes	7-20	*	*			
			3-21	yes	yes	4-21	yes	yes	5-21	yes	yes	6-21	yes	yes	7-21	yes	yes			
			3-22	yes	yes	4-22	yes	yes	5-22	yes	yes	6-22	yes	yes						
			3-23	yes	yes	4-23	yes	yes	5-23	yes	yes	6-23	yes	yes						
			3-24	yes	yes	4-24	yes	yes	5-24	yes	yes	6-24	yes	yes						
			3-25	yes	yes	4-25	yes	no	5-25	yes	yes									
						4-26	yes	yes	5-26	*	*									
						4-27	yes	yes	5-27	*	*									
						4-28	yes	yes												
ch02			ch03			ch04			ch05			ch06			ch07			ch08		
ChatGPT	20	100.0%	ChatGPT	25	100.0%	ChatGPT	28	100.0%	ChatGPT	25	100.0%	ChatGPT	23	95.8%	ChatGPT	19	100.0%	ChatGPT	11	100.0%
book	15	75.0%	book	25	100.0%	book	25	89.3%	book	24	96.0%	book	23	95.8%	book	16	84.2%	book	10	90.9%
count	20		count	25		count	28		count	25		count	24		count	19		count	11	

*	no solution in the book solution bank	ChatGPT total:	151	ChatGPT %:	99.3%
		book total:	138	book %:	90.8%
		count total:	152		

Legend	
yes	requirements met
no	requirements not met

Two sample -test (one-tailed)	
book proportion	0.908
book sample size	152
ChatGPT proportion	0.993
ChatGPT sample size	152

Pooled sample proportion	0.95066
Test statistic	3.44259
p-value	0.00029