Transforming Formative Assessment Method in Introductory Programming Course With ChatGPT

Reginamary Matthews, University of Nottingham Malaysia, Malaysia

The Southeast Asian Conference on Education 2025
Official Conference Proceedings

Abstract
ChatGPT is one of the AI chatbots that can generate programming code and explain the flow of a computer program clearly to its users. Its ability to write computer programs and detect and fix errors has been a profound debate in recent years. The typical programming assignment requires students to design and write code and test it to ensure the program works without errors. Learning both declarative knowledge (understanding programming concepts, syntax, and semantics) and procedural knowledge (applying declarative knowledge to write a program to solve a problem) is a typical pedagogical method used in a programming course. Learning programming through errors is a novel learning approach to teaching and learning programming languages. Utilising ChatGPT as an AI teaching assistant is a promising approach to adapting to this method. This study investigates how effective ChatGPT can be for learning an introductory programming language through errors. The participants in this study were Foundation Engineering students enrolled in a Python Programming course in the 2023/2024 academic year. Learning programming through errors was the primary approach introduced in lectures. The lessons were divided into two parts to assess students' abilities to learn programming through errors, both with and without the use of ChatGPT. Data was collected from assignments, final exam scores, and student portfolios. The results of this study provide insight into re-designing formative assessment methods for programming courses.

Keywords: formative assessment, programming error, ChatGPT, instructional design

iafor
The International Academic Forum
www.iafor.org

Introduction

Programming learning primarily involves two stages: (a) foundational knowledge and (b) practical understanding. The foundational knowledge includes learning programming concepts, syntax, and semantics, whilst practical understanding is the application of programming concepts and syntax to solve a programming problem. A notable amount of research has been conducted to examine fundamental programming learning issues (Srivatanakul, 2023; Wang et al., 2021), scaffolding programming learning (Shin et al., 2023; Sun et al., 2024; Zhang et al., 2023; Zheng et al., 2022), assessment methods (Riese & Bater, 2022; Thangaraj et al., 2023; Vittorini et al., 2021; Xu et al., 2023;), learners' perceptions and experiences (Kuo & Kuo, 2023; Napalit et al., 2023; Qian & Lehman, 2017), as well as the suitability of programming languages (Brown et al., 2022; Chen et al., 2018; Medeiros et al., 2019). However, no well-defined pedagogy has been found in the literature on teaching programming (Barros, 2022). The choice of programming language, programming paradigm, and organisational approaches are significant unresolved issues in programming education (Luxton-Reilly et al., 2018).

An assignment in programming courses is a standard formative assessment that requires students to design, write, and test code to ensure the program functions without errors. Students' ability to transfer declarative knowledge (understanding programming concepts, syntax, and semantics) and procedural knowledge (applying declarative knowledge to write a program) is assessed using a rubric. In lectures, a worked example, often referred to as a sample program, is commonly used to illustrate programming concepts, syntax, and semantics. In tutorials, students solve programming problems to transfer declarative knowledge into procedural knowledge with hands-on coding practice. The learning experience of writing and testing programs often improves when students can understand and fix programming errors that are not strongly highlighted in the lecture.

Learning programming through errors is a novel approach to teaching and learning programming languages (Tulis et al., 2016; Zhou et al., 2021). Beege et al. (2021) investigated the impact of various errors in worked examples on the learning process. They found that this approach can enhance learning compared to problem-solving tasks, enabling them to emphasize and reflect on the erroneous example to reinforce self-explanation. ChatGPT in programming courses promotes student-centric learning. Though ChatGPT has been heavily criticized for producing irrelevant or incorrect output, it has the potential to solve intermediate-level programming problems (Dunder et al., 2024). Utilising ChatGPT as an AI teaching assistant is a promising approach to adopting this method. Emphasizing learning computer programming through error helps bridge gaps in research on effective programming education, especially in formative assessment.

This research aims to evaluate the effectiveness of ChatGPT to scaffold formative assessment for an introductory programming course.

1. Would ChatGPT be effective in learning an introductory programming language through errors?
2. Would ChatGPT be a useful AI tool for completing a programming assignment?

Literature Review

Programming knowledge and skills are evaluated through formative and summative assessment methods. Formative assessment offers timely feedback to evaluate ongoing learning, while summative assessment evaluates learning outcomes at the end of the course. In-class tests, exams, quizzes, assignments, projects, and portfolios (Renzella & Cain, 2017) are standard assessment methods in programming courses. Developing a framework to assess complex problem-solving skills in computer and engineering education is challenging for educators and researchers (Xu et al., 2023). Nonetheless, technological advancements have enhanced formative evaluation models to meet the needs and motivate modern digital learners.

In general, there is no single definitive solution to a programming problem. The example program serves as a guide for learners to observe while writing their programs. The customary programming teaching model introduces concepts, syntax, semantics, and worked examples in lectures and tutorials. Focusing on programming language errors to enhance conceptual and procedural knowledge is not a standard pedagogical approach. Learners are expected to recognise and rectify mistakes as they practice. However, novice learners often struggle to understand programming errors when they write their programs without examples. Students who struggle or are confused in their learning become frustrated and disengage without proper support (Lodge et al., 2018).

Providing timely feedback supports effective learning (Mojtahedzadeh et al., 2024). In programming courses, feedback models include common mistakes and errors related to concepts and program logic (Haldeman et al., 2018). Programming instructors face challenges in providing effective and timely feedback to students to improve their program design, writing, and testing skills. Once instructors have evaluated the assignments and provided feedback, learners are responsible for reviewing and enhancing their understanding. The instructor cannot accept a modified or corrected program after grades have been released. This approach is practiced widely, encouraging learners to develop their logical thinking and problem-solving skills. Another challenge is that learners may struggle to redesign and rewrite a program if they do not fully comprehend the feedback, especially when there is a lack of ongoing support for learning.

Learning From Error Using ChatGPT

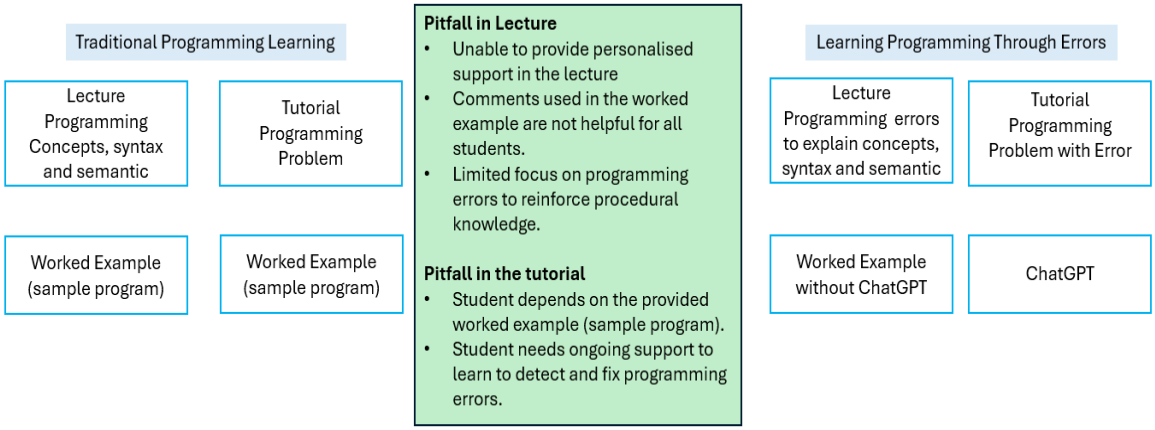
Xia et al. (2023) investigated ChatGPT's ability to generate code for an introductory programming assignment and found that learners struggled with more complex logical reasoning, which could mislead novice programmers. Logical thinking is one of the essential outcomes achieved with hands-on coding activities. A study on the large-scale analysis of ChatGPT's code generation abilities, utilising over 2,000 programming tasks in Java and Python, highlighted its limitations in handling logic errors (Nguyen et al., 2023). The effectiveness of ChatGPT in fixing bugs in code without proper prompting can produce correct output for simple bugs; however, superficial output is found for more complex programs (Li et al., 2023). In programming, different types of errors are related to programming syntax, concepts, and problem-solving logic. Though ChatGPT can explain erroneous code, it must be carefully analysed before it can be used as feedback (Lee & Ko, 2024). A shift in the pedagogical approach in programming courses is crucial when ChatGPT is incorporated into teaching, learning, and assessment.

Pedagogical approaches related to programming errors to promote computer program learning are scarce in the literature (Jerinic, 2014). Program testing and debugging are the most vital stages of programming, and they can be challenging for both beginners and instructors (Kafai et al., 2020). Testing helps to recognize programming errors that are present in the program. Debugging involves finding and fixing programming errors, which requires multiple tests to ensure a program runs without errors (Sun et al., 2024). In tutorial and practical sessions, students practice designing, writing code, and testing programs. However, students encountering programming errors may only debug outside the classroom, as instructors may not closely observe them (Fitzgerald et al., 2008). A learner's ability to acquire programming skills depends on understanding the concepts and syntax and identifying programming errors. Errors can arise from poorly designed programs, coding (concepts, syntax, and semantics), or misconceptions.

Fitzgerald et al. (2008) noted that "good programmers are not necessarily good debuggers, but good debuggers are usually good programmers." They investigated the debugging skills and behaviors of novice programmers at various institutions, suggesting that instructors may need to focus on design, writing, and debugging as one skill rather than regarding each as a distinct skill. The ability to debug a program is essential for developing stronger programming skills, which include writing, testing, and debugging.

In lectures, students gain an understanding of programming concepts, syntax, and semantics. They also work on programming problems in tutorials using worked examples provided by instructors. However, this approach has some problems. Instructors often rely on adding comments to programs to explain concepts, which makes it difficult to provide personalised support to each student. On the other hand, students tend to rely too heavily on worked examples rather than solving problems independently. The alternative approach, learning through errors using ChatGPT, encourages students to identify and correct errors independently. Instead of relying on ready-made explanations, students actively engage with debugging and problem-solving, which helps them understand programming better. This approach makes learning more interactive and independent. The Recursive Reminding Theory supports this approach by demonstrating that learning from errors enhances understanding. By utilising ChatGPT as an AI-assisted tool, students can enhance their skills through trial and error, making programming education more effective and engaging (see Figure 1).

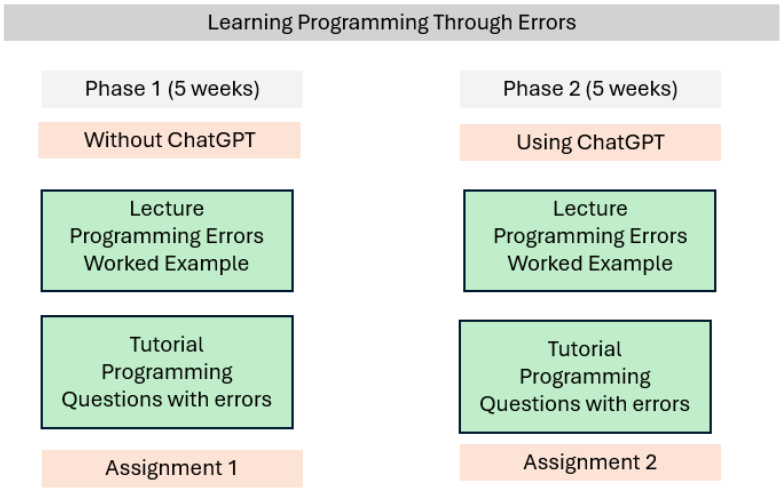
Figure 1: Learning Programming Through Errors



Method

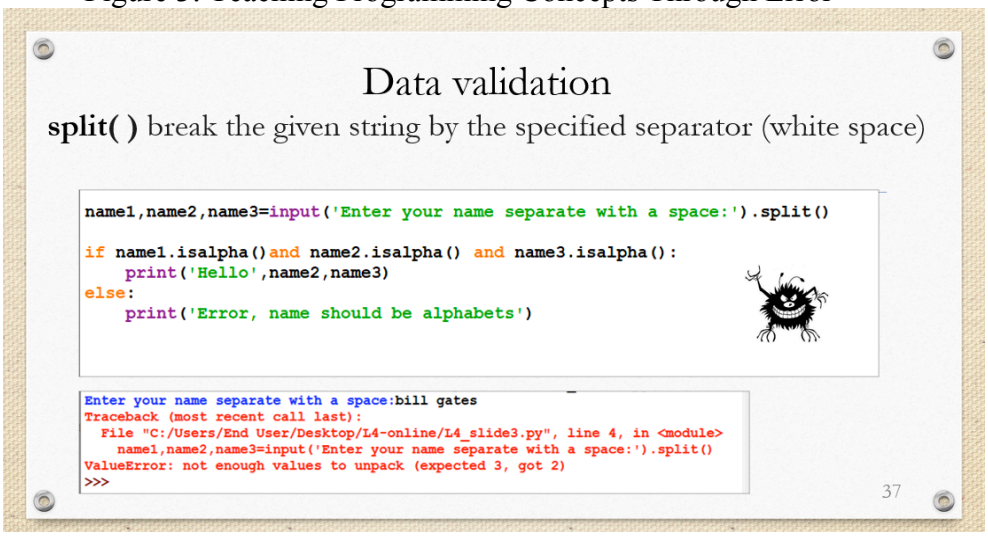
The participants in this study were one hundred and seventy Foundation in Engineering students enrolled in the computer programming module (Python) offered in the academic year 2023/2024. Learning programming through errors was the primary approach introduced in lectures and tutorials. The lesson plans and activities were developed using the TPACK (Technological Pedagogical Content Knowledge) framework (see Figure 2).

Figure 2: Programming Lesson Plan



This study employs a qualitative comparative study approach, where a programming course is conducted over 10 weeks and divided into two phases of 5 weeks each. The objective was to analyze the effectiveness of ChatGPT-assisted learning on students' programming skills by comparing their performance before and after using ChatGPT. In the first 5 weeks (Phase 1), students learn about programming errors through lectures and worked examples (see Figure 3).

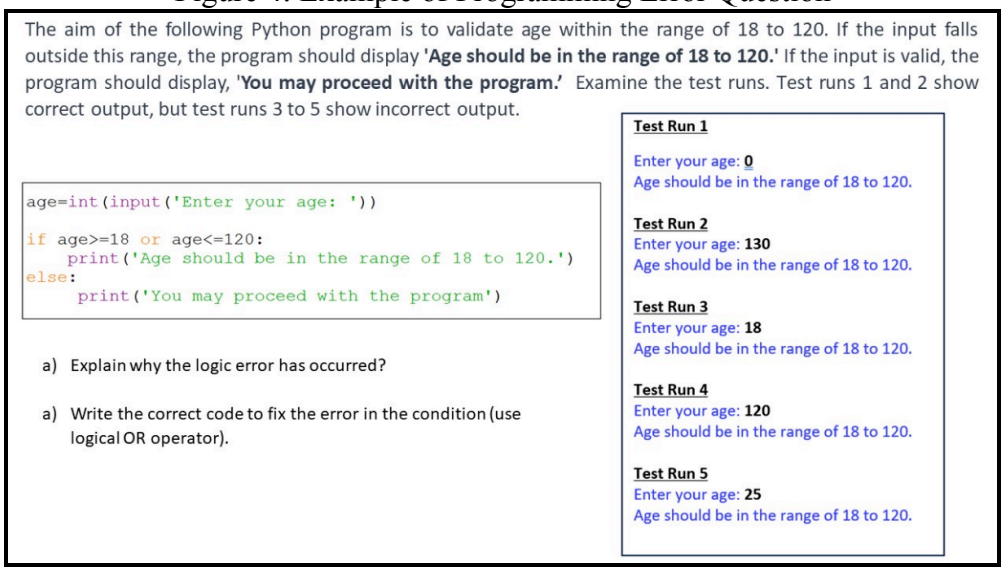
Figure 3: Teaching Programming Concepts Through Error



They also participate in tutorials that involve debugging programs with errors (see Figure 4). At the end of this phase, students complete Assignment 1, which assesses their ability to

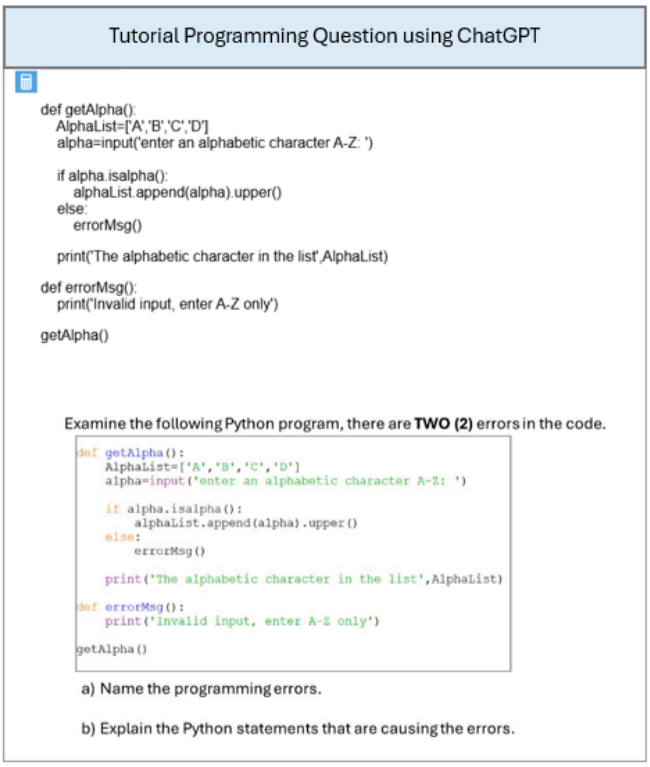
identify and fix errors without using ChatGPT. This phase represents a traditional approach to learning.

Figure 4: Example of Programming Error Question



In the next 5 weeks (Phase 2), the same pedagogical design was employed, with students attending lectures and working on programs with errors in tutorials. However, in this phase, students are encouraged to use ChatGPT as an AI-assisted tool to detect and correct errors. During tutorial sessions, students must identify errors in the program and explain why the error occurred (see Figure 5).

Figure 5: Example of Programming Error Question Using ChatGPT



This learning approach helps students understand the underlying concepts and syntax they were missing or clarifies any misconceptions. At the end of this phase, students complete Assignment 2, which evaluates their performance with ChatGPT-assisted learning.

Inductive Thematic Analysis

A reflective journal is used to gather data, where students document their experiences, challenges, and insights while working on Assignment 2 with ChatGPT, compared to Assignment 1 without ChatGPT. Twenty reflective journals were randomly selected for detailed analysis to address the research questions. Two themes were identified: (a) whether ChatGPT was found helpful for program debugging, and (b) whether it supported acquiring programming skills to analyze the effectiveness of ChatGPT in supporting programming learning through error (RQ1). Another two themes were identified for RQ2: (c) how well ChatGPT assisted in completing assignments, and (d) whether it introduced a challenge.

Results and Discussion

The analysis of the twenty reflective journals was categorized into main themes with supporting sub-themes. Five out of twenty students indicated they did not use ChatGPT to complete assignment 2. Many students found ChatGPT helpful in explaining programming concepts, code structure, and debugging in an easy-to-understand way.

ChatGPT's Effectiveness for Learning Programming Through Errors

Novice programming learners require ongoing support for their programming learning, which is a well-known limitation in typical practical classes or tutorials. Students must master programming writing skills within 10 to 12 weeks of instruction. Often, instructors struggle to support students' learning, making it a significant challenge. ChatGPT can explain the coding in smaller steps when students face difficulties. Several recurring themes identified in this study are also evident in the literature, including enhancing student engagement, making learning more enjoyable, and providing practical learning support.

"Also, its ability to provide instant feedback and suggestions is very nice; it also breaks the code down for you in simple terms, making it easier to understand what's going on if you're confused."

"It provides a detailed explanation of how code works, making learning more interactive and engaging."

The sub-themes that emerged from the main theme are as follows.

Breakdown of Coding. Worked examples used in lectures and tutorials are not compelling because students tend to write code similarly. This approach appears feasible with ChatGPT, which can generate multiple worked examples for a single problem-solving question. Students found ChatGPT helpful in explaining program fragments. Using multiple worked examples can promote program comprehension tasks. However, hands-on activity is essential to support this task. A faded-worked example strategy to promote programming knowledge and skills (Matthews et al., 2019), promising to maximize the benefits of ChatGPT.

Encouraging Exploration.

“ChatGPT doesn’t just provide me with the code; it also explains why things are done in a certain way.”

“Besides, if you are not satisfied with the code generated by ChatGPT, you can request it to regenerate a new code until you are satisfied with the result.”

Students noted that ChatGPT introduced them to new coding techniques they would not have explored otherwise. This learning approach is an important first step for them to learn, but they must also think and write code based on their own logical thinking. This approach can help develop essential logical thinking. A proper guideline on how students should use ChatGPT for learning programming is crucial. Instructors must provide a guide for students to understand the required knowledge and skills, as well as how to utilise ChatGPT to achieve them. A declaration on using AI for programming assignments would pose a challenge and may not effectively support academic integrity.

Trial and Error Approach. Students used ChatGPT iteratively, testing code, modifying it, and refining their understanding. The Recursive Reminding Theory supports this approach, which involves identifying a shift in programming pedagogy to incorporate AI chatbots, such as ChatGPT. A trial-and-error approach is a subtle learning process that enables novice learners to develop a strong cognitive understanding. This approach paves the way for redesigning the assignment into a learning portfolio, allowing students to utilise ChatGPT; however, grading and scoring students' understanding and skills warrant further investigation.

“ChatGPT helped me understand programming better by explaining errors in simple terms.”

“ChatGPT gave me a code snippet, I modified it and learned from the changes.”

“It helped me fix errors quickly, allowing me to focus on improving my programming logic.”

I experimented with different approaches by modifying ChatGPT’s suggestions, which helped me learn more about the topic.

The Usefulness of ChatGPT in Completing Programming Assignments

This theme analysed students’ reflections on whether ChatGPT was useful for completing the assignment task and whether it supported or hindered their learning. The sub-themes that emerged from the main theme are as follows.

Learner’s Awareness of Using ChatGPT. Students' excitement and engagement with ChatGPT reflected a highly positive response. However, they are also aware of its proper use for learning programming. This awareness was observed when students chose not to use ChatGPT to complete the programming assignment. Incorporating ChatGPT into teaching and learning contexts with proper guidelines and learning activities enables them to learn how to use it effectively.

“I believe students could exploit this tool to easily complete assignments without learning anything in the process. In conclusion, ChatGPT is an excellent tool for learning; however, I

believe students should use it appropriately in an educational manner to maximize its benefits."

"Due to this reason, it may become difficult in the future to add new code, and we will rely more on ChatGPT."

Combining ChatGPT With Manual Debugging. Students emphasised that ChatGPT should be used alongside lecture notes and textbooks, not as a replacement. They found ChatGPT helpful as an initial guide, but they still preferred to manually debug programs. This reflection highlights the importance of guidelines for utilising ChatGPT in a classroom setting and for assessment purposes. The initial discussion in the literature on ChatGPT was that it could pose a challenge in assessing students. This claim is acceptable if educators implement the typical programming assignment, accompanied by a declaration of how students utilised ChatGPT to complete their assignments.

"Overall, I think ChatGPT is a good learning tool, but it will not be as helpful as a teacher or even the notes, because it doesn't know you are a new learner and will just give out everything it knows, and sometimes some beginners will end up messing up everything with those codes."

Conclusion and Recommendation

The analysis of twenty reflective journals revealed that ChatGPT facilitates programming learning by providing guidance, simplifying complex concepts, and enhancing confidence. Students used it as a "coding buddy" for real-time support, enhancing their learning through trial and Error, exploration, and time-saving debugging. It also helped explain code concepts and introduce alternative approaches, encouraging experimentation with coding. However, students regard ChatGPT as a supplementary tool alongside traditional learning methods for optimal results.

The following suggestions are for assignment programming tasks using ChatGPT in an introductory course.

- Trial and Error – promotes programming skills by detecting and fixing code errors.
- Faded Worked Example – promotes programming skills through a learning cycle that includes code writing and testing.
- Modifying Program – promotes programming skills through logical thinking approaches.

Acknowledgements

This study was supported by T&L Seed Grant (grant number: UNLA0001).

Declaration

During the preparation of this paper, the author utilised Grammarly and ChatGPT to verify grammar and enhance clarity and language.

References

Barros, J. P. (2022). *Assessment of computer programming courses: A short guide for the undecided teacher*. In *Proceedings of the International Conference on Computer Supported Education (CSEDU)* (Vol. 2, pp. 549–554). https://doi.org/10.5220/0011095800003182

Beege, M., Schneider, S., Nebel, S., Zimm, J., Windisch, S., & Rey, G. D. (2021). Learning programming from erroneous worked-examples: Which type of error is beneficial for learning? *Learning and Instruction, 74*, 101497. https://doi.org/10.1016/j.learninstruc.2021.101497

Brown, N. C. C., Weill-Tessier, P., Sekula, M., Costache, A.-L., & Kolling, M. (2022). Novice use of the Java programming language. *ACM Transactions on Computing Education, 23*(1). https://doi.org/10.1145/3551393

Chen, C., Haduong, P., Brennan, K., Sonnert, G., & Sadler, P. (2018). The effects of first programming language on college students' computing attitude and achievement: A comparison of graphical and textual languages. *Computer Science Education, 29*(1), 23–48. https://doi.org/10.1080/08993408.2018.1547564

Dunder, N., Lundborg, S., Wong, J., & Viberg, O. (2024). Kattis vs ChatGPT: Assessment and Evaluation of Programming Tasks in the Age of Artificial Intelligence. *Proceedings of the 14th Learning Analytics and Knowledge Conference*, 821–827. https://doi.org/10.1145/3636555.3636882

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education, 18*(2), 93–116. https://doi.org/10.1080/08993400802114508

Haldeman, G., Tjang, A., Babes-Vroman, M., Bartos, S., Shah, J., Yucht, D., & Nguyen, T. D. (2018). Providing meaningful feedback for autograding of programming assignments. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, 278–283. https://doi.org/10.1145/3159450.3159502

Jerinic, L. (2014). *Teaching introductory programming: Agent-based approach with pedagogical patterns for learning by mistake*. International Journal of Advanced Computer Science and Applications, 5(6), 113–119. https://doi.org/10.14569/IJACSA.2014.050617

Kafai, Y. B., Evangelou, D., Fields, D. A., & Danish, J. A. (2020). *Turning bugs into learning opportunities: Understanding debugging processes, perspectives, and pedagogies*. In M. Gresalfi & I. S. Horn (Eds.), The Interdisciplinarity of the Learning Sciences, 14th International Conference of the Learning Sciences (ICLS) 2020 (pp. 374–381).

Kuo, Y. T., & Kuo, Y. C. (2023). African American students' academic and web programming self-efficacy, learning performance, and perceptions towards computer programming in web design courses. *Education Sciences, 13*(12), Article 1236. https://doi.org/10.3390/educsci13121236

Li, S., Zhou, X., & Liu, Y. (2023). *A critical review of large language model on software engineering: An example from ChatGPT and automated program repair*. arXiv. https://arxiv.org/abs/2310.08879

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). *Introductory programming: A systematic literature review*. In *Proceedings of the Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE)* (pp. 55–106). ACM. https://doi.org/10.1145/3293881.3295779

Lodge, J. M., Kennedy, G., Lockyer, L., Arguel, A., & Pachman, M. (2018). Understanding difficulties and resulting confusion in learning: An integrative review. *Frontiers in Education, 3*, 49. https://doi.org/10.3389/feduc.2018.00049

Matthews, R., Bavani, R., & Yong, S. T. (2019). Digital worked example: An experiment on strategies to enhance computer programming skills. *International Journal of Recent Technology and Engineering (IJRTE)*, *8*(3S2), 10-15. https://www.ijrte.org/wp-content/uploads/papers/v8i3S2/C11261083S219.pdf

Medeiros, R. P., Ramalho, L., & Falcão, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education, 62*(2), 77–97. https://doi.org/10.1109/TE.2018.2864133

Mojtahedzadeh, R., Hasanvand, S., Mohammadi, A., Malmir, S., & Vatankhah, M. (2024). Students' experience of interpersonal interactions quality in e-Learning: A qualitative research. *PLOS ONE, 19*(3), e0298079. https://doi.org/10.1371/journal.pone.0298079

Napalit, F., Tanyag, B., So, C. L., Sy, C., & Pedro, J. R. S. (2023). Examining student experiences: Challenges and perception in computer programming. *International Journal of Research Studies in Education, 12*(8). https://doi.org/10.5861/ijrse.2023.71

Nguyen, H. T., Zhao, Y., Xia, X., & Lo, D. (2023). *Refining ChatGPT-generated code: Characterizing and mitigating code quality issues*. arXiv. https://arxiv.org/abs/2307.12596

Qian, Y., & Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education, 18*(1). https://doi.org/10.1145/3077618

Renzella, J., & Cain, A. (2017). Supporting better formative feedback in task-oriented portfolio assessment. *Proceedings of the 2017 IEEE 6th International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 360–367. https://doi.org/10.1109/TALE.2017.8252362

Riese, E., & Bater, O. (2022). A qualitative study of experienced course coordinators' perspectives on assessment in introductory programming courses for non-CS majors. *ACM Transactions on Computing Education, 22*(4), Article 45. https://doi.org/10.1145/3517134

Shin, Y., Jung, J., Zumbach, J., & Yi, E. (2023). The effects of worked-out example and metacognitive scaffolding on problem-solving programming. *Journal of Educational Computing Research, 61*(6), 1312–1331. https://doi.org/10.1177/07356331231174454

Srivatanakul, T. (2023). Emerging from the pandemic: Instructor reflections and students' perceptions on an introductory programming course in blended learning. *Education and Information Technologies, 28*, 5673–5695. https://doi.org/10.1007/s10639-022-11328-6

Sun, D., Looi, C. K., Li, Y., Zhu, C., & Cheng, M. (2024). Block-based versus text-based programming: A comparison of learners' programming behaviors, computational thinking skills and attitudes toward programming. *Educational Technology Research and Development, 72*(2), 1067–1089. https://doi.org/10.1007/s11423-023-10328-8

Thangaraj, J., Ward, M., & O'Riordan, F. (2023). A systematic review of formative assessment to support students learning computer programming. *OpenAccess Series in Informatics, 112*. https://doi.org/10.4230/OASIcs.ICPEC.2023.7

Tulis, M., Steuer, G., & Dresel, M. (2016). Learning from errors: A model of individual processes. *Educational Psychologist, 51*(1), 25–39. https://doi.org/10.1080/00461520.2015.1122015

Vittorini, P., Menini, S., & Tonelli, S. (2021). An AI-based system for formative and summative assessment in data science courses. *International Journal of Artificial Intelligence in Education, 31*, 159–185. https://doi.org/10.1007/s40593-020-00230-2

Wang, W., Kwatra, A., Skripchuk, J., Gomes, N., Milliken, A., Martens, C., Barnes, T., & Price, T. (2021). *Learning barriers when using code examples in open-ended programming*. In Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE). https://doi.org/10.1145/3430665

Xia, Y., Huang, Z., Zhang, Y., Zheng, L., & Zhong, H. (2023). *Assessing the promise and pitfalls of ChatGPT for automated CS1-driven code generation*. Proceedings of the 17th International Conference on Educational Data Mining (EDM 2024). https://educationaldatamining.org/edm2024/proceedings/2024.EDM-long-papers.7/2024.EDM-long-papers.7.pdf

Xu, X., Shen, W., Islam, A. Y. M. A., & Zhou, Y. (2023). A whole learning process-oriented formative assessment framework to cultivate complex skills. *Humanities and Social Sciences Communications, 10*(1). https://doi.org/10.1057/s41599-023-02200-0

Zhang, J. H., Meng, B., Zou, L. C., Zhu, Y., & Hwang, G. J. (2023). Progressive flowchart development scaffolding to improve university students' computational thinking and programming self-efficacy. *Interactive Learning Environments, 31*(6), 3792–3809. https://doi.org/10.1080/10494820.2021.1943687

Zheng, L., Zhen, Y., Niu, J., & Zhong, L. (2022). An exploratory study on fade-in versus fade-out scaffolding for novice programmers in online collaborative programming settings. *Journal of Computing in Higher Education, 34*(3), 489–516. https://doi.org/10.1007/s12528-021-09307-w

Zhou, Z., Wang, S., & Qian, Y. (2021). Learning from errors: Exploring the effectiveness of enhanced error messages in learning to program. *Frontiers in Psychology, 12*, Article 768962. https://doi.org/10.3389/fpsyg.2021.768962