Jung Hyun MOON
ORCID: 0009-0008-6619-756X
NYU Shanghai, CHINA

Fangqing Quinn HE
ORCID: 0009-0000-1589-9459
Researcher, USA

Shiyuan Sissy TIAN
ORCID: 0009-0005-2462-5528
Cornell University, USA

Jace HARGIS, PhD
ORCID: 0000-0002-9372-2533
Researcher, USA

ABSTRACT

The significant breakthroughs in large language models (LLMs) in recent years have enabled the development of numerous conversational artificial intelligence (AI) tools, such as OpenAI ChatGPT and GitHub Copilot. These tools have transformed the landscape of computing education by offering personalized assistance, instant feedback, debugging suggestions, and code completion support. Through a case study based on extensive teaching experience, we explore the benefits and challenges of using conversational AI in creative coding, providing practical guidelines for both learners and instructors. The research categorizes students into beginner, intermediate, and advanced levels, analyzing their interactions with AI tools across nine dimensions including usage frequency, prompt engineering skills, and learning experiences. Our findings indicate that conversational AIs can substantially improve learning efficiency and engagement, and they also require careful course design and management to prevent over-reliance and develop critical problem-solving skills. Further, Our findings reveal that while advanced students effectively leverage AI tools to enhance their capabilities, beginners and intermediate learners often struggle, facing issues such as over-reliance on AI-generated code and difficulty in critical assessment. The study highlights both the potential benefits of AI tools in providing personalized, immediate feedback and their limitations, including the risk of hindering fundamental skill development and creativity. Based on these findings, we propose practical guidelines for learners and instructors of creative coding on effectively utilizing these tools. We aim to lower the barrier for creative coding practitioners and inspire further exploration in computing education, particularly within the art and design fields.

Keywords: Large Language Model (LLM), Artificial Intelligence (AI), Conversational AI, Learning Efficiency, Learner Engagement, Computing Education, Creative Coding, ChatGPT, GitHub Copilot

INTRODUCTION

Creative coding, which merges programming with artistic expression, is becoming essential in education, fostering technical and creative skills and lowering the barrier for artists, designers, and musicians to use programming for diverse and innovative works (Greenberg, 2007; Peppler & Kafai, 2009). However, current teaching and learning methods often struggle to balance creativity and technicalities, fail to support diverse student needs, and hinder effective student interaction and exploration. This makes it crucial to explore the most effective approaches for creative coding education.

Conversational AI tools represent a significant advancement in the field of creative coding education. Built on Large Language Models (LLMs), such as ChatGPT, act like personal assistants that understand

your questions, provide instant feedback, and tailor explanations to your unique approach to learning. These AI tools offer specific benefits for creative coders that cater to their unique needs. For instance, when creating digital artwork, these AI tools can generate code snippets tailored to algorithmic visualization, debug creative implementation errors, and suggest innovative enhancements to artistic projects. This helps creative coders experiment with new techniques, iterate on their designs rapidly, and overcome technical barriers that might otherwise hinder their creative process. The personalized assistance and immediate support provided by these tools significantly enhance the learning experience, making it more accessible and effective for creative coding practitioners.

However, using this approach is not all smooth sailing. There are specific limitations to integrating conversational AI in creative coding education. For creative coders, the AI tools might sometimes provide incorrect or suboptimal solutions that fail to capture the nuances of artistic algorithms and creative expressions. Their understanding of highly context-specific artistic problems can be limited, leading to suggestions that lack the necessary creativity or originality. Additionally, over-reliance on AI can hinder the development of critical problem-solving skills essential for innovative design and artistic experimentation. Creative coders might find themselves constrained by the AI's suggestions, potentially stifling their unique artistic vision and creativity.

Given these points, it's crucial to establish guidelines for using conversational AI effectively in the context of creative coding. Our research addresses the following questions:

1. Which aspects of programming can be effectively assisted by conversational AI?
2. Which aspects of programming are less effective when assisted by conversational AI in the learning process?
3. How can **instructors** effectively integrate conversational AI into creative coding education to enhance teaching outcomes?
4. How can **learners** effectively integrate conversational AI into creative coding education to enhance learning outcomes?

It is important to note that the effectiveness discussed in this case study pertains specifically to the AI tools themselves, rather than the overall educational outcomes. Here, effectiveness is defined as the ability to maximize the utility of conversational AI tools by formulating appropriate prompts with the correct techniques to obtain responses that align with the users' intentions. This ensures that users can achieve their desired results without disrupting their natural thought process, creativity, or development flow.

To answer these questions, we conducted a case study based on the primary author (PA)'s teaching experience and observations from four courses that guide students in learning creative coding with conversational AI. The main contributions of this work are:

- **Learning:** A review of the opportunities and challenges of learning creative coding with conversational AI.
- **Teaching:** A practical guideline for instructors on teaching creative coding with conversational AI.

LITERATURE REVIEW

Creative Coding Education

Creative coding (Greenberg, 2007) is the practice of using programming languages and computational techniques as a medium for artistic expression, emphasizing creativity and experimentation over traditional problem-solving. It is utilized by artists, digital designers, musicians, and performers to create digital art, interactive installations, generative music, and responsive environments. Popular tools for creative coding include p5.js (p5.js, n.d.), Processing (Processing, n.d.), OpenFrameworks (openFrameworks, n.d.), TouchDesigner (TouchDesigner, n.d.), and Max/MSP (Cycling '74, n.d.).

The current state of creative coding education reflects its growing importance and widespread adoption in various educational institutions (Peppler & Kafai, 2009). From primary schools to universities, creative coding has become an integral part of curricula, particularly with the rise of STEAM education that emphasizes the integration of STEM (Science, Technology, Engineering, and Mathematics) with the arts (Conradty & Bogner, 2019; Shatunova et al., 2019; Hill & Hargis, 2024). This interdisciplinary approach fosters a holistic learning environment where students can develop both their technical and

creative skills. Detailed case study research suggests that creative coding is beneficial for enhancing computational thinking and contextualized computing abilities, achieving the dual goals of teaching fundamental programming skills and comprehensive cognitive development (Hoppe & Soh, 2020; Peppler & Kafai, 2009). Additionally, by visualizing programming languages, creative coding captures students' attention and interest, thereby boosting engagement and enhancing the overall learning experience (Terroso & Pinto, 2022). More importantly, as a tool for artistic expression, creative coding significantly benefits students in design, art, and creativity, enabling them to explore new forms of digital artistry and innovation (Greenberg, Kumar, & Xu, 2012; Hoppe & Soh, 2020). The growing emphasis on creative coding in education and its potential to enrich holistic student development underscores the importance of enhancing creative coding education.

There is a growing body of research focused on the teaching and learning of creative coding. Benedetti, Elli, and Mauri (2020) developed strategies and a framework for teaching creative coding to university students with diverse backgrounds, emphasizing the importance of immediate visual feedback, interactive learning, scoped coding, and project-based learning (PBL) in fostering creativity and technical skills. As online resources increasingly serve as the primary environment for self-directed learning and experimentation in creative coding beyond formal institutional courses, many existing research have been focused on them. Nylén et al. (2015) analyzed the use of forums and blogs for massive open online creative coding courses, finding it challenging to foster high-quality discussions due to diverse student levels and the hands-on nature of creative coding. Another specific example of student projects explored the learning and appropriation processes in collaborative creative coding spaces, suggesting applying code to new fields and employing it for artistic expression is an effective method for learning (Judith, Benjamin, & Rebecca, 2019). In addition, pioneering studies have explored the use of AI and automation in learning creative coding. Evans (2024) integrated automated feedback into a creative coding course by implementing script templates for student exercises, highlighting the need to balance feedback on fundamental concepts with nurturing student creativity. Jonsson and Tholander (2022) conducted a workshop on "Programming with AI" for students in a creative programming course, emphasizing the creative, embodied, and craft-oriented nature of creative coding making it crucial for students to understand how to effectively interact and co-create with the AI system.

Despite their contributions, existing research reveals significant gaps and opportunities for improvement in teaching and learning creative coding. There is an urgent need for educational methods that prioritize creativity over technicalities and cater to diverse student needs. Methods with conversational AI tools that could support students' effective interaction and exploration of the system and balance student creativity with system performance could possibly address the limitations of traditional learning methods. Our research aims to develop comprehensive guidelines to harness the full potential of creative coding education.

Computing Education with Conversational AI

Among the various functions of conversational AI tools in programming, code generation, debugging, and code explanation are the most frequently utilized (Hargis, & Gessner, 2024; Perera, Wijayanayake, & Prasadika, 2024; Wade, Hargis, & Gessner, 2024). These functions offer numerous benefits for programming learning. Studies show that students using Gen AI for programming learning can acquire new syntax, code constructs, and problem-solving methods, while also gaining a better synthesis of concepts through AI-provided example codes and explanations (Denny et al., 2024; Yan, Nakajima, & Sawada, 2024; MacNeil et al., 2022; Xue et al., 2024). Additionally, conversational AI can effectively assist with debugging by providing explanations to help programmers better handle error messages and facilitating discussions on how to resolve them (Leinonen et al., 2023; Yan, Nakajima, & Sawada, 2024; Xue et al., 2024). With these benefits, instructors, and learners can leverage conversational AI tools as an additional resource for improving knowledge construction through tailored explanations and examples.

Conversational AI not only provides direct benefits to programmers but also significantly enhances the overall learning experience and skill development for students. One key advantage is interactive engagement, where AI-enabled systems offer immediate, personalized feedback by recognizing individual learning preferences and knowledge levels (Hake, 2002; Mallik & Gangopadhyay, 2023). This interactive AI partnership has been shown to significantly increase student learning engagement and outcomes (Huang, Lu, & Yang, 2023). Students report that using natural language for interaction makes tasks easier, allowing them to better analyze problems and understand the material (Brachten et al.,

2020; Yan, Nakajima, & Sawada, 2024). Additionally, conversational AI breaks down complex tasks, addresses syntax issues, and provides scaffolding, thereby reducing the cognitive load (Brachten et al., 2020). This makes learning more efficient and less frustrating, allowing learners to focus more deeply on complex problems and the overall design of their programs, fostering algorithmic thinking and problem-solving skills (Abbasi, Kazi, & Hussaini, 2019; Brachten et al., 2020; Schmidhuber, Schlögl, & Ploder, 2021). Moreover, prompting and evaluating AI responses promotes metacognitive learning and critical thinking skills, encouraging students to reflect on and regulate their cognitive processes, and fostering effective learning patterns for their future learning (Prather et al., 2023; Tankelevitch et al., 2024).

In addition to the benefits of conversational AI in general programming and computing education, these tools also offer significant advantages in the specific domain of creative coding. Research has shown that AI tools not only assist with programming but also enhance the creative process. Artist interviews reveal that they find AI useful for enhancing their work, generating expressive prompts for inspiration, and exploring new creative directions (Angert et al., 2023). Studies by Kery and Myers (2017) and Angert et al. (2023) demonstrated that AI effectively supports creative coding by quickly processing various creative ideas, transforming them into low-level program syntax, and enabling rapid exploration of many variations. This allows artists to focus on larger creative leaps without getting bogged down by technical difficulties (Jacobs et al., 2017). Furthermore, Jonsson and Tholander (2022) found that for university students learning creative coding, conversational AI acts as a collaborator in the creative process, generating suggestions and expressions that inspire further development. Additionally, AI systems help students by handling programming complexities, thus allowing them to realize their creative ideas more effectively. This dual role of AI—both as a technical assistant and a creative partner—illustrates its potential to reshape the field of creative coding, making it an invaluable tool for both artists and students.

Despite the potential of conversational AI in creative coding, there are significant challenges and a lack of comprehensive research in this context. Students report initial misunderstandings and receive random and inaccurate feedback, adding to their workload and frustration (Jonsson & Tholander 2022; Qureshi, 2023). Educators observe workflow challenges for students in balancing the human-AI speed and cognitive disparity, and in combining creativity with AI-provided elements (Evans, 2024; Kalota, 2024; Yan, Nakajima, & Sawada, 2024). These challenges in creative coding education with conversational AI extend beyond general computer science education, as creative coders focus on artistic creation and tools to aid it, rather than technical jargon and intricacies. Creative coders often struggle to ask effective questions and understand the limitations of AI tools in artistic creation. Given these issues, it is important to provide guidelines specifically targeting creative coding education, instead of general computer science. Moreover, even within the few existing research, they predominantly focus on the impact of AI on artistic creation and student outcomes rather than on the learning process, usage methods, and teaching guidelines. Since conversational AI is a double-edged sword in computing education, it is crucial to develop specific guidelines and examples for its effective integration into the domain of creative coding. Our research aims to provide this essential support for both learners and instructors.

METHODS

In Fall 2023, Spring 2024 and Summer 2024, the primary author (PA) developed and taught three bachelor's level courses and one master's level course on creative coding. This case study design includes 57 students (21 males and 36 females) with varying levels of computer science proficiency through four courses between fall 2023 and spring 2024. Among these students, 21 were beginners with no prior coding experience to learning Object-oriented programming (OOP), 18 were at an intermediate level with a basic understanding of OOP and good knowledge of p5.js, and 18 were advanced students with strong computer science backgrounds and proficiency in multiple programming languages and frameworks. Complete participant demographic information is provided in table 1.

Table 1. Participant Demographics

Foundational Creative Coding Course (Spring 2024)	
Total Participants	16
Gender	5 male, 11 female
Nationality	10 Chinese and 6 International (US, Thailand, Italy, Russia)
Academic Year	7 Freshmen, 3 Sophomores, 2 Juniors and 4 Seniors
Major	8 Undecided Major 2 Computer Science 2 Interactive Media Arts 1 Social Science 1 Humanities 1 Business and Finance 1 Media, Culture, and Communication / Business of Entertainment
Programming Background	Mostly beginners except one advanced student
Goals and Project Types	Introduce programming fundamentals and creative coding basics through generative/interactive visualizations and animations, culminating in a completed web project.
Tools	p5.js, HTML/CSS/JS
AI Tool Usage	**Not recommended**, but documentation of experiences required if used

Intermediate Creative Coding Course (Spring 2024)	
Students	12
Gender	2 male, 10 female
Nationality	10 Chinese and 2 International (US, Brazil)
Academic Year	8 Sophomores, 4 Seniors
Major	7 Interactive Media Arts 1 Interactive Media and Business 1 Computer Science 1 Interactive Media Arts / Computer Science 1 Interactive Media Arts / Business of Entertainment 1 Interactive Media Arts / Business and Marketing
Programming Background	Mostly intermediate-level students, two experienced students
Goals and Project Types	Enhance creative coding skills with a focus on realistic algorithmic animation, utilizing mathematical, physical calculations, and advanced programming concepts by exploring generative artwork, algorithmic animations, and interactive video installations.
Tools	p5.js, HTML/CSS/JS
AI Tool Usage	**Recommended**, with documentation and evaluation of outcomes

Advanced Creative Coding Course (Fall 2023)	
Students	12

Gender	7 male, 5 female
Nationality	8 Chinese and 4 International (US, Mongolia, South Korea)
Academic Year	2 Sophomores, 1 Junior, 9 Seniors
Major	9 Interactive Media Arts 2 Interactive Media Arts / Computer Science 1 Interactive Media Arts / Business and Finance 1 Interactive Media Arts / Creativity and Innovation
Programming Background	Strong background in computer science and interactive media arts
Goals and Project Types	Develop a shared virtual space on the web employing advanced creative coding skills, including 3D animation, server-side programming, and machine learning.
Tools	p5.js, three.js, node.js, ml5.js
AI Tool Usage	**Recommended**, with documentation and evaluation of outcomes

AI Art Course in Master's Program (Summer 2024)	
Students	17
Gender	7 male, 10 female
Nationality	8 Chinese and 9 International (US, Europe, South Korea)
Academic Year	17 first-year master's students (one-year program)
Major	17 Interactive Media Arts
Programming Background	Varied, from absolute beginners to experienced programmers (Beginner: 6 Mid: 8 Advanced: 3)
Goals and Project Types	Explore newly developed machine learning models and their applications in creative coding, producing AI-driven art, interactive experiences, and generative visualizations using AI.
Tools	p5.js, ml5.js, and other ML models
AI Tool Usage	**Strongly encouraged** to use AI tools, with constant discussion and documentation of experiences

In one foundational creative coding course, conversational AI tools were not introduced by the primary author (PA), but were observed to be spontaneously used by students to learn creative coding materials. In the other three courses, two conversational AI tools, ChatGPT and Github Copilot, were introduced by the PA during individual meetings. When students spontaneously used conversational AI tools, they were asked to document their usage experiences. When the PA introduced the conversational AI tools to facilitate learning, in addition to students documenting their experiences, the PA also observed their interactions with the tools during one-on-one individual meetings.

Note that there may be potential self-selection bias due to voluntary participation in discussing and documenting AI tool usage, and the observational nature of the study, which may limit the extent to which the findings can be generalized to other educational settings and populations.

The data collection methods included the PA's observations on students' interactions with ChatGPT and GitHub Copilot. The PA frequently discussed the effective uses of AI tools and any confusion regarding them during individual meetings with students, and checked their documentation of the process afterward. During the observations, the PA noticed common patterns and tailored his guidance accordingly. Upon review, it became clear that these patterns were consistent based on the students'

programming experiences. The trends and guidelines were documented in a detailed report using consistent prompts. This report included insights on how students used these tools in creative coding, the effectiveness of various prompts, strategies for coding tasks, and students' critical engagement with the tools. Combined with the students' documentation, this observation report was analyzed across nine specific dimensions for students at three levels (beginner, intermediate, and advanced):

- Frequency of AI tool usage and the extent of student reliance on AI tools.
- Prompts provided by students.
- How students processed AI responses.
- Methods used by students to seek clarification when in doubt.
- Mistakes generated by AI tools and students' responses to them.
- Learning experiences and confusion during the process.
- The number of iterations in these processes.
- Whether AI tools accelerated students' progress and enhanced their capabilities.
- The effectiveness of various aspects of AI tool usage.

The PA consistently discussed the analyzed patterns and revised guidelines with two other department instructors and the co-authors. Based on the results, we summarized the opportunities and limitations of conversational AI tools for creative coding learners and proposed a guideline for instructors interested in incorporating conversational AI into their creative coding teaching. These findings are detailed in the Discussion section of this paper.

FINDINGS

The following observation analysis is categorized by student programming experience into three levels – beginner, mid-level, and advanced – highlighting their conversational AI usage, prompt engineering skills, ability to clarify and follow up, confusion levels, mistake detection, iterative processes, learning experiences, and impact on capabilities.

Table 2. Observation Analysis Categorized by Student Programming Experience

Beginner Level Students	
Frequency of AI Tool Usage	Infrequent for coding. (Frequent for writing)
Reliance on AI Tools	Mixed reliance on AI tools, either not using them at all or relying heavily when they do. AI tool usage was generally discouraged; Exercises were simple enough to be completed using class example codes. However, those who used AI tools tended to rely on them more substantially.
Prompt Engineering	Unable to craft useful prompts due to a lack of programming knowledge, leading to broad and less relevant responses from the AI.
Clarification and Follow-Up Questions	Often failed to identify key information and could not formulate effective follow-up questions, accepting AI responses without critical consideration.
Confusion Levels	Easily confused by AI responses, frequently using generated code without understanding.
Detecting Mistakes by AI	Generally unable to detect AI mistakes.
Amount of Iteration	Extremely limited.
Learning Experience	Generally negative, as relying on AI tools reduced opportunities for learning and creative exploration.
Accelerated / Enhanced Capabilities	Work processes were skipped, and capabilities were not enhanced by AI tools.

Mid-Level Students	
Frequency of AI Tool Usage	Frequent.
Reliance on AI Tools	Moderate reliance, with students making efforts to assess the responses.
Prompt Engineering	Able to provide effective prompts but struggle with complex programming techniques and processes.
Clarification and Follow-Up Questions	Able to form follow-up questions for clarification and extract useful information.
Confusion Levels	Still experienced confusion and may use AI-generated code snippets without fully understanding them.
Detecting Mistakes by AI	Generally unable to detect mistakes made by AI tools and may give up using the code rather than cross-checking with other references to resolve issues.
Amount of Iteration	Limited, focusing more on finding immediate solutions rather than exploring various possibilities.
Learning Experience	Quickly learned functionalities but may develop habits that neglect the importance of programming fundamentals and the effort to produce complex code by themselves.
Accelerated / Enhanced Capabilities	AI tools accelerate their work process but do not enhance their capabilities.

Advanced Students	
Frequency of AI Tool Usage	Very frequent.
Reliance on AI Tools	Low reliance, with critical assessment of responses.
Prompt Engineering	Provided excellent prompts achieving intended outcomes.
Clarification and Follow-Up Questions	Formulated effective follow-up questions and extracted valuable information from AI responses.
Confusion Levels	Capable of resolving confusion through multiple follow-up questions due to their solid programming foundation.
Detecting Mistakes by AI	Detected and fixed incorrect information by cross-checking with other references and were able to provide correct information back to the AI.
Amount of Iteration	Engaged in extensive iteration for various experiments and exploration.
Learning Experience	Rapid learning of new concepts and techniques in diverse programming. languages and frameworks.
Accelerated / Enhanced Capabilities	AI tools accelerated their work process and enhanced their capabilities, enabling more advanced experimentation and problem-solving.

While advanced students effectively utilize conversational AI, a number of issues were identified among students at other levels. Interestingly, many students reported finding ChatGPT extremely useful for solving their coding problems. However, from an instructor's perspective, these students could not use the conversational AI tools effectively. The outcomes produced were mundane in both creative and technical aspects, as the AI tools reduced opportunities for students to explore creatively and develop unique ideas and outcomes on their own. This also diminished their chances to deepen their understanding of programming and engage in meaningful problem-solving processes. The following are common issues identified among beginner and mid-level students:

- Students often accepted ChatGPT's responses without thoughtful consideration or evaluation, indicating a lack of critical thinking skills and understanding of the computational process.
- Advanced syntax, such as p5.Vector and ES6, confused students. They tended to use these complex elements without understanding them, mistakenly believing they were the only solutions.
- Asking ChatGPT for entire code drafts (e.g., drawing a face or a fish) and using them directly constrained creativity. Students struggled to break down tasks due to a weak foundation in computational thinking.
- Students generally accepted ChatGPT's initial suggestions without asking follow-up questions or experimenting with various solutions.
- The extensive information and overloaded text in ChatGPT's responses distracted students, making it difficult for them to identify the core ideas.
- Combining multiple code snippets generated by ChatGPT led to patchy, redundant, and unoptimized code, lacking intentionality.
- Students predominantly used ChatGPT as a problem-solving tool rather than experimenting with various approaches and methodologies.
- Students rarely inquired about higher-level programming concepts, instead asking for specific code snippets and usage clarifications.
- Students generally found GitHub Copilot confusing and not particularly useful. They struggled to process its suggestions due to a lack of foundational programming knowledge, and irrelevant auto-completions and advanced syntax further confused them.
- Most students did not know how to use GitHub Copilot effectively, primarily relying on its auto-completion feature rather than its conversational capabilities or using comments to provide prompts, data, and examples of code snippets.

DISCUSSION

ChatGPT can be used in many helpful ways for learning and problem-solving. It can engage in conversations, answer questions, and provide detailed responses. To use it effectively, users could role-play specific scenarios by clearly stating the background and context, and assigning roles for themselves and ChatGPT. Asking specific questions is crucial, and if responses are unclear, follow-up questions should be posed. Users can guide ChatGPT by asking it to remove certain responses and focus on specific concepts. Sharing any confusion and asking for clarification will help users obtain the intended answers.

However, this approach is not always suitable for programming education. Programming requires precise values and specific solutions, and outcomes can vary with different techniques. This challenge is amplified in creative coding due to the inherent variability and creativity in results.

In this section, we would like to explore the opportunities and limitations of conversational AI tools in the context of creative coding education, and then propose two practical guides for learners and instructors to facilitate their usage. The primary objective is to provide answers to the following four research questions:

1. Which aspects of programming can be effectively assisted by conversational AI?
2. Which aspects of programming are less effective when assisted by conversational AI in the learning process?
3. How can **instructors** effectively integrate conversational AI into creative coding education to enhance teaching outcomes?
4. How can **learners** effectively integrate conversational AI into creative coding education to enhance learning outcomes?

Opportunities of conversational AI in creative coding education

Conversational AI tools, such as ChatGPT and GitHub Copilot, have great potential in creative coding education. They can be applied in various ways to enhance the learning process. They also offer code completion, suggesting the next part of a code sequence, which helps learners grasp the flow of programming. Debugging assistance is another valuable feature, as ChatGPT and GitHub Copilot can help identify and fix errors in code. Additionally, they can generate code documentation adding comments on each line, code block entire code and the concepts and techniques of the programming, making it easier for students to understand how different parts of their code work. As learning and

tutoring tools, ChatGPT and GitHub Copilot provide instant feedback and explanations, making complex concepts and techniques more accessible.

Below are example prompts that learners can ask conversational AI tools to facilitate their learning.

- **Code Generation:** “ChatGPT, can you write a p5.js sketch that draws a spinning ellipse at the center of the canvas?”
- **Code Completion:** “ChatGPT, here is my unfinished code snippet. Can you complete my code so that the ellipse rotates at the center of the canvas?”
- **Debugging Assistance:** “ChatGPT, I received this error message when I ran my p5.js sketch. Can you rewrite the code, debug it for me, and explain why it happens?”, "ChatGPT, the ellipse doesn't rotate properly at the center of the canvas. Can you fix it and explain what was wrong?"
- **Documentation Generation:** “ChatGPT, can you generate documentation for the transformation functions that I just wrote?”, "ChatGPT, the transformation functions are pretty confusing. Can you add comments to each line of the code below?"
- **Instant Feedback & Explanation:** “ChatGPT, can you explain what `translate()` function is?”

The always-available, customizable, and interactive features of conversational AI tools help resolve some of the pain points associated with traditional methods of teaching creative coding.

One issue is the lack of personalized learning where instructors often cannot provide individualized attention to every student, leading to potential gaps in higher level applications. There is often limited immediate feedback, which means students might struggle with errors or misconceptions for longer periods. In contrast, ChatGPT and GitHub Copilot can provide support to learners at any time and from anywhere, students can quickly get started and help and move forward with their projects. Personalized learning is a key advantage, as these tools can tailor their responses to individual directions and specific questions. This leads to increased productivity, saving students time and helping them apply various coding patterns.

Additionally, access to learning resources can be uneven, with some students lacking the tools or support they need outside the classroom. Students have had to use official documentation of the programming language and utilize search engines for references and solutions. This might waste their time due to incorrect irrelevant information or overwhelming and lengthy tutorials. Conversational AI tools can filter and extract relevant information for learners, significantly reducing the information barrier for those who are self-learning and lack access to human instructors.

Limitations of conversational AI in creative coding education

While ChatGPT and GitHub Copilot offer many benefits, they have limitations. In fact, inappropriate use of these conversational AI tools can negatively impact learners' motivation and lead to confusion. We would like to discuss several limitations that we have observed while integrating AI platforms in teaching and learning:

- The quality of conversational AI's responses heavily depends on the prompts provided by learners. However, many creative coding practitioners may lack the necessary prompt engineering skills to effectively communicate their needs, which can result in less relevant or accurate answers from the AI.
- The AI might suggest syntax and techniques that are too advanced, causing students to struggle with applying or using the code without a solid grasp of the concepts, potentially leading to unproductive frustration.
- Although ChatGPT and GitHub Copilot perform well with basic tasks, they may struggle with cutting-edge tools not included in the current training cycle and advanced libraries that lack sufficient documentation.
- Conversational AI tools can sometimes offer incomplete code suggestions that require verification against official documentation due to occasional low-quality training data. This can be particularly challenging for beginners, who may lack the background knowledge to understand official documentation and may quickly lose confidence.
- These tools sometimes provide mundane examples that hinder learners' creativity. This can be especially unhelpful for creative coding practitioners who look to example codes for inspiration.
- There is a risk that students might not learn the fundamental principles of coding if they depend too much on AI-generated high-level solutions. The ease of using AI tools may provide answers

without allowing learners time to process the information, ultimately hindering the development of critical thinking, creativity, and problem-solving skills.

How Learners Can Use Conversational AI Tools More Effectively

To help learners use conversational AI tools more effectively, we illustrate the opportunities and challenges with practical examples. We experimented with various use cases in creative coding and reviewed what works and what doesn't. This serves as a cheat sheet for learners, including example queries to generate effective answers and avoid ineffective ones.

Based on our observations and experiments, we summarized what conversational AI tools can be effective, ok and not ideal for in table 3. Again, as we mentioned previously in the introduction, the "effectiveness" discussed in this case study and in Table 3 refers to the ability of AI tools to produce responses that align with users' intentions without disrupting their natural thought process, creativity, or development flow.

Table 3. Effectiveness of ChatGPT and GitHub Copilot for Various Programming Tasks

Effectiveness	Tasks	Details
Great for:	**Programming languages with extensive documentation and abundant online resources** Example: Web development tools like HTML and CSS.	ChatGPT and GitHub Copilot excel at generating and troubleshooting code for HTML and CSS. The extensive references and tutorials available online can make finding precise solutions time-consuming. However, ChatGPT and GitHub Copilot provide up-to-date and optimal solutions, streamlining the process. GitHub Copilot accelerates development by offering real-time code suggestions within integrated development environments (IDEs).
	Frameworks with unique syntax and a steep learning curve Example: Web development libraries like Bootstrap, Tailwind CSS, and React.js.	Due to their steep learning curves, these libraries require significant experience to master. ChatGPT and GitHub Copilot provide accurate and extensive explanations and solutions for developing with these advanced frameworks, enhancing productivity and learning. ChatGPT can generate code examples, snippets, and modules and explain functional programming concepts effectively.
	Visual development environments with graphical programming interfaces, extensive documentation, and large communities Example: Touch Designer and Unity.	The abundant documentation, references, and tutorials from official sources and community members make ChatGPT more effective for accurate solutions. ChatGPT is especially beneficial for students lacking programming experience, helping them integrate and manage code in projects where graphical user interface (GUI) tools are predominantly used.
	Converting code between similar syntax languages Example: Processing to p5.js or OpenFrameworks, and vice versa.	ChatGPT and GitHub Copilot excel at converting code into different programming languages or libraries, particularly between similar syntax languages like Processing, p5.js, and OpenFrameworks.
	Creating functions with complex calculations Example: Physical calculations using vectors and rotation angles in 3D space.	ChatGPT and GitHub Copilot are excellent for performing complex mathematical or physical calculations using specific equations. GitHub Copilot's context-aware code suggestions within an IDE are practical for efficiently implementing functions with intended calculations.
Okay for:	**Advanced libraries with solid but limited documentation and small communities** Example: three.js.	ChatGPT may provide useful information and outlines but can sometimes offer inaccurate or less reliable suggestions based on customized code snippets. GitHub Copilot, however, is better suited for advanced libraries like three.js. It learns from the user's coding

		style and provides more accurate code generation and completion, significantly improving productivity.
Not Ideal for:	**Learning programming fundamentals** Example: Basic JavaScript or p5.js	ChatGPT and GitHub Copilot often introduce advanced techniques and new syntax in ES6, which can confuse beginners. Additionally, ChatGPT tends to provide an overwhelming amount of completed code and explanations that are challenging for beginners to grasp.
	Creative coding with programming libraries in the context of visual arts Example: p5.js and Processing	Suggestions from ChatGPT and GitHub Copilot might be too mundane and not conducive to fostering creativity. Instead, students should be encouraged to experiment with various drawing techniques and visual outcomes independently.

How Instructors Can Use Conversational AI Tools More Effectively

In this section, we focus on the effective usage of conversational AI tools from the perspective of instructors. We propose the four guidelines for instructors interested in utilizing conversational AI to teach creative coding:

1. Tailor ChatGPT Use to Student Levels
2. Encourage step-by-step learning
3. Enhance verification practices and critical thinking process
4. Monitor students' progress and provide feedback

In the appendix, we will expand on each guideline and demonstrate how to apply them in practice.

CONCLUSION AND FUTURE WORK

This work emphasizes the importance of discussing conversational AI in the context of creative coding education. As noted, few prior studies have focused on empowering creative coding practitioners to leverage the potential of conversational AI. Our study highlights the potential benefits and challenges associated with integrating tools like ChatGPT and GitHub Copilot into creative coding education, providing practical insights into their effective use.

We offer specific directions and guidelines for:

- **Learners**: Those interested in learning creative coding with conversational AI.
- **Instructors**: Those interested in teaching creative coding with conversational AI.

Our main contributions include:

1. **Learning**: A review of the opportunities and challenges of learning creative coding with conversational AI.
2. **Teaching**: A practical guideline for instructors on teaching creative coding with conversational AI.

The objective is to lower the barrier for creative coding practitioners who are unfamiliar with or lack confidence in using conversational AI. By facilitating their initial engagement with these AI tools, this research aims to reduce potential confusion and frustration, thereby motivating practitioners to further explore and utilize conversational AI.

It is important to note that this work was conducted when the application of conversational AI in creative coding was still in its early stages. We advocate for further research that explores conversational AI within the context of computing education, particularly for coders in the art and design fields, including artists, creative coders, and musicians. Additionally, we recognize the limitations of our study and suggest future research to address important topics not covered in this work.

This study primarily focuses on methodology and can be enhanced through the application of the guidelines in real-world scenarios. In our future work, we plan to develop a customization guide for ChatGPT, design tailored activities and assignments, conduct experiments with students, and gather their feedback to refine the guidelines. Additionally, the study primarily addresses conversational AI as

a whole without examining the differences between various tools and their respective pros and cons. We plan to conduct further research comparing these AI tools, which will enable us to provide informed tool selection strategies for learners and instructors.

Lastly, we would like to advocate for more future works on a) effective personalization prompt engineering techniques for creative coding practitioners to apply; and b) conversational AI tools integration with creative coding develop tools (e.g., a plugin that allows creative coders to use code suggestion copilot tools in p5.js web editor). These efforts will further enhance the accessibility and effectiveness of conversational AI in creative coding education.

REFERENCES

Abbasi, S., Kazi, H., & Hussaini, N. (2019). *Effect of Chatbot Systems on Student's Learning Outcomes*. *63*, 49–63.

Angert, T., Suzara, M., Han, J., Pondoc, C., & Subramonyam, H. (2023). Spellburst: A Node-based Interface for Exploratory Creative Coding with Natural Language Prompts. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 1–22. https://doi.org/10.1145/3586183.3606719

Benedetti, A., Elli, T., & Mauri, M. (2020). *Drawing with code: The experience of teaching creative coding as a skill for communication designers* (p. 3488). https://doi.org/10.21125/edulearn.2020.0982

Beth Kery, M., & Myers, B. A. (2017). Exploring exploratory programming. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 25–29. https://doi.org/10.1109/VLHCC.2017.8103446

Brachten, F., Brünker, F., Frick, N. R. J., Ross, B., & Stieglitz, S. (2020). On the ability of virtual agents to decrease cognitive load: An experimental study. *Information Systems and E-Business Management*, *18*(2), 187–207. https://doi.org/10.1007/s10257-020-00471-7

Conradty, C., & Bogner, F. X. (2019). From STEM to STEAM: Cracking the Code? How Creativity & Motivation Interacts with Inquiry-based Learning. *Creativity Research Journal*, *31*(3), 284–295. https://doi.org/10.1080/10400419.2019.1641678

Cycling '74. (n.d.). *Max/MSP*. https://cycling74.com/products/max

Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., & Reeves, B. N. (2024). Prompt Problems: A New Programming Exercise for the Generative AI Era. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, 296–302. https://doi.org/10.1145/3626252.3630909

Evans, A. (2024). Integrating Automated Feedback into a Creative Coding Course. *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 2*, 799. https://doi.org/10.1145/3649405.3659490

Greenberg, I. (2007). *Processing: Creative Coding and Computational Art*. Berkeley, CA: Apress.

Greenberg, I., Kumar, D., & Xu, D. (2012). Creative coding and visual portfolios for CS1. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 247–252. https://doi.org/10.1145/2157136.2157214

Hake, R. (2002). Lessons from the Physics Education Reform Effort. *Conservation Ecology*, *5*(2), art28. https://doi.org/10.5751/ES-00286-050228

Hargis, J., & Gessner, R. (2024). Connecting functional educational technology to higher education andragogy using Generative Artificial Intelligence. *Glokalde SoTL Journal Special Issue on Generative Artificial Intelligence, 10*(2), Article 1.

Hill, C., & Hargis, J. (2024). If ChatGPT is writing the courses and the assignments then why do we need faculty or students? An ethics lesson on academic integrity and Generative AI in Higher Education. New Directions for Teaching and Learning.

Hoppe, D., & Leen-Kiat Soh. (2020). Thesis: Engagement and computational thinking through creative coding. University of Nebraska-Lincoln. https://doi.org/10.13140/RG.2.2.13155.99364

Huang, A. Y. Q., Lu, O. H. T., & Yang, S. J. H. (2023). Effects of artificial Intelligence–Enabled personalized recommendations on learners' learning engagement, motivation, and outcomes in a flipped classroom. *Computers & Education*, *194*, 104684. https://doi.org/10.1016/j.compedu.2022.104684

Jacobs, J., Gogia, S., Měch, R., & Brandt, J. R. (2017). Supporting Expressive Procedural Art Creation through Direct Manipulation. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, 6330–6341. https://doi.org/10.1145/3025453.3025927

Jonsson, M., & Tholander, J. (2022). Cracking the code: Co-coding with AI in creative programming education. *Proceedings of the 14th Conference on Creativity and Cognition*, 5–14. https://doi.org/10.1145/3527927.3532801

Judith, A., Benjamin, E., & Rebecca, S. (2019). *Programming the Postdigital: Curation of Appropriation Processes in (Collaborative) Creative Coding Spaces*.

Kalota, F. (2024). A Primer on Generative Artificial Intelligence. *Education Sciences*, *14*, 172. https://doi.org/10.3390/educsci14020172

Leinonen, J., Hellas, A., Sarsa, S., Reeves, B., Denny, P., Prather, J., & Becker, B. A. (2023). Using Large Language Models to Enhance Programming Error Messages. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 563–569. https://doi.org/10.1145/3545945.3569770

MacNeil, S., Tran, A., Mogil, D., Bernstein, S., Ross, E., & Huang, Z. (2022). Generating Diverse Code Explanations using the GPT-3 Large Language Model. *Proceedings of the 2022 ACM Conference on International Computing Education Research - 2*, 37–39. https://doi.org/10.1145/3501709.3544280

Mallik, S., & Gangopadhyay, A. (2023). Proactive and reactive engagement of artificial intelligence methods for education: A review. *Frontiers in Artificial Intelligence*, *6*, 1151391. https://doi.org/10.3389/frai.2023.1151391

Nylén, A., Thota, N., Eckerdal, A., Kinnunen, P., Butler, M., & Morgan, M. (2015). Multidimensional analysis of creative coding MOOC forums: A methodological discussion. *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, 137–141. https://doi.org/10.1145/2828959.2828971

openFrameworks. (n.d.). *openFrameworks*. https://openframeworks.cc

Peppler, K., & Kafai, Y. (2009). Creative coding: Programming for personal expression. In Proceedings of the 8th International Conference on Computer Supported Collaborative Learning (CSCL) (2, pp. 76-78). Rhodes, Greece.

Perera, K. G. D. K., Wijayanayake, J., & Prasadika, J. (2024). *Factors Affecting the Effectiveness of Generative Artificial Intelligence Apps on University Students' Programming Language Learning in Sri Lanka: A Systematic Literature Review*.

Prather, J., Reeves, B. N., Denny, P., Becker, B. A., Leinonen, J., Luxton-Reilly, A., Powell, G., Finnie-Ansley, J., & Santos, E. A. (2023). "It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Transactions on Computer-Human Interaction*, *31*(1), 4:1-4:31. https://doi.org/10.1145/3617367

Processing Foundation. (n.d.). *Processing*. https://processing.org

p5.js. (n.d.). *p5.js*. https://p5js.org

Qureshi, B. (2023). Exploring the Use of ChatGPT as a Tool for Learning and Assessment in Undergraduate Computer Science Curriculum: Opportunities and Challenges. *2023 9th*

International Conference on E-Society, e-Learning and e-Technologies, 7–13. https://doi.org/10.1145/3613944.3613946

Schmidhuber, J., Schlögl, S., & Ploder, C. (2021, September 8). *Paper Presentation: Cognitive Load and Productivity Implications in Human-Chatbot Interaction*. https://doi.org/10.13140/RG.2.2.19641.75364

Shatunova, O., Anisimova, T., Sabirova, F., & Kalimullina, O. (2019). STEAM as an Innovative Educational Technology. *Journal of Social Studies Education Research*, *10*(2), 131–144.

Terroso, T., & Pinto, M. (2022). Programming for Non-Programmers: An Approach Using Creative Coding in Higher Education. *DROPS-IDN/v2/Document/10.4230/OASIcs.ICPEC.2022.13*. Third International Computer Programming Education Conference (ICPEC 2022). https://doi.org/10.4230/OASIcs.ICPEC.2022.13

Tankelevitch, L., Kewenig, V., Simkute, A., Scott, A. E., Sarkar, A., Sellen, A., & Rintel, S. (2024). *The Metacognitive Demands and Opportunities of Generative AI* (arXiv:2312.10893). arXiv. https://doi.org/10.48550/arXiv.2312.10893

TouchDesigner. (n.d.). *TouchDesigner*. https://derivative.ca

Wade, J., Hargis, J., & Gessner, R. (2024). SlideSpace: Generative Artificial Intelligence (GenAI) environment for individually optimized learning. American Society of Engineering Education conference proceedings, Portland, Oregon June 23-26, 2024.

Xue, Y., Chen, H., Bai, G. R., Tairas, R., & Huang, Y. (2024). Does ChatGPT Help With Introductory Programming?An Experiment of Students Using ChatGPT in CS1. *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training*, 331–341. https://doi.org/10.1145/3639474.3640076

Yan, W., Nakajima, T., & Sawada, R. (2024). Benefits and Challenges of Collaboration between Students and Conversational Generative Artificial Intelligence in Programming Learning: An Empirical Case Study. *Education Sciences*, *14*(4), Article 4. https://doi.org/10.3390/educsci14040433

APPENDIX

Appendix A. Guidelines for Instructors to Teach Creative Coding with Conversational AI

This appendix provides an in-depth exploration of the four proposed guidelines for instructors to effectively use conversational AI tools in teaching creative coding.

Guideline 1: Tailor ChatGPT use to student levels

When using ChatGPT and GitHub Copilot in creative coding education, it's important to adapt AI tools based on students' programming experience and proficiency. Beginners need to develop their foundational skills, while advanced students can explore complex techniques. Matching AI suggestions to skill levels prevents over-reliance, confusion, and frustration, and promotes effective learning.

For Beginners:

Discourage students from heavily relying on ChatGPT and GitHub Copilot to prevent over-reliance and limit creativity. Encourage them to explore programming fundamentals and basic techniques to develop a solid foundation. Avoid using AI tools for drafting code from scratch to ensure students understand basic concepts and techniques. Instructor guidance is crucial to prevent confusion and overwhelm from irrelevant responses and advanced syntax and techniques. Limit exposure to advanced syntax so students can focus on the basics.

Here are the steps we suggest to follow to design the experiments for beginners:

1. Instructors should provide code examples developed based on the course curriculum, copying and pasting all code into one file to upload to ChatGPT.
2. ChatGPT can accept files with extensions depending on the programming language and format, such as .pdf, .js, .py, .java, and more.
3. Use simple, clear examples to explain concepts, adding detailed explanations as comments.
4. Select appropriate syntax and techniques for beginners, asking ChatGPT to avoid using advanced syntax and functionalities. For example, use `` `let` ``, not `` `var` ``; use basic loops (i.e., for with index `` `i` ``), avoiding advanced loops (i.e., `` `for...of` ``, `` `forEach` ``); and avoid complex features like arrow functions from ES6 (i.e., `` `() => {}` ``).
5. After customization, allow students to use ChatGPT for debugging and require them to document their reflections afterward.

For Mid-Level Students:

Encourage continued experimentation with basic techniques while expanding explorations into advanced programming concepts. Limit exposure to advanced syntax to ensure focus on the basics and monitor AI usage to ensure effective use and prevent over-reliance.

Below, we listed the steps we suggest for mid-level students:

1. Instructors should develop course example codes and include them in one file to upload to ChatGPT.
2. Limit the use of advanced techniques and complex syntax from ES6.
3. Encourage students to ask ChatGPT to use metaphors and real-world analogies for conceptual questions.
4. Have students ask ChatGPT to create functions that realize simple behaviors, such as a ball falling down and stopping when the mouse is in the ball's area.
5. Allow exploration of Object-Oriented Programming (OOP) with practical tasks like, "Can you create a class for a bouncing ball in p5.js?" to practice developing various objects using classes.
6. Require students to document their reflections after using ChatGPT.

For Experienced Students:

Greater flexibility can be granted when using conversational AI tools, allowing students to explore more advanced, customized content as needed. We recommend the following steps for experienced students:

1. Instructors need to ensure awareness of data privacy and credit issues while programming.
2. There is no need to provide course code examples; students can create their own code by following project prompts.
3. Promote conversational, real-time interactive learning and step-by-step problem-solving. Students can ask questions to:
    1. utilize ChatGPT and GitHub Copilot for repetitive tasks and code suggestions,
    1. create functions that realize complex mathematical or physical calculations,
    1. evaluate and critically analyze the code provided by ChatGPT (i.e., "Review this code snippet and suggest improvements."),
    1. make connections between coding, art theory, and design concepts (i.e., "List useful functions that could incorporate the principles of design movement Dadaism into your p5.js projects."),
    1. and employ higher-level thinking for meta-cognition (i.e., "How can I optimize this algorithm?").
4. Instructors should observe and discuss students' reflections after using AI.

Guideline 2: Encourage step-by-step learning

Sequential learning can be promoted by advocating for a step-by-step approach. Students are encouraged to break tasks into smaller steps and iterate on them with AI to improve response relevance, helping them build a solid understanding of detailed coding techniques and the overall flow of programming concepts.

The context, background, and requirements need to be defined; the task or problem is explained to provide context, the code's purpose is described, the programming language specified, and any additional libraries or tools required are listed. The computational process can be described step-by-

step. Instead of asking for the entire script at once, the coding task can be broken down into manageable pieces. The logic and sequence of the program are clearly articulated, with detailed explanations provided for each step in the computational process. Students are encouraged to ask specific questions to clarify their understanding and receive precise guidance.

Table 4 provides tips and examples for better prompt engineering:

Table 4. Best Practices of Prompt Engineering for Creative Coding with ChatGPT

Tip	Practice	Example
Be Specific and Clear	Define context and requirements	"I am making an animation with snowflake-like particles using p5.js. I just started learning it and am not familiar with JavaScript, so please avoid suggesting advanced techniques and programming concepts."
	Provide detailed questions or tasks	"Please provide me with a starting point on how to write the class for the snowflakes. They should look like circles drifting downwards."
Break Down Complex Tasks	Divide tasks into smaller parts	Instead of "Draw snowflakes," break it down into smaller steps: "1) Create a circle to represent a snowflake, 2) populate them at random coordinates at the top of the canvas, 3) have them move down, and 4) if they reach the bottom of the canvas, stop the particles there."
Provide Examples and Context	Include examples to clarify your request	"Here's my p5.js function to have the particles move down. I tried to add 'drifting' behaviors, but it doesn't work. Can you help me debug it?" (followed by the code).
	Describe the desired outcome	"I would like to implement a function that changes the speed (drifting intensity) of the snowflakes based on wind. Can you make a function in the class?"
Ask for Step-by-Step Guidance	Request detailed instructions	"Guide me through constructing the particles and iterating each object, step-by-step."
	Ask for explanations of each step	"What does each line of the code do to make the snowflakes move, fall, and display? Can you add comments on each line?"
Encourage Explanations and Clarifications	Ask for detailed explanations	"I saw you applied the `noise()` function to the drift function. What does this function do? Can you explain it in detail?"
	Request clarifications if needed	"Explain how to use the `noise()` function in p5.js to simulate wind affecting the snowflakes."
	Ask for Metaphorical Explanations	"I am still confused about Object-Oriented Programming. Explain the concept using a metaphor? Consider me as a 10-year-old child."
Use Iterative Refinement	Refine prompts based on responses	"This works, but can it handle a larger number of snowflakes?"
	Encourage improvements	"The sketch is somewhat slow. Can you optimize this p5.js animation to make the snowflakes move more smoothly?"
Set Constraints and Preference	Specify constraints or preferences	"Write this function in p5.js without using vanilla JavaScript or external libraries."
	Indicate complexity level	"Provide a beginner-friendly explanation! My instructor told me not to use ES6 syntax except for 'let' and 'class'."

Guideline 3: Enhance verification practices

Encouraging students to cross-check AI-generated code against official documentation and other reliable sources is essential. This practice ensures the accuracy of the code and deepens their understanding of the programming concepts involved. Students should be guided to consult documentation, coding standards, and community forums to verify and validate the code produced by AI.

AI tools should be used as supplements, not crutches. Promote critical thinking by encouraging students to critically evaluate AI suggestions, explore alternative solutions, and be creative in modifying and optimizing code. Emphasize the importance of understanding the underlying principles and logic behind

the code rather than relying solely on AI-generated outputs. This approach helps students develop a robust problem-solving mindset and enhances their ability to write efficient and effective code independently.

Guideline 4: Monitor and provide feedback

It is crucial to regularly assess student progress and offer feedback on AI tool usage. By consistently evaluating how students use AI tools, instructors can ensure that these tools are being used effectively and prevent over-reliance on AI for coding tasks. Providing constructive feedback helps students refine their coding practices, enhances their problem-solving skills, and fosters a deeper understanding of programming concepts. Monitoring also allows for the identification of common issues or misconceptions, enabling targeted support and intervention to improve overall learning outcomes.

BIODATA and CONTACT ADDRESSES of the AUTHOR/S

Jung Hyun Moon is an Assistant Arts Professor of IMA at NYU Shanghai. As an interaction designer and creative coder, he creates interdisciplinary work in new media, bridging the experimental and mainstream in visuals and performing arts in close collaboration with choreographers, dancers and musicians. Moon is renowned for his innovative applications of algorithmic animations, machine learning, real-time motion tracking, audio visualization, and projection mapping. Most recently, he led the design and development of the ml5.js website, contributing to making ml5.js more accessible. Moon's work has been showcased at notable venues in China, such as the Shanghai Concert Hall, Shanghai Planetarium, YUZ Museum, and China Shanghai International Arts Festival. His productions have also been featured at NASA Space Apps Challenge and Brooklyn Academy of Music (BAM) in the USA, as well as in some of Korea's most prestigious museums.

Jung Hyun Moon (Assistant Arts Professor of IMA)
Contact Addresses: Shanghai, China
E-mail: jh.moon@nyu.edu
URL: https://shanghai.nyu.edu/academics/faculty/directory/jung-hyun-moon

Fangqing (Quinn) He is currently a researcher in creative coding education and machine learning education. She earned her bachelor's degree at NYU Shanghai and her master's degree at Harvard University, specializing in Computer Science and Human-Computer Interaction. Her recent work focuses on machine learning for creative coders and artificial intelligence education for children.

Fangqing (Quinn) He (Applied Machine Learning Scientist)
Contact Addresses: Texas, USA
E-mail: fh805@nyu.edu
URL: https://quinnhe.net/about

Shiyuan Sissy Tian is currently a designer and creative technologist at NYU Shanghai. Shiyuan merges diverse academic backgrounds in art, computer science, marketing, and psychology. Driven by a passion for exploring cultural, spiritual, and emotional dimensions of design, she aspires to create

innovative solutions that positively impact humanity's well-being through emotionally resonant experiences.

Shiyuan Sissy Tian (Graduate Student)
Contact Address: Beijing, China
E-mail: st4228@nyu.edu
URL: https://shiyuantian.com/about

Dr. Jace Hargis is currently researching generative artificial intelligence (GenAI) in higher education andragogy. He has worked as a Professor and various leadership roles in Qatar, China, UAE and the US (California, Florida and Hawaii). Previously, he worked as a Vice President of Curriculum and Faculty Development at Kean University; Assistant Vice Chancellor and Professor at Wenzhou-Kean University; Professor and Director of the Center for Teaching and Learning (CTL) at New York University Shanghai; a CTL Director at the University of California San Diego; a Professor and Associate Provost in Hawaii; a College Director in Abu Dhabi, UAE; an Associate Professor and Assistant Provost in northern California; and an Assistant Professor and Director of Faculty Development in Florida. He has authored a textbook, an anthology and published over 160 academic articles as well as offered presentations at academic conferences and workshops. He has a BS in Oceanography from Florida Institute of Technology, an MS in Environmental Engineering Science and a PhD in Science Education from the University of Florida. Dr. Hargis' research focuses on how people learn while integrating appropriate, relevant and meaningful instructional technologies.

Jace Hargis, PhD (Researcher)
Contact Address: USA
Email: jace.hargis@gmail.com
URL: https://jacehargis.wixsite.com/jace