Issues in Information Systems
Volume 25, Issue 3, pp. 371-385, 2024

DOI: https://doi.org/10.48009/3_iis_2024_128

Generative AI's impact on programming students: frustration and confidence across learning styles

Nada Hashmi, *Babson College, nhashmi@babson.edu*
Zhi Li, *Babson College, zli@babson.edu*
Salvatore Parise, *Babson College, sparise@babson.edu*
G. Shankaranarayanan, , *Babson College, gshankar@babson.edu*

Abstract

The integration of Generative AI (GenAI) tools into programming education is reshaping pedagogical methodologies and student experiences. Our study investigates the impacts of GenAI on confidence and frustration levels in programming among business students, with a focus on the interaction between these tools and different learning styles defined by the VARK model—Minimalist Learner, Listener Doer, Seer Doer, and Comprehensive Learner—and GenAI usage profiles—Beginner, Moderate User, Student Expert, and GenAI Expert. Using survey data from 48 students in a graduate programming course, we assessed how these learning styles and GenAI usage profiles impact students' confidence and frustration when learning to program. Our analysis, conducted using Python, shows that Seer Doer learners and Comprehensive Learners, who generally prefer visual and kinesthetic modalities, along with Moderate Users and Student Experts, experienced increased confidence. In contrast, Listener Doer and Beginners encountered higher levels of frustration. This study highlights the need for tailored educational strategies that align GenAI tool integration with the specific learning styles and previous GenAI experiences, thereby enhancing learning outcomes and minimizing student frustration in programming.

Keywords: learning styles, GenAI tools, programming, information technology, graduate students

Introduction

In today’s digital economy, learning programming is an increasingly important skill for non-technical higher education students, especially those in management and business fields. Programming skills enable these students to understand and leverage software technologies to automate routine and mundane tasks, make data-driven decisions, and help solve business challenges. Buitrago Florez et al. (2017) posit that computational thinking is critical for solving complex problems as well as understanding human thinking and behavior. They further posit that learning programming is one way of training students in computational thinking. The rise of citizen developers – business professionals who use no-code or low-code tools to develop applications – underscores the importance of these skills. Gartner predicts that by 2024, 80% of technology products and services will be built by those outside of the traditional information technology (IT) group (Gartner press release, 2021). This trend is driven by several factors including the lack of IT professionals to meet demand, limits to IT budgets, and the proliferation of low code and no code tools and platforms, including artificial intelligence (AI)-based assistants. This democratization of software development can lead to faster innovation and agile responses to business challenges and opportunities. Finally, Chilana et al. (2015) show that there exists a category of students, referred to as “conversational programmers”, who want to learn programming so that they can speak the “programmers’ language”

allowing them to have effective conversations with programmers in their roles as entrepreneurs and business decision-makers.

Generative AI (GenAI) can transform the way programming is taught in higher education, particularly for students who may not have a technical background. According to recent research by McKinsey, developers using GenAI tools can significantly improve both programmer productivity and their user experience (Deniz et al, 2023). For example, GenAI can expedite the manual and repetitive work associated with software development by auto-filling standard code, including functions, as well as documenting code functionality. Through effective prompting, programmers can iteratively work with the GenAI to debug and update existing code. GenAI prompting can also provide helpful coding strategies, by creating drafts, that will accelerate both programmer learning and new code development. In this way, these tools help novice programmers build their digital literacy on complex topics and enhance their coding skills, especially when writing software using high-level languages like Python, that may have otherwise only been received from experienced programmers. GenAI is further shown to improve student's self-efficacy and motivation with programming and serve as an effective tool to teach computational thinking. (Yilmaz &Yilmaz, 2023).

Becker et al. (2023) emphasize that GenAI can impact not just what is taught, when it is taught, and to whom it is taught but also how it is taught. However, there is scant research on how to effectively incorporate GenAI technologies for teaching programming into the higher education classroom. Courses are often comprised of students with different programming expertise, experiences, and learning styles. In addition, students' awareness and exposure to GenAI technologies may also have an impact on this learning. In this paper, we used a survey methodology in multiple sections of a Graduate course to better understand how students' learning styles and GenAI usage impacts their confidence and frustration levels when learning to program. Our study contributes to aligning GenAI tools with different learning styles to craft educational strategies to teach programming, thus helping students learn more effectively.

Literature Review

Generative AI in Education

The fast progress in artificial intelligence (AI) has spurred the creation of powerful tools, with Large Language Models (LLMs) emerging as one of the most notable examples. One prominent illustration of this progress is ChatGPT, created by OpenAI and launched in November 2022. Operating on the Generative Pre-trained Transformer architecture, ChatGPT stands out for its ability to provide human-like responses to queries. It proficiently navigates various subjects, offering answers and explanations, generating ideas, and engaging in interactive dialogues. This versatility enables ChatGPT to become one of the fastest applications to reach 100 million users in history (Hu, 2023).

The influence of AI on education has been profound, notably in improving personalized learning and administrative efficiency (Chen et al., 2020). Programming education has particularly benefited, shifting from a specialized skill to a foundational requirement in sectors like healthcare, finance, and transportation. Despite its growing importance, learning programming is often challenging, leading many students to give up due to insufficient support, debugging complexities, and the intricacies of coding (Gomes & Mendes, 2014; Yusoff et al., 2020). Generative AI tools like ChatGPT can mitigate these challenges by providing prompt and mostly accurate responses, offering support and guidance in programming pursuits. Past studies have shown that ChatGPT can be effectively integrated into programming courses, demonstrating positive impacts on programming education (Jacques, 2023; Bringula, 2024; Liu et al., 2024). Although research has highlighted the positive impacts of GenAI on programming education, it is not explicit as to when and how GenAI creates a positive impact. In this research we examine two challenges with learning

programming – frustration with the learning process and building confidence in the learner's ability to write code.

Frustration in Programming

Frustration in programming within educational environments can significantly influence learning outcomes, engagement, and student persistence. Understanding and mitigating frustration is crucial for enhancing the learning experience and ensuring student success in programming courses.

Rodrigo and Baker (2009) explored the detection of student frustration in an introductory programming course using behavioral metrics such as compilation frequency and error patterns. Their findings indicated that specific behaviors like frequent compilation errors and editing at the same location were indicative of higher levels of frustration. This suggests that frustration can be identified through observable behaviors, providing educators with actionable insights to address these issues.

Similarly, Bosch and D'Mello (2015) investigated the affective states of novice programmers and noted that frustration, along with confusion and boredom, were prevalent among students learning programming. They highlighted that these emotional states could significantly impact the learning process, potentially decreasing motivation and academic performance. This underscores the importance of addressing emotional factors to improve educational outcomes.

Further insights into programming education reveal that tailored feedback and supportive educational tools are crucial in mitigating frustration. Tools that offer step-by-step problem-solving guidance or simplify the coding process, like educational microworlds or games, have been shown to enhance engagement and reduce frustration (Xinogalos et al., 2015). This approach also promotes a deeper understanding of programming concepts, potentially leading to more positive learning outcomes. Given the recognized issue of frustration in programming and the effectiveness of various educational tools in addressing it, one could argue that providing GenAI tools to students could reduce or eliminate this frustration. However, an important question arises: do GenAI tools, given their unique capabilities, inadvertently introduce new sources of frustration for students? Our study examines and measures how student learning profiles and previous GenAI usage impact the *frustration levels of students using GenAI tools to code*. By addressing these aspects, we aim to provide a comprehensive understanding of the role of GenAI in programming education and its impact on student emotions and learning outcomes.

Confidence in Using GenAI for Programming

Confidence in learning programming is a crucial factor influencing students' ability to successfully engage with computer science education. Several studies highlight how educational approaches and psychological readiness impact this confidence. For instance, the introduction of pair programming early in a computer science course can enhance confidence among students, particularly those who initially have low confidence levels. Pairing students based on instructors' assessments of their programming confidence allows for tailored guidance, significantly improving their exercise completion rates and overall satisfaction with the learning process (Wood et al., 2013).

GenAI tools could be likened to 'pair programming'. These tools have been seen as friendly assistants within programming environments, significantly enhancing the user experience, particularly for those new to coding. This personification as a "buddy" extends beyond mere functionality, fostering a supportive and interactive workspace that promotes learning and development (Biswas, 2022). The conversational nature of ChatGPT allows it to serve as a peer with whom programmers can "talk" through their coding issues,

making the often-solitary task of programming more engaging and less intimidating. Tian et al. (2023) found that ChatGPT not only enhanced the understanding and retention of programming principles but also boosted students' confidence as they progressed through programming courses. Hence, one could argue that using a GenAI tool would be parallel to pair programming and thus would improve the students' ability to write code. We posit that the confidence of the students' in using GenAI tools would impact their ability to write code.

Recent research that examined the confidence of students in using GenAI tools found that confidence increased with experience. The same research also found that a subset of students who had never used GenAI tools felt confident in using them (Kelly et al, 2023). This research examined the use of AI tools across all applications and did not restrict it to computer science education or more specifically, to programming courses.

However, previous studies did not consider the students' learning styles nor whether the confidence increase was towards programming or using a GenAI tool to code. Our study examines and measures how student learning profiles impact the *confidence levels of students using GenAI tools to code*.

VARK: Student Learning Styles

Research on student learning has evolved into sophisticated models to enhance interaction with educational content. These models integrate various student-related factors, including personality traits, cognitive processing strategies, and classroom behaviors. For instance, Halawa et al. (2015) highlighted the benefits of e-learning systems customized to students' psychological types, which boost engagement and learning effectiveness. Similarly, Jindal and Samanta (2013) explored how preferences in information processing impact learning outcomes. Yao et al. (2022) analyzed the interplay between teaching behaviors and student receptivity, emphasizing the significance of interpersonal interactions in educational success. Early work by Fleming and Mills (1992) investigated how individual information acquisition preferences influence learning pathways.

This study focuses on the instructional implications of these models in programming education, examining the effects of aligning teaching strategies with VARK (Visual, Aural, Read/write, Kinesthetic) learning styles on student confidence and frustration when using GenAI tools. The VARK learning styles model, developed by Neil D. Fleming in 1987 and later validated by Fitkov-Norris and Yeghiazarian in 2015, helps faculty develop material that aligns with the four different sensory modality preferences: Visual, Aural, Read/write, and Kinesthetic. Each modality represents how a student prefers to learn information. For instance, a visual learner benefits from diagrams, figures, and pictures; an aural learner from lectures and recordings; a read/write learner from textbooks and other reading materials; and a kinesthetic learner from lab work, workshops, and experiential learning. Students generally possess multimodalities, may switch from one modality to another, or develop a different dominant modality over time (Svinicki & Shi, 2010). Ultimately, the VARK learning style model allows educators to develop frameworks and consider learning styles as a foundation for their materials (Jindal et al., 2013).

Learning Modality of Generative AI Tools

Given this study examines the impact of students' learning styles on their levels of frustration with and confidence in using generative AI tools for programming, it is essential to consider the modalities of GenAI tools in relation to the four modalities outlined by the VARK learning styles framework. GenAI tools, such as ChatGPT, primarily engage users through text-based interactions, which involve reading and writing.

This characteristic aligns with the read/write modality that emphasizes learning through written language (Harshvardhan et al., 2020).

In addition, while Ebert et al. (2023) have argued that a limited kinesthetic modality exists within the programming environment, in the realm of programming, 'doing' often involves the development and execution of code. Although these tools can generate and execute code to some extent, Ebert et al. argued that they do not provide a hands-on environment for testing and running code independently. Instead, students must transfer the code to their development environments to execute it. However, Wilson and Nishomoto (2023) argued the 'doing' in the coding assignments supported a kinesthetic modality. They found students who engaged in in-person sessions with GenAI tools showed better understanding and performance in coding assignments.

Therefore, while we can assert the primary educational value and utility of GenAI tools to the read/write modalities, we also posit that GenAI tools possess an inherent kinesthetic modality for the domain of coding.

Ethics Statement

This study was approved by our Institutional Review Board (IRB). We have no conflicts of interest to declare.

Methodology

The study included 81 business graduate students from three different sections of an introduction to programming class that focuses on teaching Python and SQL for data science purposes. Students were taught SQL in the first half of the semester and Python in the second half. The teaching style and material included lectures, hands-on exercises, and supplementary materials outside of the class. The course encouraged and enforced the usage of GenAI tools. While most students defaulted to using ChatGPT-4, there were no restrictions placed on the type of GenAI tool to be used.

For this study, students were asked to complete a pre and post survey in which their learning styles were assessed using the VARK questionnaire, which evaluates preferences across four dimensions: Visual, Aural, Read/Write, and Kinesthetic. Similarly, in the pre-survey, their GenAI usage was measured by their frequency of using GenAI tools for various tasks such as academic assistance, writing aid, coding help, language learning, career advice, artistic inspiration, mental health support, entertainment, daily life assistance, and news and information. The pre-survey was sent to students via email before the semester started, asking them to complete it. After completion of the course, a survey was administered to collect data on student's frustration and confidence with using a GenAI tool to code. There were 66 students who completed the pre and post surveys. There were 48 students (32 males and 16 females) who completed and provided their VARK learning styles. Thus, for our final models, our *n* was 48.

In particular, the confidence question aimed at capturing their confidence in using a GenAI tool for programming: *'I feel confident in my ability to use ChatGPT in the future to help me write Python programs'*. The frustration question aimed at capturing the frustration of using the GenAI tool for programming: *'Using ChatGPT to learn Python was frustrating'*. Both questions were rated on a 5-point agreement Likert scale (Strongly Disagree, Disagree, Nether Agree nor Disagree, Agree, Strongly Agree).

To create the profiles, we used K-Means Clustering in Python to create student learner profiles encompassing all learning modalities, grouping students based on similar learning preferences. Similarly, clusters were created to profile GenAI usage, focusing on the frequency and type of tasks for which students use GenAI. Both types of student clusters served as the independent variables in the regression analysis models, with confidence and frustration as the dependent variables.

Results

We used K-Means Clustering to determine optimal number of clusters for GenAI usage and used the elbow method and silhouette score. We determined that the optimal number of clusters for GenAI usage was four. The elbow method showed a noticeable bend at four clusters, where the inertia dropped sharply from 300 to 200, indicating diminishing returns on variance explained with additional clusters. Similarly, the silhouette score, which measures the quality of clustering, dipped and stabilized at around four clusters with a value of approximately 0.24, suggesting that this number of clusters provided the best balance between cluster cohesion and separation.

Similarly, based on the elbow method and silhouette score, we determined that the optimal number of clusters for student learning profiles using the VARK scores was also four. The elbow method was seen to be around 3 or 4, with the inertia dropped sharply from 150 to 90 at four clusters, indicating diminishing returns on variance explained with additional clusters. Similarly, the silhouette score dipped and stabilized at four clusters with a value of approximately 0.25, suggesting that four clusters provided the best balance between cluster cohesion and separation.

The four resulting clusters for GenAI usage are labeled in Figure 1:

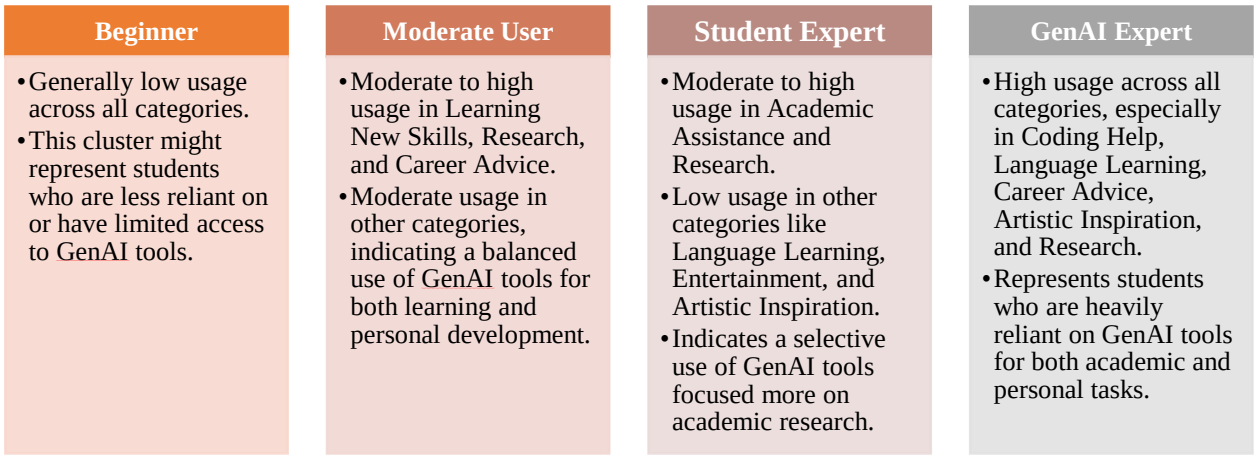

Figure 1: GenAI Usage Profiles

A Beginner is a student who is an infrequent user of GenAI tools across all the different tasks. Similarly, there is an increase in frequency of using GenAI tools from Beginner to a Moderate User. Student Experts and GenAI Experts are frequent users of GenAI tools but for different purposes. A Student Expert frequently uses GenAI tool for academic purposes while a GenAI expert frequently uses GenAI tools for all the tasks.

The four resulting clusters for student learner profiles are labeled in Figure 2:

Minimalist Learner	Listener Doer	Seer Doer	Comprehensive Learner
• **Visual**: Low (4.18) • **Aural**: Moderate (7.09) • **Read/Write**: Low (2.82) • **Kinesthetic**: Moderate (6.91) • **Profile**: Generally lower preference for all learning styles, least reliant on read/write.	• **Visual**: Moderate (9.88) • **Aural**: High (13.13) • **Read/Write**: Low (4.25) • **Kinesthetic**: High (14.25) • **Profile**: Prefers auditory and kinesthetic learning modes, less reliant on visual and read/write methods.	• **Visual**: Moderate (9.45) • **Aural**: Low (6.50) • **Read/Write**: Moderate (4.95) • **Kinesthetic**: Moderate to High (10.55) • **Profile**: Visual and kinesthetic learner, less engaged through aural and read/write methods.	• **Visual**: High (11.89) • **Aural**: High (11.89) • **Read/Write**: Moderate to High (9.22) • **Kinesthetic**: High (13.11) • **Profile**: Strong all-around learner with no specific aversion to any learning style, slightly more visual/aural.

Figure 2: Student Learner Profiles

From the clusters, a general learner profile for each cluster emerged and was labeled accordingly. A Minimalist Learner scored low across all four modalities for information processing while a Comprehensive Learner can learn from any of the modalities equally well. A Listener Doer learner relies primarily on aural and kinesthetic means of learning (e.g. lectures and hands-on practical assignments) while a Seer Doer relies on visual and kinesthetic modalities (e.g. figures and hands-on practical assignments). As such, the following figures show the student distribution for each of the clusters (Figure 3):

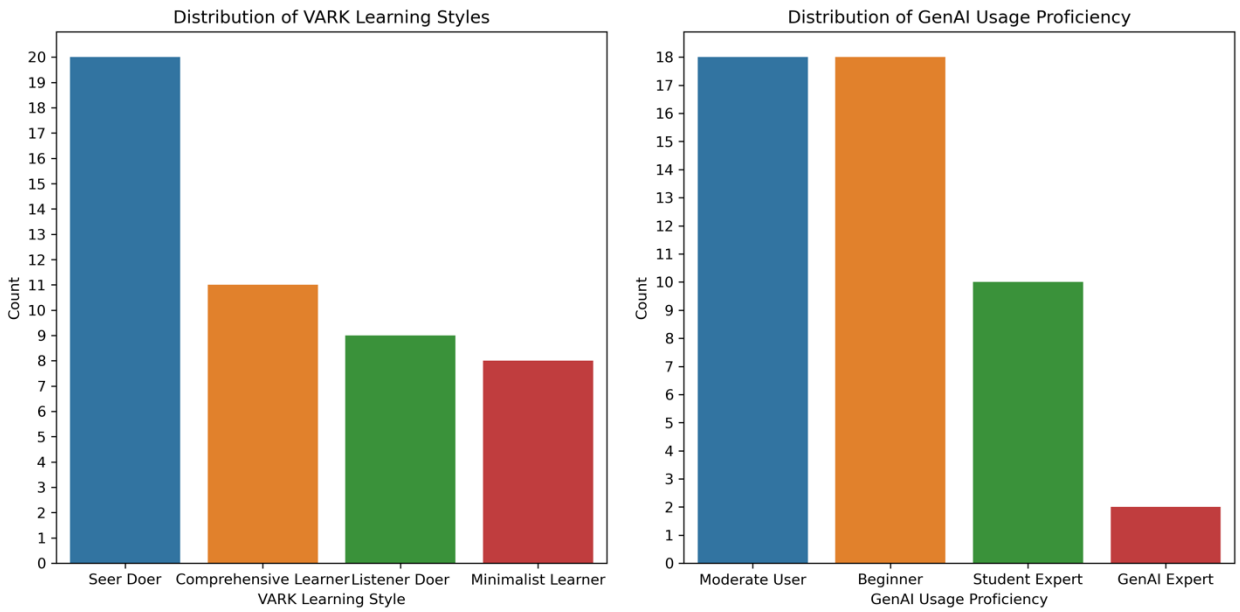

Figure 3: Student Distribution of Vark Learning Styles and GenAI Usage Proficiency

Confidence in Using GenAI for Programming

Multiple linear regression was used to test if our eight independent variables (Minimalist Learner, Listener Doer, Seer Doer, Comprehensive Learner, Beginner, Moderate User, Student Expert, and GenAI Expert) predicted 'Confidence' in using GenAI for programming.

Table 1: 'Confidence' OLS Regression Model Results

R-squared:	0.2690	F-statistic:	2.5170
Adj. R-squared:	0.1620	Prob (F-statistic):	0.0363
	Coefficient	**Standard Error**	**P>|t|**
Const	2.8746	0.1357	0.0000
Minimalist Learner	0.2018	0.2846	0.4824
Seer Doer	1.1243	0.2059	0.0000
Listener Doer	0.5211	0.2756	0.0657
Comprehensive Learner	1.0274	0.2555	0.0002
Beginner	0.1226	0.2421	0.6152
Moderate User	0.7029	0.2442	0.0063
Student Expert	1.2463	0.2835	0.0001
Gen AI Expert	0.8027	0.5377	0.1431

We found that:

- Comprehensive Learner strongly predicted confidence (β = 1.0274, p = 0.0002).
- Seer Doer strongly predicted confidence (β = 1.1243, p = 0.0000).
- Moderate User predicted confidence (β = 0.7029, p = 0.0063).
- Student Expert strongly predicted confidence (β = 1.2463, p = 0.0001).

Frustration in Using GenAI for Programming

Multiple linear regression was used to test if our eight independent variables (Minimalist Learner, Listener Doer, Seer Doer, Comprehensive Learner, Beginner, Moderate User, Student Expert, and GenAI Expert) predicted 'Frustration' in using GenAI for programming.

Table 2: 'Frustration' OLS Regression Model Results

R-squared:	0.1990	**F-statistic:**	1.7000
Adj.R-squared:	0.0820	**Prob (F-statistic):**	0.1450
	Coefficient	**Standard Error**	**P>|t|**
Const	1.3134	0.1541	0.0000
Minimalist Learner	0.2996	0.3231	0.3592
Seer Doer	-0.0874	0.2338	0.7105
Listener Doer	0.8720	0.3128	0.0080
Comprehensive Learner	0.2292	0.2900	0.4339
Beginner	1.0286	0.2748	0.0006
Moderate User	0.2411	0.2772	0.3895
Student Expert	0.2495	0.3218	0.4426
GenAI Expert	-0.2057	0.6104	0.7378

We found that:

- Listener Doer predicted frustration (β = 0.8720, p = 0.0080).
- Beginner strongly predicted frustration (β = 1.0286, p = 0.0006).

We tested for gender effects and found no significant effects in any of the models.

Discussion

In this study, we examined the predictors of student frustration and confidence in using generative AI tools to learn programming, identifying distinct learner profiles as significant predictors in these domains. From the perspective of learning styles, our analysis revealed that the Listener Doer profile predicted frustration, with a regression coefficient (β) of 0.8720 and a p-value of 0.0080 ($p < 0.01$). This finding makes intuitive sense if we look at GenAI as a tool that supports visual learners more than auditory learners. Further, although GenAI tools can support "reading" of outputs out loud to users, this feature is not beneficial when it comes to learning programming. Research has shown that frustration with programming is common, and it appears that GenAI increases frustration in learners that rely more on auditory learning.

Interestingly, none of the other learner profiles found GenAI frustrating. It is not surprising that Comprehensive Learner and Seer Doer categories experienced no frustration in the process of learning to create code using GenAI tools. The former is adept at learning well in multiple different ways and GenAI appears to enhance their learning by supporting one or more modes of learning, especially, visual, reading, and kinesthetic. The Seer Doer group appears to be adequately supported by GenAI in the two of its strongest learning modes, visual and kinesthetic. Prior research (Baer et al., 2023) states that GenAI tools offer limited support for kinesthetic learners arguing that GenAI does not offer an integrated environment for creating code and testing. The code is created in one platform (e.g., ChatGPT) and needs to be tested in a different coding platform (e.g., Jupyter.py). However, if kinesthetic learners learn from "doing", the use of different platforms (and not an integrated environment) does not appear to matter. If the advantage of having an integrated environment is the superior support for the workflow, many of the coding platforms now integrate GenAI tools (e.g., Replit now has an AI-assistant built-in). Future research should examine the role of GenAI in these integrated environments.

It is surprising that the Minimalist Learners, who are not strong in any of the learning modes, are also not frustrated when learning coding using GenAI tools. It is possible that the ability to iterate quickly and easily with GenAI, provides these learners with the ability to experiment with different alternatives to solve their coding problems. This might explain the lack of frustration in the learning process.

From the perspective of GenAI usage, the Beginner profile was a significant predictor of frustration, yielding a β of 1.0286 and a p-value of 0.0006 ($p < 0.001$). This result is typical of users that are trying GenAI for the first time. As research has shown that learning code can cause frustration, having an alternate learning environment that is not completely familiar appears to add to the frustration. Hence the Beginners found the GenAI learning environment significantly more frustrating due to the added complexity of learning how to use the GenAI tool besides learning how to create code. We did not find any evidence that the others (Moderate User, Student Expert, and GenAI Expert) experienced any frustration using GenAI to create code. This result appears to indicate that some familiarity with the GenAI environment makes it easier for learners to focus more on using GenAI to create code, instead of trying to learn/master both at the same time. This is further supported by the negative coefficient for GenAI Experts that points to a reduction in frustration (though statistically insignificant) with using GenAI to create code. The other two categories did experience frustration that is not statistically significant.

Various learner types were found to significantly predict confidence in their ability to code using GenAI tools. The Comprehensive Learner profile had a β of 1.0274 ($p = 0.0002$) and the Seer Doer profile had a β of 1.1243 ($p < 0.0001$). These results indicate that these learners felt confident in using GenAI tools to create code. This confirms the findings from previous research that GenAI acts like a partner or buddy that helps with their coding ability for these learner classes. Neither Minimalist Learner nor the Listener Doer profiles felt confident in their abilities to code with GenAI. When we connect this finding back to our

finding with frustration, the Listener Doer profile found GenAI frustrating to use when learning how to create code. This may contribute to the lack of confidence in their ability to create code using GenAI tools. However, the Minimalist Learner profile did not exhibit any statistically significant frustration using GenAI to learn and create code. In this case, frustration does not explain the lack of confidence. A possible explanation for the lack of confidence here may be that the learners did not understand the code solution they obtained using GenAI. In other words, while they were able to identify a solution for the coding problem using GenAI, they did not learn much from it, explaining the lack of frustration. The code segment remained a black box that solved the problem but did not enhance their understanding of either the code or the process of creating the right code. Hence, they lacked confidence in their ability to create code using GenAI tools.

The Moderate User profile had a β of 0.7029 ($p = 0.0063$), and the Student Expert profile demonstrated the strongest association with confidence, with a β of 1.2463 ($p = 0.0001$). These results are not surprising and suggest that these two classes of users benefited the most from GenAI support. Familiarity with using GenAI appears to help these users with their confidence in their ability to use GenAI tools for creating code confirming the findings from Kelly et al. (2023). The Moderate User exhibited some confidence ($p < 0.01$), supporting the familiarity argument. However, the GenAI Experts who should have exhibited the highest confidence did not show a statistically significant increase in confidence. This might be explained by the fact that these users were confident with their programming ability and the GenAI tool did not increase this confidence significantly. In other words, the GenAI Expert is a technically savvy group that is already confident with their ability to code and GenAI has no significant additional impact. Finally, the Beginner profile had the lowest coefficient with respect to confidence in their ability to create code with GenAI tools. This confirms that GenAI familiarity (or the lack thereof) does play a critical role in the confidence of learners use these tools to create code. Another possible explanation why Beginners and GenAI Experts did not gain confidence is that the workflow did not support learning sufficiently due to the disruptive process. Students had to go back and forth between the GenAI tool (e.g., ChatGPT) and the programming environment (e.g., Jupyter notebooks), which probably minimized the learning experience and therefore confidence.

These findings suggest that learner typologies significantly influence both the positive and negative psychological responses to the use of generative AI tools in educational settings specific to programming. The results of this study have several implications for educators aiming to teach programming using GenAI tools. First, understanding that different learning styles and levels of GenAI tool usage impact students' confidence and frustration can help educators tailor their teaching strategies more effectively. For instance, Comprehensive Learners and Seer Doer learners showed higher confidence levels, indicating that these groups might benefit more from GenAI tool integration in their learning processes. In terms of implications, this means that integration of GenAI within the programming environment (IDE) is important for the user experience and learning.

However, the significant frustration experienced by Listener Doer learners and Beginners suggests that these students might need additional support learning GenAI tools. Educators need to understand the types of frustration their students could face while using GenAI tools for programming. In our case, we asked a small group of students the frustrations they faced while using GenAI tools and found the following:

1. *Expectations Not Met:* Users might expect GenAI tools to understand and execute complex instructions flawlessly, but when these tools failed to do so, it led to significant frustration. This usually resulted in mismatch of the what the students hoped the code would accomplish and the prompting they provided.

One of the students mentioned the following: *"ChatGPT has a mind of its own, which made it more difficult for me to complete this assignment."*

In this instance, the student was referring to an assignment in which the students were asked to modify existing code to make it more modular and dynamic.

2. *Debugging Complexities:* While GenAI tools helped, they did not always provide accurate debugging help, leading to further confusion and frustration for users trying to resolve coding issues.

   At the time of writing for this paper, students reported debugging only worked well for commonly used or well-known libraries (e.g. pandas). However, libraries that were updated, outdated or not frequently used (e.g. tensorflow, tweepy, or gnews), GenAI tools failed to help debug accurately. In addition, when students encountered technical issues where libraries required drivers to be installed correctly, GenAI tools were unable to identify and help with these problems.

3. *Lack of Context Understanding:* GenAI tools struggled with understanding the broader context of a coding project, offering suggestions that were technically correct but contextually irrelevant, which created frustration for users.

   As an example, the students were provided with a dataset containing flight information from a specified period. This dataset included expected and actual departure and arrival times, along with other flight-related details such as the airline. The students were tasked with calculating flight delays, identifying which airlines had the most delays overall, and determining if there were any patterns, such as seasonal delays. Upon completion of the assignment, students compared their results with those of their classmates and discussed any discrepancies that arose.

   Many students reported that the GenAI tool 'misunderstood' the task and provided incorrect answers. Upon inspecting their prompts and the resulting code, it was found that the GenAI had calculated 'delays' differently than expected, such as calculating delays per day instead of per airline. Students realized they needed to provide precise GenAI prompts to clarify how to calculate a 'delay', etc. to ensure a correct response.

4. *Over-reliance and Skill Degradation:* Some students reported they became overly reliant on GenAI tools, which hindered the development of their problem-solving skills and led to frustration when they needed to solve problems independently.

   The students were asked to complete a midterm that was carefully crafted to force students to think about how to solve the problem versus generate code with a GenAI tool. After completing the exam, a student reflected, *"I started by relying on ChatGPT to do all the thinking for me, but when that didn't work, I had already run out of time."*

5. *Disruptive Workflow:* At the time of writing this paper, GenAI tools did not integrate well with the coding environment. A typical workflow for a student using a GenAI tool was as follows:

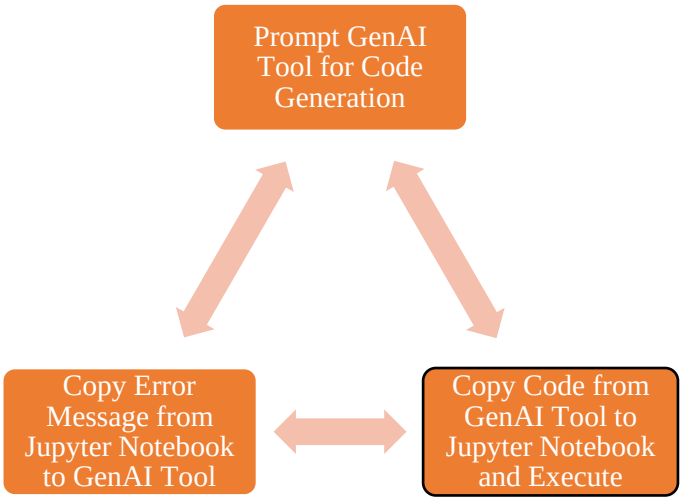

Figure 4: Typical Workflow for Code Generation using a GenAI Tool

This workflow required a student to switch from one tool to another which increased frustration for many students. However, it is conceivable as the GenAI tools advance, they will be able to integrate a coding environment or many coding environments will integrate GenAI tools (e.g. Visual Studio Code already has GitHub Copilot integrated in its environment). At the time of writing this paper, Jupyter notebooks did not natively integrate any GenAI capabilities.

In conjunction with the results of this study, educators can consider providing additional sessions for the Listener Doer learner and Beginners of GenAI that target specifically on how to use GenAI tools for programming. This could include supplementary resources and strategies on prompt engineering that showcase specific examples.

Frustration and Confidence continue to be common constructs used to predict and assess likelihood of future coding and programming (Drosos, Guo, & Parnin, 2017, Putnam, Deng, & DeSoto, 2022). As such, it has been long recognized that educators need to provide a learning experience that considers the learning styles of the students. Given the availability and the ease of access to GenAI tools, educators must not only consider learning styles but also the familiarity with and the impact of GenAI tools for teaching coding/programming. The more we familiarize the GenAI tools and their appropriate use, the better the learning experience and learning success might be. As educators, particularly in coding/programming classes, we must devote time to familiarize students with the GenAI tools. It is mostly a certainty that they will use it whether we recommend it or not. For non-technical students such as those majoring in business and related areas, the more they are familiarized with GenAI tools, the better they will get at learning and creating code. The impact of GenAI tools in programming education is going to be significant. Our study provides a step towards that understanding for educators.

Conclusion

In this study we examined the impact of GenAI tools in the context of learning programming/coding. Understanding from prior research that learning coding can contribute to frustration, we studied whether this frustration can be mitigated with the use of GenAI tools. We further examined whether the use of GenAI tools can help with the confidence of students in their ability to create code. Using learning styles (VARK) and familiarity with using GenAI (GenAI-usage) as indicators, we created four clusters based on

learning styles and four other clusters based on GenAI usage. Treating these clusters as independent variables, we examined their impact on frustration and confidence. Our study identifies that Seer Doer and Comprehensive Learner profiles, who excel with visual and kinesthetic modalities, along with GenAI Moderate Users and Student Experts, experienced increased confidence. Listener Doer learner profiles experienced frustration and so did GenAI Beginners.

Today, learning how to code is sought after by a very large body of students that major in business and other areas not typically associated with this skill. Leveraging GenAI to educate such students, who may not have an inclination, or the necessary technical background, is an obvious choice. This study highlights the need for tailored educational strategies that align GenAI toolswith the specific learning styles and previous GenAI experiences, thereby ensuring that students learn without frustration and enter the world confident in their ability to use GenAI tools to create and learn programming.

Future research should aim to increase the sample size to ensure the results are more generalizable. Additionally, including previous programming experience as a control variable could provide further insights into how prior knowledge impacts the effectiveness of GenAI tools in programming education. This could help in identifying whether experienced programmers benefit differently from GenAI tools compared to beginners, allowing for more personalized and effective teaching strategies. Finally, future GenAI tools aim to support multiple modalities, allowing users to input and receive output in the form of text, audio, visuals, and videos. Therefore, further research on how these multi-modality GenAI tools support different programmer learner profiles is needed.

By addressing these aspects, future studies can contribute to a more comprehensive understanding of the role of GenAI tools in programming education and help educators develop more effective, tailored teaching methods.

References

Becker, B. A., Craig, M., Denny, P., Keuning, H., Keisler, N., Leinonen, J., Luxton-Reilly, A., Malmi, L., Prather, J., & Quille, K. (2023). *Generative AI in introductory programming*. Computing Curricula 2023 – Report by the joint task force on computing curricula ACM, IEEE-CS, and AAAI. http://csed.acm.org/wp-contents/uploads/2023/03/version-beta-v2.pdf

Bosch, N., & D'Mello, S. (2015). The affective experience of novice computer programmers. *International Journal of Artificial Intelligence in Education, 27*, 181-206.

Bringula, R. (2024). ChatGPT in a programming course: Benefits and limitations. *Frontiers in Education, 9*. https://www.frontiersin.org/articles/10.3389/feduc.2024.1248705

Chen, L., Chen, P., & Lin, Z. (2020). Artificial intelligence in education: A review. *IEEE Access, 8*, 75264–75278. https://doi.org/10.1109/ACCESS.2020.2988510

Chilana, P. K., Alcock, C., Dembla, S., Ho, A., Hurst, A., Armstrong, B., & Guo, P. J. (2015). Perceptions of non-CS majors in intro programming: The rise of the conversational programmer. *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 251-259. https://doi.org/10.1109/VLHCC.2015.7357224

Deniz, B. K., Gnanasambandam, C., Harrysson, M., Hussin, A., & Srivastava, S. (2023). *Unleashing developer productivity with generative AI*. McKinsey. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai

Drosos, I., Guo, P. J., & Parnin, C. (2017). HappyFace: Identifying and predicting frustrating obstacles for learning programming at scale. *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 171-179. https://doi.org/10.1109/VLHCC.2017.8103463

Ebert, C., Louridas, P., & Ebert, C. (2023). Generative AI for software practitioners. *IEEE Software, 40*(1), 30-38.

Fitkov-Norris, E., & Yeghiazarian, A. (2015). Validation of VARK learning modalities questionnaire using Rasch analysis. *Journal of Physics: Conference Series, 588*, 012048. https://doi.org/10.1088/1742-6596/588/1/012048

Fleming, N. D., & Mills, C. (1992). Not another inventory, rather a catalyst for reflection. *To Improve the Academy, 11*, 137-155. https://doi.org/10.1002/j.2334-4822.1992.tb00213.x

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research, 87*(4), 834-860. https://doi.org/10.3102/0034654317710096

Gartner. (2021). Gartner says the majority of technology products and services will be built by professionals outside of IT by 2024. Stamford, CT. https://www.gartner.com/en/newsroom/press-releases/2021-06-10-gartner-says-the-majority-of-technology-products-and-services-will-be-built-by-professionals-outside-of-it-by-2024

Gomes, A., & Mendes, A. (2014). A teacher's view about introductory programming teaching and learning: Difficulties, strategies, and motivations. *2014 IEEE Frontiers in Education Conference (FIE) Proceedings*, 1–8. https://doi.org/10.1109/FIE.2014.7044086

Halawa, M. S., Hamed, E. M. R., & Shehab, M. E. (2015). Personalized e-learning recommendation model based on psychological type and learning style models. *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, 578-584. https://doi.org/10.1109/INTELCIS.2015.7397281

Harshvardhan, G. M., Gourisaria, M. K., Pandey, M., & Rautaray, S. (2020). A comprehensive survey and analysis of generative models in machine learning. *Computational Science Review, 38*, 100285.

Hu, K. (2023, February 2). ChatGPT sets record for fastest-growing user base. *Reuters*. https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/

Jacques, L. (2023). Teaching CS-101 at the dawn of ChatGPT. *ACM Inroads, 14*(2), 40–46. https://doi.org/10.1145/3595634

Jindal, M., Kharb, P., & Samanta, P. (2013). Comparative analysis of instructional learning preferences of medical students of first and seventh semester. *International Journal of Physiology, 1*, 32-36.

Kelly, A., Sullivan, M., & Strampel, K. (2023). Generative artificial intelligence: University student awareness, experience, and confidence in use across disciplines. *Journal of University Teaching & Learning Practice, 20*(6), 12.

Leite, W., Svinicki, M., & Shi, Y. (2010). Attempted validation of the scores of the VARK: Learning styles inventory with multitrait–multimethod confirmatory factor analysis models. *Educational and Psychological Measurement, 70*, 323-339. https://doi.org/10.1177/0013164409344507

Liu, R., Zenke, C., Liu, C., Holmes, A., Thornton, P., & Malan, D. J. (2024). Teaching CS50 with AI: Leveraging generative artificial intelligence in computer science education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2024)* (p. 1927). Association for Computing Machinery. https://doi.org/10.1145/3626253.3635427

Putnam, A., Deng, W., & DeSoto, K. (2022). Confidence ratings are better predictors of future performance than delayed judgments of learning. *Memory, 30*(5), 537-553.

Rawat, S., Makwana, K. K., Pathak, R. R., & Rathod, N. M. (2023). Identification of preferred learning styles as per VARK model in the undergraduate medical students. *National Journal of Physiology, Pharmacy and Pharmacology, 13*(10), 2049-2054. https://doi.org/10.5455/njppp.2023.13.06280202207032023

Rodrigo, M. M. T., & Baker, R. S. J. d. (2009). Coarse-grained detection of student frustration in an introductory programming course. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop (ICER '09)* (pp. 75–80). Association for Computing Machinery. https://doi.org/10.1145/1584322.1584332

Wilson, S. E., & Nishimoto, M. (2024). Assessing learning of computer programming skills in the age of generative artificial intelligence. *Journal of Biomechanical Engineering, 146*(5), 051003. https://doi.org/10.1115/1.4064364

Xinogalos, S., Satratzemi, M., & Malliarakis, C. (2015). Microworlds, games, animations, mobile apps, puzzle editors, and more: What is important for an introductory programming environment? *Education and Information Technologies, 22*, 145-176.

Yao, T., & Yang, X. (2022). Analysis of the association between teachers' classroom teaching behaviors and students' knowledge acceptance based on psychological data analysis. *Occupational Therapy International, 2022*. https://doi.org/10.1155/2022/2661398

Yilmaz, R., & Yilmaz, F. G. K. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy, and motivation. *Computers and Education: Artificial Intelligence, 4*, 100147. https://doi.org/10.1016/j.caeai.2023.100147

Yusoff, K. M., Ashaari, N. S., Wook, T. S. M. T., & Ali, N. M. (2020). Analysis on the requirements of computational thinking skills to overcome the difficulties in learning programming. *International Journal of Advanced Computer Science and Applications (IJACSA), 11*(3). https://doi.org/10.14569/IJACSA.