Caliphate Journal of Science & Technology (CaJoST)

**ISSN:** 2705-313X (**PRINT**); 2705-3121 (**ONLINE**) **Review Article**

**Open Access Journal available at: https://cajost.com.ng/index.php/files and https://www.ajol.info/index.php/cajost/index**



**DOI: https://dx.doi.org/10.4314/cajost.v7i2.4**

**Article Info**

Received: 8th January 2025

Revised: 27th April 2025

Accepted: 3rd June 2025

# The Imparts and Pedagogical Challenges of AI-Generated Codes on Computer Programming Education at Introductory Level: Curriculum Design Recommendations

Anthony Kwubeghari

Depart of Computer Engineering, Enugu State University of Science and Technology, Enugu State, Nigeria.

***Corresponding author's email:**

kwubeghari.anthony@esut.edu.ng.

**Cite this:** *CaJoST*, 2025, 2, 221-227

The variety of generative artificial intelligence (AI) applications has led to a sharp rise in interest in their application in Learning programming and the process of writing programs. From the reviewed literature, researchers are yet to find best ways to integrate AI-generated programs in the pedagogy of teaching and learning programming especially in the introductory level where understanding the fundamentals is crucial. Researchers and teachers must prioritize academic integrity and fair assessment of students' programming skills in the era of Large Language Models (LLM) over viewing AI as a threat. How are computing educators planning to modify their courses in response to the increasing proliferation of AI code generation and explanation tools, considering the current state of affairs and looking into the future when AI capabilities are likely to improve? This paper harnessed the experiences of using LLM (AI code generating tools) for teaching and learning computer programming at the introductory level based on the perspective of the students, the teachers and the researchers. This paper also gathered various perspectives on the above question based on the existing works, position papers and workshops by the computing community. Recommendations on the possible best ways to integrate LLM or AI code-generating tools were made. By providing recommendations for teachers and curriculum developers on how to successfully integrate these tools into teaching and learning programming, this paper will advance knowledge of the possible advantages and consequences of using them in educational environments.

**Keywords:** Large Language Model, AI Code generating tools, Introductory Programming Education, Novice Programming, Artificial Intelligence

## 1. Introduction

Using Artificial Intelligence (AI) and Machine Learning (ML) powered software tools to write a computer code is becoming normal. A programmer or a student gives the AI tool a specification of the task to be performed by the code instead of manually typing each line of code. With the aid of advanced Large Language Model (LLM) and Generative AI techniques, the AI tools automatically suggest or generate code based on the required functionality specified by the user [1].

In the past ten years, there has been a noticeable advancement in programming education. While it was once thought that only a select few people should be proficient in programming, it is now a prerequisite for many different professions. These days, programming is a vital tool for coming up with creative solutions to challenging issues in a variety of industries, such as finance, healthcare, and transportation. For this reason, learning programming is essential for anyone who wishes to succeed in a variety of fields, particularly business [2].

For educators and students taking introductory programming classes and associated courses, AI-generated code offers both opportunities and obstacles. The quick availability and accessibility of these technologies raises the possibility that instructors will be unprepared for the profound effects of AI-generated code on teaching methods. As a result, we must immediately evaluate our teaching methods in light of this new technology [3].

Code generation technologies are developing rapidly and it is hard to ignore their prevalence among aids for learning within computer science courses. They have a great potential to bring benefits in educational realm if being approached and used in a smart way. Sheard, et al [4] pointed out that there may be some kind of backlash if the teachers don't teach the students how to use and integrate AI tools in their learning because students will definitely use

  

them and the workplaces will require them (students) to use such tools in future.

It is unclear how teachers should adjust to this new technology that our students can access for free [5]. As asked by Lau and Guo [6]: How are computing educators planning to modify their courses in response to the increasing proliferation of AI code generation and explanation tools, taking into account the current state of affairs and projecting into the future when AI capabilities are likely to improve? The aim of this paper is to gather various view points on the above questions based on the existing works, position papers and workshops by the computing community and then make recommendations.

### 1.1 Motivation.

Returning to the university setting after more than 27 years away and finding that the introductory level of computer programming education was still taught in the same way, I was inspired to look for a better teaching method in this era of LLM. The curriculum developers and teachers act as though there is no such thing as a large language model (in general) or AI code generation (in particular). There should be established or de facto best practices for integrating AI-generating tools into introductory computer programming education in order to preserve academic integrity.

## 2. Review of Relevant Literatures on Code Generation Tools in Introductory Computer Programming Education

Sarsa, et al [7] foresee that the features of generative models for computing education practice and research will only improve over time with the continuing evolution of these models. The work also infer that modern machine learning models like OpenAI Codex offer many benefits for programming course designers, although potential challenges should not be ignored.

Poldrack, et al [8] agree with the potential power of AI code generation tool but strongly suggests through their finding that human validation is needed to guarantee high accuracy.

According to Lau and Guo [6], many intended to take prompt action to deter cheating in the short term. In addition, opinions are divided over how to use these AI tools in the long run. Some wanted to outlaw them and keep teaching programming fundamentals, while others are planning to incorporate them into classes to help students get ready for the workforce.

From the research of Yilmaz and Yilmaz [2], with appropriate measures to mitigate the limitations, it would be useful to integrate generative AI tools in programming courses considering the benefits they offer in programming teaching.

Research on introductory programming sequences in computing education highlights the potential of open-source AI-driven code generation tools. However, educators must quickly identify opportunities and challenges to ensure the effectiveness and coordination of these tools in shaping educational outcomes [3].

Denny, et al [9] found that students value these tools because they can provide prompt and engaging assistance when needed. Furthermore, they expressed a strong preference for elements that allowed them to continue being independent while learning, such as scaffolding, which help the users solve problems by guiding them through the process rather than giving them the solutions directly.

Educators are re-evaluating learning objectives and seeking innovative teaching strategies to adapt to changes in course and curriculum. There are no research-based pedagogies or best practices for integrating these tools with courses. The future of programming skills may shift from providing students with basic understanding to using AI tools for necessary programming. This shift could lead to computer science becoming a subject with extensive knowledge, with few students needing real programming skills [4].

Introductory courses aim to teach students to read and understand code, which is increasingly important due to LLM-based tools' ease of generation. Efficient prompting for LLMs is crucial for end-user programming activities [10].

Inspired by the findings from surveys that demonstrate that the learners on Degree Apprenticeship Programme have a great curiosity in learning and exploiting emerging AI technology, Petrovska, et al [11] explores how Generative AI can be integrated into software development education.

Doughty et al [12] studied the use of LLMs in Python programming courses, finding that GPT-4 can generate clear, accurate, and high-quality multiple-choice questions (MCQs) with alignment with learning objectives, providing valuable insights for educators using advanced generative models.

Pan et al [13] study reveals AI Content detectors' ineffectiveness in distinguishing between human-written and AI-generated code, raising concerns about potential exploitation of this flaw by LLM users.

Logacheva, et al [14] demonstrate that AI-generated programming problems can be a valuable addition to introductory programming courses, because students get access to a limitless supply of practice material to cater for their individual learning objectives and areas of interest. Denny, et al [15] and Logacheva, et al [14] agreed that LLM tools in introductory programming courses are viable learning resources for students. According to Becker et al [3], when using generative AI tools, educators and curriculum planners must carefully consider how much emphasis should be placed on writing and implementing code as opposed to specifying, reading, comprehending, tracing, testing, and debugging code.

  

. From the exploratory survey of Liang, et al [16], developers are more inclined to use AI programming assistants due to their capacity for autocompletion, speeding up programming tasks, and helping them remember syntax. These features outweigh the developers' need for assistance in coming up with ideas for solving problems. Furthermore, there is a gap between the needs of developers and the output of the tools, even with the high performance of cutting-edge AI programming assistants. One example of this is the need to account for non-functional requirements.

Generative AI is significantly impacting programming education, affecting assessment, classroom practices, and learning goals. It raises moral and ethical issues, potentially changing computing and influencing students' and teachers' majors. The success of generative AI depends on its benefits for students and teachers, and educators must stay updated on its capabilities and minimize risks [17].

Human instructors provide timely and quality feedback in large undergraduate programming courses, but scaling their support is challenging due to their limited availability. Students often prioritize assignment completion over long-term learning and conceptual mastery, relying on Artificial Intelligence for assistance [9].

AI can transform education by enabling adaptive and transformative landscapes. Hence, balancing challenges and AI's potential for enriched, innovative education is crucial for the future [18].

## 3. Method

The approaches used in this work are in the following order.

***Defining the scope***: The title suggests the scope of the work. Then the criteria for selecting papers are as explained later in this section. The scholarly sources identified for pertinent materials are Web of Science, Google Scholar, MDPI, and Scopus.

***Searching for the Research Papers***: Several pertinent studies that have conducted research in this field were located using keywords like "code generation tools," "introductory programming education," and "higher education," as well as any combination of these terms. The obtained articles are chosen from conference proceedings and journals. To avoid leaving out any relevant work, the references of earlier works were also consulted; and the synonyms of the searched terms were further considered.

***Screening the papers*:** Usman et al [19] quality assessment criteria for the selection of articles were applied. Such criteria involve asking important questions like: Was the abstract well-defined? Were the introduction, literature review, and pertinent works fully explained? Were the methodology and outcome evaluation sections clearly stated? We choose the articles published in prestigious journals and conference proceedings based on the Web of Science Core Collection after eliminating duplicate and unnecessary articles.

***Analyze****:* An essential component of a systematic review is assessing the scientific merit of the shortlisted articles. This assessment is conducted using the progressive assessment procedure outlined in Rowe et al [20]. Research concepts, study designs, data collection methods, data analysis techniques, discussions, and results are all included in this process [21].

***Presentation:*** In the last stage, the article, "Pedagogical Challenges of Ai-Generated Codes on Computer Programming Education at Introductory Level: Curriculum Design Recommendations**"** was structured. The content of the article was arranged and presented for easy reading and understanding.

## 4. Results and Discussion

This section discusses the imparts of the use of Ai-code generating tools in teaching and learning programming in three perspectives: Teachers' teaching experiences, students learning experiences at the introductory stage and researchers' experiences. The possible advantages and the corresponding pitfalls as identified by many researchers ([22], [7], [3], [17], [8], [2], [6], [30]) are discussed in this section.

### 4.1 The Imparts of Ai-Generated Code on Teachers of Introductory Programming Education

The advantages of these tools in helping teachers create a variety of programming exercises and provide thorough explanations were emphasized by Sarsa et al [7]. Code generation tools, according to Becker et al [3], open doors for innovative pedagogical strategies. Not only can these tools generate exercises, but they can also provide more thorough explanations of the generated exercises than teachers could. Teachers can save time and effort and give their students access to useful resources by using these technologies. These features of code generation tools give teachers access to a wide range of materials that can significantly improve students' educational experiences by giving them thorough support to succeed in programming.

There are serious concerns whether teachers are ready to handle the influence of these tools on educational practices. Becker, et al [3] opined that educators will lose out on opportunities to shape what opportunities arise and what challenges continue if the effectiveness and proliferation of these tools continue to advance quickly without prompt, deliberate, and coordinated efforts.

### 4.2 The Imparts of Ai-Generated Code on Students of Introductory Programming Education

Code generation tools can assist students with a variety of tasks related to programming, such as

 

fixing bugs in programming, software engineering projects, creating exercises that provide clear explanations and examples of programming constructs and algorithmic patterns, mapping problems to solutions, and exam preparation [22].
Researchers caution against relying too much on the tool in favour of learning how to code independently. These possible pitfalls show how important it is to use caution and critical thinking when utilizing these kinds of tools, particularly when teaching students. It's crucial to strike a balance between using these tools and helping students learn the fundamentals of programming and problem-solving in order to guarantee their learning progress and theoretical comprehension [22].

### 4.3 The Imparts of Ai-Generated Code on Researchers

Large Language model provides researchers with a number of benefits. It can aid in the research writing process quite well. In its most basic form, it can enhance writing by highlighting typographical errors and fixing them, offer sophisticated vocabulary, and suggest improvement techniques. This frees up more time for experimentation and implementation on the part of the researchers. In order to aid researchers in understanding published work on a given subject, the model can also summarize the work. By examining a particular subject, it can also offer hints and research topics [23].

### 4.4 Challenges of Ai-Generated Codes to Introductory Computer Programming Education

Since LLMs are still an emerging technology, it's crucial to be aware of the following application-related difficulties:

- ✓ ***Academic Integrity*** – Prather, et al [24] summarized the following common practices that may undermine academic integrity through the use of AI code generation tools: plagiarism, unauthorized resources, collusion (collaboration among students without official approval but submitted as individual work), contract cheating (requesting another person to produce a work but submitted as your work), and falsification (inventing data or results; LLM can produce incorrect result).
- ✓ ***Incorrectness*** - Errors in the analysis or generation of code can result in problems and defects in program. Consistent testing and validation procedures are needed to confirm the accuracy of content produced by LLM [25].
- ✓ ***Rapid Evolution*** - LLM technology is still evolving, and these models' efficiency varies with time [25]. Teachers and students need to keep up with these developments and modify their teaching methods accordingly.
- ✓ ***Intellectual Property*** - The huge training data of LLMs may contain copyrighted materials that may lead to legal issues when using these tools for code generation [25].
- ✓ ***Lack of Essential Skills*** - Users of LLMs must create pertinent prompts and verify their outcomes. Because many users are not yet proficient enough with prompt patterns to interact with these models effectively, the usability of LLMs depends on user expertise [25]. Students' prompts are not up to par when they lack the necessary skills to fully utilize Generative AI tools. Consequently, it is imperative to encourage students to develop the capacity to utilize AI tools efficiently and comprehend how to ask questions and evaluate answers [29].
- ✓ ***Security*** – Pearce, et al [30] found that AI code generators are vulnerable to security risk both during training and generation. Therefore users should be alert.
- ✓ ***Human Oversight*** - Given the current state of LLM maturity, human oversights are maintained during usage to ensure people make informed decisions and ensure ethical compliance. The Artificial Intelligence Risk Management Framework (AI RMF 1.0) by National Institute of Standards and Technology of US Department of Commerce also provides important context for proper use of generative AI tools [28].
- ✓ ***Over Reliance on Generative AI Tools*** - critical thinking and problem-solving skills can be limited by the ease with which generative tools get answers, problem-solving strategies, and scientific text generation [23]. Although the most recent LLM releases are frequently very powerful, they are not a universally applicable solution to every student's problem. Teachers, students, organizations, and institutions all need to know when to use LLMs and what kinds of problems work best for them. In particular, users need to be aware of the risks and ways to mitigate them in order to apply LLMs effectively in the modern world.
- ✓ ***Adversarial Attack*** - Any machine learning model has the potential to receive adversarial machine learning attack. Attackers can tamper with the data used to train the AI model or use inputs that appear normal but have hidden features that can generate confusion for the AI system.

## 5. Recommendations

Based on detail study of the concerns of the stakeholders in computer programming education, especially introductory level, we offer the following recommendations.

i. ***Teach Students About Proper and Improper Use****:* This includes the rationale behind the choices you made regarding the learning objectives. A concise statement on your syllabus should include the following:

  

When using AI is appropriate (provide a list of specific tasks and activities); When using AI is inappropriate (provide a list of specific tasks and activities); How to apply AI (idea generation, paraphrasing, summarizing, etc.); How to properly cite the use of AI in assignments (footnotes, APA, supporting documents, etc.); Penalties for breaking the AI policy of the course or the assignment (such as an F on the assignment) [29].

ii. ***Code comprehension and critical thinking skills*:** In addition to understanding the fundamentals, emphases should be placed more on code comprehension and critical thinking skills from curriculum design stage to classroom teaching. This will enable the students to develop the competency to read and understand code thoroughly in this era of LLM.

iii. ***Live coding Assessment**:* This approach enables students to solve problems during Lab or practical classes. This assessment method removes the danger of plagiarism and any AI assistance.

iv. ***Test-driven assessment before coding [18]:*** This approach enables students to write test before coding, hence encouraging critical thinking in solving problem. This approach enables students to first of all tackle smaller problems, understand the domain, ensures code correctness and nurtures analytical mindset for solving problems. At the end, over reliance on AI assistance is reduced.

v. ***Use AI Code Generating Tools as Additional Learning Materials**:* Use AI to produce several code snippet examples that highlight particular ideas. This gives students a variety of viewpoints for approaching problem-solving. Students can also practice various approaches to the same problem by using AI to generate variations of programming exercises.

vi. ***Use AI Code Generating Tools as Teaching Aids for Debugging Exercises*:** Employ AI to produce code that contains deliberate bugs or logical errors. Then, students can practice debugging—an important competency for programmers. Also give students a copy of AI-generated code to review, and encourage them to point out any problems or possible improvements.

vii. ***Use AI Code Generating Tools as Collaborative Learning Resources***: Attach students to groups to examine and enhance code generated by artificial intelligence. This encourages critical thinking and collaboration. AI-generated code can be use to initiate discussions regarding coding conventions, best practices, and problem-solving approaches in the classroom.

viii. ***Use AI Code Generating Tools to Aid Creativity*:** To help students learn coding in an enjoyable and interesting way, encourage them to use AI to generate ideas or templates for creative coding projects.

ix. ***Teachers Must be Versatile in AI Code Generating Tools**:* The era of Large Language Model (LLM) has come to stay. Students will use it whether teachers encourage it or not. The only way to help students is to have expert knowledge of LLM.

## 6. Conclusions

Identifying quick fixes to the current state of affairs is important as the academic community struggles with these new challenges. While researchers try to find ways to integrate AI-generated programs in the pedagogy, educators and institutions need to adapt to the changing environment. They need to think about ways to maintain academic integrity and guarantee a rigorous and equitable assessment of students' programming skills. Rather than just viewing AI as a danger to academic integrity, we should consider its potential and power to improve the efficiency of assessments, teaching and learning. Just as intelligent Integrated Development Environment (IDE) with many features for easy coding are acceptable, Large Language Models for code generation have come to stay. The issue now is to find the best way to incorporate it in teaching and learning programming at the introductory level.

## Conflict of Interest

The authors declare no conflict of interest.

## References

[1] SonarSource website [online]. (2023).Available:https://www.sonarsource.com/learn/ai-code-generation/

[2] Yilmaz, R., & Yilmaz, F. G. K. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. Computers in Human Behavior: Artificial Humans, 1(2), 100005. https://doi.org/10.1016/j.chbah.2023.100005

[3] Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., & Santos, E. A. (2023a). Programming Is Hard – Or at Least It Used to Be. https://doi.org/10.1145/3545945.3569759

[4] Sheard, J., Denny, P., Hellas, A., Leinonen, J., Malmi, L., & Simon. (2024, March). Instructor Perceptions of AI Code Generation Tools-A Multi-Institutional Interview Study. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education

 

(1), 1223-1229. https://doi.org/10.1145/3626252.3630880

[5] Finnie-Ansley, J., Denny, P., Luxton-Reilly, A., Santos, E. A., Prather, J., & Becker, B. A. (2023, January). My ai wants to know if this will be on the exam: Testing openai's codex on cs2 programming exercises. In Proceedings of the 25th Australasian Computing Education Conference, 97-104. https://doi.org/10.1145/3576123.3576134

[6] Lau, S., & Guo, P. (2023). From" Ban it till we understand it" to" Resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In Proceedings of the 2023 ACM Conference on International Computing Education Research, (1), 106-121. https://doi.org/10.1145/3568813.3600138

[7] Sarsa, S., Denny, P., Hellas, A., & Leinonen, J. (2022, August). Automatic generation of programming exercises and code explanations using large language models. In Proceedings of the 2022 ACM Conference on International Computing Education Research (1), 27-43. https://doi.org/10.1145/3501385.3543957

[8] Poldrack, R. A., Lu, T., & Beguš, G. (2023). AI-assisted coding: Experiments with GPT-4. https://doi.org/10.48550/arXiv.2304.13187.

[9] Denny, P., MacNeil, S., Savelka, J., Porter, L., & Luxton-Reilly, A. (2024a). Desirable Characteristics for AI Teaching Assistants in Programming Education. In Proceedings of the 2024 on Innovation and Technology in Computer Science Education, (1), 408-414. https://doi.org/10.1145/3649217.3653574

[10] Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024b). Prompt Problems: A new programming exercise for the generative AI era. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education, (1), 296-302. https://doi.org/10.48550/arXiv.2311.05943

[11] Petrovska, O., Clift, L., Moller, F., & Pearsall, R. (2024). Incorporating Generative AI into Software Development Education. In Proceedings of the 8th Conference on Computing Education Practice, 37-40. https://doi.org/10.1145/3633053.3633057

[12] Doughty, J., Wan, Z., Bompelli, A., Qayum, J., Wang, T., Zhang, J., ... & Sakr, M. (2024). A comparative study of AI-generated (GPT-4) and human-crafted MCQs in programming education. In Proceedings of the 26th Australasian Computing Education Conference, 114-123. https://doi.org/10.1145/3636243.3636256

[13] Pan, W. H., Chok, M. J., Wong, J. L. S., Shin, Y. X., Poon, Y. S., Yang, Z., ... & Lim, M. K. (2024). Assessing AI Detectors in Identifying AI-Generated Code: Implications for Education. In Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training, 1-11. https://doi.org/10.48550/arXiv.2401.03676

[14] Logacheva, E., Hellas, A., Prather, J., Sarsa, S., & Leinonen, J. (2024). Evaluating Contextually Personalized Programming Exercises Created with Generative AI. https://doi.org/10.48550/arXiv.2407.11994.

[15] Denny, P., Khosravi, H., Hellas, A., Leinonen, J., & Sarsa, S. (2023). Can we trust AI-generated educational content? comparative analysis of human and AI-generated learning resources. https://doi.org/10.48550/arXiv.2306.10509.

[16] Liang, J. T., Yang, C., & Myers, B. A. (2024, February). A large-scale survey on the usability of ai programming assistants: Successes and challenges. In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, 1-13. https://doi.org/10.48550/arXiv.2303.17125

[17] Becker, B. A., Craig, M., Denny, P., Keuning, H., Kiesler, N., Leinonen, J., ... & QUILLE, K. (2023b). Generative AI in Introductory Programming.

[18] Tang, C. M., Ng, V. S. C., Leung, H. M. F., & Yuen, J. C. H. (2024). AI-Generated Programming Solutions: Impacts on academic integrity and good practices. In CSEDU (2), 478-485. https://doi.org/10.5220/0012563600003693

[19] Usman I., Mashaim F., Sajid M., Muhammad F. Z., Shahid H., Fabiano P. (2024). Insider threat mitigation: Systematic literature review. Ain Shams Engineering Journal, 15(12), 103068, ISSN 2090-4479, https://doi.org/10.1016/j.asej.2024.103068. https://www.sciencedirect.com/science/article/pii/S209044792400443X

[20] Rowe, M., Frantz, J., & Bozalek, V. (2012). The role of blended learning in the clinical education of healthcare students: A systematic review. Medical Teacher, 34(4), e216–e221. https://doi.org/10.3109/0142159x.2012.642831

[21] Heyvaert, M., Hannes, K., Maes, B., & Onghena, P. (2013). Critical Appraisal of Mixed Methods Studies. Journal of Mixed Methods Research, 7(4), 302-327. https://doi.org/10.1177/1558689813479449 (Original work published 2013)

[22] Shynkar, A. (2023). Investigating the influence of code generating technologies on learning process of novice programmers in higher education computer science course (Bachelor's thesis, University of Twente).

[23] Rahman, M. M., & Watanobe, Y. (2023). ChatGPT for education and research:

  

Opportunities, threats, and strategies. Applied Sciences, 13(9), 5783. https://doi.org/10.3390/app13095783

[24] Prather, J., Denny, P., Leinonen, J., Becker, B. A., Albluwi, I., Craig, M., ... & Savelka, J. (2023). The robots are here: Navigating the generative ai revolution in computing education. In Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (pp. 108-159) https://doi.org/10.1145/3623762.3633499

[25] Robert, J., & Schmidt, D. (2024). 10 Benefits and 10 Challenges of Applying Large Language Models to DoD Software Acquisition. https://doi.org/10.58012/ygk8-kf82.

[26] Lyu, W., Wang, Y., Chung, T., Sun, Y., & Zhang, Y. (2024). Evaluating the effectiveness of llms in introductory computer science education: A semester-long field study. In Proceedings of the Eleventh ACM Conference on Learning@ Scale, 63-74. https://doi.org/10.1145/3657604.3662036

[27] Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? assessing the security of github copilot's code contributions. In 2022 IEEE Symposium on Security and Privacy (SP), 754-768. doi: 10.1109/SP46214.2022.9833571.

[28] NIST. (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). https://doi.org/10.6028/NIST.AI.100-1

[29] Generative AI Tools for Teaching and Learning | Center for Teaching & Learning | Utah Tech University. (n.d.). https://ctl.utahtech.edu/aitools/

[30] Frankford, E., Höhn, I., Sauerwein, C., & Breu, R. (2024, May 30). A Survey Study on the State of the Art of Programming Exercise Generation using Large Language Models. https://doi.org/10.48550/arXiv.2405.20183.

  