Book Chapter

Can Generative Artificial Intelligence Outperform Self-Instructional Learning in Computer Programming?: Impact on Motivation and Knowledge Acquisition

Rafael Mellado[1]* and Claudio Cubillos[2]

[1]Escuela de Comercio, Pontificia Universidad Católica de Valparaíso, Valparaíso 2340025, Chile
[2]Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Valparaíso 2340025, Chile

***Corresponding Author: Rafael Mellado,** Escuela de Comercio, Pontificia Universidad Católica de Valparaíso, Valparaíso 2340025, Chile

Published **October 30, 2025**

This Book Chapter is a republication of an article published by Rafael Mellado and Claudio Cubillos at Applied Sciences in May 2025. (Mellado, R.; Cubillos, C. Can Generative Artificial Intelligence Outperform Self-Instructional Learning in Computer Programming?: Impact on Motivation and Knowledge Acquisition. Appl. Sci. 2025, 15, 5867. https://doi.org/10.3390/app15115867)

How to cite this book chapter: Rafael Mellado, Claudio Cubillos. Can Generative Artificial Intelligence Outperform Self-Instructional Learning in Computer Programming?: Impact on Motivation and Knowledge Acquisition. Top 10 Contributions in Applied Sciences. Hyderabad, India: Academic Reads. 2025.

Author Contributions: Conceptualization, R.M. and C.C.; methodology, R.M. and C.C..; soft-ware, R.M.; validation, R.M., and C.C.; formal analysis, C.C.; investigation, R.M. and C.C.; resources, R.M. and C.C.; data curation, C.C.; writing— original draft preparation, R.M.; writing—review and editing, C.C.; visualization, R.M. and C.C.; supervision, C.C.. All authors have read and agreed to the published version of the manuscript.
Funding: This research received no external funding.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki, and the protocol was approved by the Bioethics and Biosecurity Committee of Pontificia Universidad Católica de Valparaíso (protocol code BIOEPUCV-H 659-2023 on 29 June 2023).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Data are available in the article.

Conflicts of Interest: The authors declare no conflict of interest.

Abstract

Generative artificial intelligence tools, such as Microsoft Copilot, are transforming the teaching of programming by providing real-time feedback and personalized assistance; however, their impact on learning, motivation, and cognitive absorption remains underexplored, particularly in university settings. This study evaluates the effectiveness of Microsoft Copilot compared to instructional videos in teaching web programming in PHP, implementing a quasi-experimental design with 71 industrial engineering students in Chile, divided into two groups: one using Microsoft Copilot and the other following instructional videos, with pre-and post-tests applied to measure knowledge acquisition while surveys based on the Hedonic-Motivation System Adoption Model (HMSAM) assessed cognitive absorption (enjoyment, control, immersion, curiosity) and technology acceptance (perceived usefulness, ease of use, and intention to adopt). The results show that, while both methods improved learning, students who used instructional videos achieved greater knowledge gains, higher levels of curiosity, and a stronger intention to continue using the technique, suggesting that instructional videos, by providing structured explanations and reducing cognitive load, may be more effective in the early stages of programming learning. In contrast, AI tools could be more beneficial in advanced stages where students require adaptive feedback, providing empirical evidence on the comparative effectiveness of AI-based and video-based instruction in teaching programming and highlighting the importance of balancing structured learning with AI-driven interactivity, with the recommendation that educators integrate both approaches to optimize the learning experience, using videos for initial instruction and AI tools for personalized support.

Keywords

Generative Artificial Intelligence; Programming education; Cognitive absorption; Technology adoption.

1. Introduction

1.1 Background

In recent years, generative artificial intelligence (AI) tools, such as Microsoft Copilot, Google Gemini, and ChatGPT, have begun to significantly transform the educational landscape [1]. These technologies assist in solving complex problems and promote a more personalized and adaptive learning experience [2]. Tools like Microsoft Copilot have proven particularly useful in programming by offering real-time code suggestions, facilitating understanding concepts, and resolving errors [3]. However, their impact on student learning and motivation has yet to be sufficiently explored, especially within university educational contexts.

One of the main challenges in adopting technological systems in education is measuring the motivation and enjoyment that users experience when interacting with these tools. The theory of cognitive absorption suggests that immersion and enjoyment during technology use can significantly influence adoption and effective learning [4]. However, measuring these emotional and cognitive dimensions remains challenging, requiring validated instruments and rigorous methodological approaches [5]. In the case of generative AI tools, it is crucial to understand how they impact the learning experience and whether they generate positive emotional effects that enhance knowledge retention.

Learning to program web systems, particularly in languages like PHP, represents a significant challenge for university students [6-8]. The complexity of the concepts, the need for logical skills, and the steep learning curve contribute to the difficulty [9]. Universities face the challenge of designing effective pedagogical strategies that enable students to overcome these barriers. Traditionally, instructional videos have been a common tool for teaching programming, but their effectiveness compared to more interactive methods, such as generative AI, has not yet been sufficiently studied [10].

In Latin America, research on the impact of AI tools in education is still in its early stages [11,12]. Students in this region exhibit levels of digital literacy that differ from those in Europe, Asia, and the United States, which may influence the adoption and effectiveness of these technologies [13]. Additionally, there is a significant gap in the scientific literature addressing these differences and exploring how AI tools can be adapted to specific educational contexts. This study aims to contribute to closing this gap by evaluating the impact of Microsoft Copilot on the learning and motivation of university students in the Latin American context.

1.2 Theoretical Framework

Learning programming, especially in languages like PHP for web development, presents significant challenges for university students. Among the most common difficulties are the comprehension of abstract concepts, the resolution of syntactic and logical errors, and the lack of motivation to persist in the face of complex problems [9]. These barriers affect academic performance and can generate frustration and disinterest among students, hindering their learning process [14]. In response to this scenario, educational institutions have sought to implement innovative pedagogical strategies that allow these limitations to be overcome and promote more effective and motivating learning.

In recent years, active learning techniques have gained relevance in education, particularly in technical fields such as programming [15]. Instructional videos have been widely used to facilitate the understanding of complex concepts, as they allow students to learn at their own pace and review the content as many times as necessary [10]. However, these resources are often unidirectional and do not always promote active interaction between the student and the learning material.

On the other hand, generative artificial intelligence tools, such as Microsoft Copilot, represent an evolution in active learning strategies. These tools provide immediate feedback and adapt their assistance based on the individual needs of the student [3]. Generative AI fosters a more interactive and personalized approach by offering real-time code suggestions and contextualized explanations, enhancing comprehension and student motivation.

The Hedonic-Motivation System Adoption Model (HMSAM) provides a useful theoretical framework for understanding how generative AI tools like Microsoft Copilot can influence technology adoption and effective use in educational contexts. This model focuses on hedonic motivation, the pleasure or enjoyment users experience when interacting with a technology, as a key factor in its adoption [16]. According to HMSAM, cognitive absorption, which includes dimensions such as enjoyment, immersion, and curiosity, plays a crucial role in perceiving a technology's usefulness and ease of use.

The HMSAM is based on three main constructs: perceived enjoyment, perceived control, and immersion. Perceived enjoyment refers to the extent to which the use of a system is perceived as pleasurable and fun, serving as a critical predictor in hedonic contexts [17]. Perceived control reflects the sense of autonomy and mastery that the user experiences when interacting with the system, which increases their confidence and willingness to adopt the technology [18]. Finally, immersion describes a psychological state in which the user feels fully absorbed by the technological experience, a key factor in systems such as virtual reality and video games [19].

In the context of programming education, each construct of the HMSAM plays a distinct role [20,21]. For example, 'perceived usefulness' and 'ease of use' help assess how well students perceive these tools as effective in supporting coding tasks, especially important in introductory programming, where clear scaffolding is essential. Enjoyment and curiosity are crucial for maintaining engagement during error-prone debugging or logic design activities. 'Focused immersion' and 'temporal dissociation' are frequently reported by students when deeply engaged in solving complex programming challenges. 'Control' reflects the learner's perceived autonomy when navigating through AI-suggested solutions, and 'behavioral intention to use' is critical for understanding whether students would continue using AI tools beyond the classroom setting. These constructs were, therefore, selected not only for their theoretical grounding in HMSAM but also for their practical alignment with the motivational, emotional, and cognitive demands inherent to programming education [20].

In line with this, Chakraborty [22] noted that GenAI fosters human-machine collaboration, enabling personalized and adaptive learning while supporting experiential approaches within the HMSAM framework. The study highlights the integration of GenAI in curriculum design, teaching, and assessment, and its potential to equip students with key competencies for the future workforce. These constructs influence the intention to use and the effective adoption of hedonic systems. For example, in video games, perceived enjoyment and immersion are significant predictors of the intention for continued use [23]. In social media applications, perceived control and immersion explain the adoption of new functionalities [24].

In the context of programming education, HMSAM suggests that tools that generate a pleasurable and motivating user experience are more likely to be adopted and used effectively. For instance, if students find using Microsoft Copilot enjoyable and immersive, allowing them to become absorbed in the coding process, they are more likely to perceive the tool as useful and easy to use, which in turn may increase their intention for continued use [16]. This theoretical approach is particularly relevant to the present study, as it allows for the evaluation of the cognitive impact of generative AI tools on learning and their ability to foster states of immersion, enjoyment, and motivation. By considering both the utilitarian and hedonic aspects of the experience, HMSAM offers a comprehensive perspective for understanding how the adoption and continued use of technologies like Microsoft Copilot can positively influence academic performance and students' willingness to engage actively and sustainably with programming.

Furthermore, the theory of cognitive absorption can be enriched by considering knowledge dynamics and knowledge fields—rational, emotional, and spiritual—as proposed by [25-27] in the context of learning. These approaches suggest that constructs such as enjoyment, curiosity, and immersion depend on the transformation of knowledge across domains: for instance, the rational understanding of PHP programming structures may evolve into an emotional state of satisfaction or curiosity when solving practical problems or even a spiritual connection when perceiving a broader purpose in technological learning. Within the scope of this study, such transformations may occur during active interaction with Microsoft Copilot, which encourages autonomous exploration, as well as through instructional videos, which provide structured guidance, thereby enhancing the adoption and effectiveness of both tools under the HMSAM framework.

1.3 Objectives and Research Questions

The present study aims to evaluate the capacity of a generative AI (Microsoft Copilot) compared to an instructional video to generate learning and positive emotional effects when university students practice web programming topics in PHP.

The research questions to be considered in this study are:
RQ1: What differences in learning exist between students who practice web programming using Microsoft Copilot versus an instructional video?
RQ2: What differences exist between students who practice web programming using Microsoft Copilot and those using an instructional video regarding cognitive absorption effects: enjoyment, control, focused immersion, temporal dissociation, and curiosity?
RQ3: What differences in effects exist between students who practice using Microsoft Copilot and those using an instructional video in the dimensions of technology acceptance: ease of use, perceived usefulness, and intention to use?

2. Related Works

2.1 Educational Models, Motivation, and Technology Acceptance

The flipped classroom model is a widely researched approach in current education, which has garnered interest for its potential to foster active student participation. Bishop et al. [28] conducted a study that categorized various research on this model based on in-class and out-of-class activities, assessment systems, and methodological strategies. Although a positive perception was observed among students, the results revealed a lack of robust empirical evidence supporting significant improvements in academic performance. In a complementary study, Sung et al. [29] demonstrated that including mobile devices can enhance learning outcomes when combined with interactive teaching methods. However, Bernard [30] provided a different perspective by showing that, in certain contexts, asynchronous interaction in distance education environments can be even more effective than face-to-face interaction.

Digital technologies, beyond their use in the flipped classroom, have also been analyzed in the field of education. Gordillo [31], for example, compared the effects of game-based learning with the use of videos in software engineering courses, concluding that games designed by instructors generate better content retention outcomes. In line with this, Lowry et al. [16] highlighted "cognitive absorption" as a key factor that fosters motivation in hedonic educational systems. However, similar to the case of flipped methodologies, Bernard et al. [30] emphasized the heterogeneity of these findings, noting that not all technologies or strategies exhibit the same level of effectiveness.

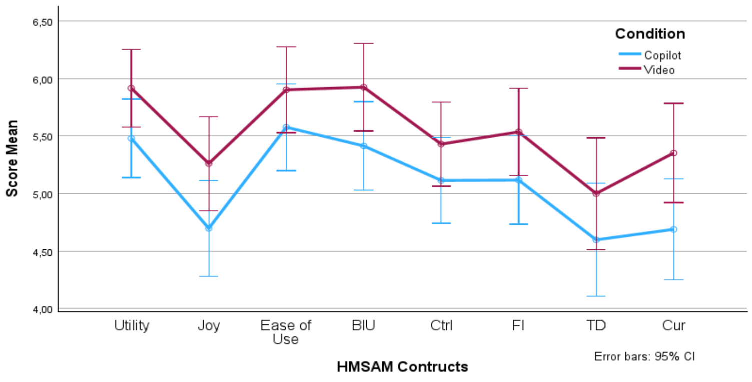
Motivation in learning is another area of study that has received special attention. From the self-determination theory perspective, Deci and Ryan [32] emphasized the importance of intrinsic and extrinsic motivation in academic performance and student well-being. This theoretical framework aligns with the findings of Jena et al. [33], who revealed substantial improvements in self-regulation and academic performance when using Web 2.0 tools for collaborative learning. For instance, a collaborative platform facilitated continuous interaction in a language course, demonstrating increased student participation and motivation. Similarly, Huang and Mizumoto [34] showed that ChatGPT use in EFL classrooms enhanced students' intrinsic motivation and writing self-efficacy when structured guidance was provided, reinforcing the motivational benefits of GAI in educational settings.

Although Jena et al. [33] attribute success primarily to social interaction, Litman [35] argues that individual curiosity (conceptualized as a driver of personal inquiry) emerges as the true catalyst in knowledge acquisition. Krouska et al. [36] found that generative AI tools like ChatGPT enhance student motivation by promoting enjoyment, effort, outcome evaluation, perceived relevance, and interaction. These effects stem from the chatbot's conversational and social features, which foster quality engagement and positively influence academic performance. These differences underscore the need to delve deeper into the contextual factors that may modulate motivation. Similarly, Dousay [37] examined how the design of multimedia resources impacts student motivation, highlighting the importance of having reliable measurement tools. Meanwhile, Lowry et al. [16] proposed the Hedonic-Motivation System Adoption Model (HMSAM), in which cognitive absorption (understood as total immersion in an activity) directly influences the intention to use playful educational systems. Even so, Abdelshiheed et al. [38] emphasized that metacognition and motivation in intelligent tutoring systems are also essential for preparing for future learning.

In the field of technology acceptance, Davis [39] was a pioneer in describing how perceived usefulness and ease of use influence the adoption of digital systems. Ghimire & Edwards [40] emphasized that generative AI adoption in educational settings is closely tied to perceived usefulness and ease of use, core constructs of the Technology Acceptance Model (TAM). Educators are more likely to adopt GenAI when it enhances teaching effectiveness and is user-friendly, underlining the need for supportive integration strategies. Later, Venkatesh et al. [41] integrated multiple theories to formulate the Unified Theory of Acceptance and Use of Technology (UTAUT), which has demonstrated robustness in identifying key factors in adopting technological tools. Both Davis and Venkatesh agree that the perception of ease of use plays a determining role in technology acceptance, a finding also supported by Steinert et al. [42], who used advanced language models to provide formative feedback and foster self-regulated learning. Lin & Ng [43] explored user motivations and concerns regarding generative AI on platforms like Reddit, identifying utilitarian, hedonic, and social gratifications, along with creativity enhancement and technical/social problems. These factors affect engagement and highlight the need for user-centered, ethically grounded AI systems that address technological capabilities and societal implications to foster broader acceptance.

In contrast, Clark and Mayer [44] posited that instructional design has a greater impact than the technological platform used, emphasizing that technology alone does not guarantee positive outcomes. While not denying the relevance of technology acceptance, this argument highlights the need to align the adoption of new tools with carefully designed pedagogical strategies. Al-Abdullatif [45] highlighted that AI literacy and perceived ease of use are key to GenAI acceptance among university instructors, mediated by smart TPACK and perceived trust. The study emphasizes the need for educators to strengthen their foundational knowledge and pedagogical adaptability to integrate GenAI technologies effectively into their teaching practices. On the other hand, cognitive load theory (with an emphasis on managing students' cognitive resources) has guided multiple instructional design proposals. Paas and Van Merriënboer [46] established that excessive cognitive load can negatively impact learning, particularly in complex tasks requiring high processing levels. Similarly, Mayer [47] formulated principles based on the cognitive theory of multimedia learning, emphasizing the need to use visual and textual elements complementarily to avoid overloading working memory.

In another study, Martins [48] demonstrated that including interactive annotations in educational videos can enhance student comprehension by focusing attention and reducing extraneous cognitive load. However, the effects of such interventions are not always uniform, as Bernard et al. [30] observed significant variations depending on the type of interaction promoted, highlighting that the implementation of asynchronous or synchronous strategies can lead to divergent outcomes. Meanwhile, Abdelshiheed et al. [38] suggest that even in the presence of well-designed multimedia, metacognition acts as a critical factor in learning transfer, which is why cognitive load should not only be mitigated but also strategically managed. Despite the advances above, gaps in the literature remain. On the one hand, some studies, such as those by Bishop & Verleger [28] and et al. [29], have primarily focused on student perceptions or short-term outcomes without providing longitudinal follow-up of the effects on academic performance. Similarly, Bernard et al. [30] highlight the heterogeneity in the effect size of interactive strategies, suggesting the need to examine in greater detail the role of context, discipline, and student characteristics.



On the other hand, deeper explorations are needed regarding how motivation, in its various dimensions (intrinsic, extrinsic, social, and curiosity-based), is modulated by factors such as institutional culture, educational level, or the nature of the subject matter. Likewise, while models such as those proposed by Davis [39] and Venkatesh et al. [41] have provided robust theoretical frameworks, Clark & Mayer [44] emphasize the need to validate these models across diverse environments with heterogeneous characteristics empirically. Finally, it is noted that metacognition and self-regulation require more specific approaches, particularly when integrating complex technologies such as intelligent tutoring systems.

Thus, the reviewed literature reveals a convergence around the relevance of motivation, instructional design, and technology acceptance as pillars of technology-mediated learning. However, divergences persist regarding the efficacy of strategies and digital tools and methodological challenges that prevent a definitive consensus. It is necessary to conduct studies with longer timeframes and greater experimental rigor to clarify the conditions under which innovative educational models and emerging technologies generate positive and sustainable impacts on learning. Only then will it be possible to develop more comprehensive, adaptable, and effective approaches within the growing educational ecosystem.

2.2 Computer Programming

Traditional methods of teaching programming, such as lectures and paper-based exercises, have been widely used in higher education [9,14,34,49]. However, these approaches are often criticized for their lack of interactivity and adaptability to the individual needs of students [49,50].

Yang et al. [51] proposed PSFinder, a tool capable of identifying coding screencasts in online videos to improve automation in software engineering. This work shares with Codemotion [52] the use of machine learning algorithms, particularly computer vision, for video processing. However, while PSFinder focuses on classifying videos to facilitate automated debugging and library recommendations, Codemotion emphasizes interactivity with programming content. As a limitation, PSFinder experiences difficulties classifying videos with large moving objects, whereas Codemotion does not evaluate its effectiveness across various video formats.

Using videos as an educational resource in teaching programming has been extensively studied. Tutorly [53] proposes interactive tutoring based on language models to enhance the learning experience for students. The tool integrates as a JupyterLab extension and guides learners through multimodal conversations that adapt to each individual's progress. One of its main contributions is the video transcript segmentation system, which achieves 73.7% accuracy within five-second margins. However, it faces challenges with lengthy videos, suggesting the need to divide content into shorter clips to optimize accuracy.

In parallel, Codewit.us [54] is a tool that combines videos with interactive coding exercises to reinforce learning. This platform was implemented in introductory programming courses at institutions such as the University of California, Santa Barbara, where the performance of 156 students was evaluated. The results indicate that those who used Codewit.us, integrating videos and practical exercises, showed significantly higher interaction frequency than those who accessed these resources separately.

While both studies highlight the importance of incorporating multiple teaching modalities, their focus differs: Tutorly uses language models for personalized tutoring, whereas Codewit.us synchronizes videos and exercises to encourage continuous practice. Together, these works suggest the need to evaluate such systems in different educational contexts and consider their integration into various platforms as future research directions.
W. Liu et al. [53] explored the impact of blended learning supported by live streaming for programming students, comparing the experiences of full-time students with those who also work. Their study, conducted at a university in Taiwan with 54 participants, revealed that working students preferred code annotations to review material at their own pace. In contrast, full-time students benefited more from flipped classrooms and video-based resources. This approach is comparable to solutions based on video-supported programming labs, as analyzed by McGowan et al. [55]. These researchers noted that interactive video-based learning environments enhance the understanding and retention of programming concepts, especially when they include practical tasks and guided exercises. However, both studies agree on the need to explore the scalability of these strategies further across diverse institutions and heterogeneous student populations.

Regarding pedagogical innovations, gamification represents another avenue of research to improve programming education. Mellado and Cubillos [56] demonstrated that using reward techniques contributes to better performance in teaching data structures. Their proposal aligns with the findings of Ferreira et al. [57], who emphasize the importance of feedback and continuous assessment in the programming learning process. While Mellado & Cubillos [56] focus on motivation through playful incentives, Ferreira et al. [57] highlight traditional pedagogical strategies to reinforce learning. Both perspectives recognize the value of active learning, leaving open the possibility of combining gamification with structured feedback in future research.

Regarding the challenges of learning programming, Kadar et al. [58] examined the difficulties faced by students without prior computing education. Their conclusions complement those of Esche and Weihe [59], who analyzed how various pedagogical foundations impact the teaching of programming. While Kadar et al. [58] identified structural issues in how programming is taught, Esche & Weihe [59] focused on the effect of video-based pedagogy on students' self-efficacy. Nevertheless, both studies agree on the need to broaden the generalization of their results to different educational levels and student profiles.

The reviewed studies show that programming education has evolved by incorporating advanced technologies, from artificial intelligence to gamification strategies and hybrid teaching models. However, common challenges persist, such as the lack of longitudinal research to assess long-term impacts and the adaptation of methodologies to diverse student profiles. Future research could focus on integrating these approaches and analyzing potential synergies between AI-based tutoring, gamification, and blended learning to enhance the effectiveness of programming education. Additionally, it is necessary to validate these approaches in varied educational contexts and with heterogeneous populations to consolidate their applicability and scalability.

2.3 Artificial Intelligence in Education

The evolution and application of generative artificial intelligence in programming have generated growing interest in academic literature [60]. Various studies have examined the impact of this technology on education, code improvement, content generation, and software development [61-64]. Recent generative AI advancements have significantly improved code quality across various programming contexts. For instance, Nettur et al. [65] found that GPT-4o, when guided by a chained few-shot prompting approach, outperformed other methods in generating Cypress automation code, excelling in completeness, syntactic accuracy, and maintainability. These findings suggest that the quality of AI-generated code is advancing rapidly, offering robust support for programming tasks when paired with effective prompting strategies.

Furthermore, generative AI tools have increasingly demonstrated their capacity to streamline programming tasks, particularly in web development. Mahadevappa et al. [66] highlight that such tools can automate the generation of content, design, and web code, significantly reducing the time and expertise required for development. Similarly, Ho et al. [67] emphasize that generative AI can improve student satisfaction and technology acceptance in programming courses by enabling the creation of user interface materials through simple text-based prompts. Their study, focused on App Inventor environments, shows that GAI saves instructors time in material preparation and enhances the quality and efficiency of instruction, ultimately benefiting student motivation and learning outcomes. This capability underscores the potential of AI to support both professional and educational contexts, particularly for languages like PHP used in web programming. In the context of this study, these advancements suggest that tools like Microsoft Copilot could enhance learning efficiency, provided students are equipped to harness their automation features effectively. Additionally, Jayachandran [68] notes that generative AI has been integrated into competitive programming events for university students, lowering participation barriers and boosting interest in programming, further illustrating its educational potential. This aligns with the present study’s exploration of Microsoft Copilot, highlighting the importance of considering how prompt design and tool capabilities influence outcomes in academic settings.

Regarding the use of generative artificial intelligence in computer programming education, several studies have analyzed its effect on the training of future programmers. For example, Keuning et al. [69] investigated students' perceptions of AI tools in programming courses, finding that the acceptance and use of such tools vary depending on the structure of each course and students' prior familiarity with the technology. Similarly, Yilmaz and Karaoglan Yilmaz [70] reported that incorporating ChatGPT into programming instruction enhances self-efficacy and student motivation, suggesting that GenAI facilitates the learning process and promotes greater academic engagement. However, Wilson and Nishimoto [71] cautioned that using these tools may complicate the assessment of student effort and actual comprehension, prompting the development of new, more appropriate evaluative methods.

In parallel, Shanshan and Sen [72] investigated the usefulness of AI-generated content in program debugging. Their results suggest that advanced integration of AI into programming tools increases performance and computational thinking; however, not all levels of integration show statistically significant benefits, indicating the need for further studies to understand the scope of such integration.

When discussing automation and code improvement in software development, Sajja et al. [73] evaluated the impact of GenAI on code quality and maintenance, highlighting its potential to automate repetitive tasks and enhance the productivity of development teams. Yehia [74] also described generative AI as a transformative technology capable of generating novel content across multiple domains. However, Liu and Li [75] emphasized the challenges associated with collaborative programming between humans and AI, underscoring coordination issues and the importance of considering ethical aspects when integrating these tools into educational and professional environments.

On the other hand, Boguslawski et al. [76] examined how language models influence the motivation of programming students. According to their findings, these models can foster autonomy and competence but do not replace the social support necessary for robust and meaningful learning. This latter aspect is particularly relevant, suggesting that the motivation to program with AI may involve additional dimensions beyond mere technological availability.

The study by [77] explores the impact of generative AI on teaching programming in higher education, comparing its effectiveness with video-based learning. Through an experiment involving 40 computer engineering students, learning outcomes, intrinsic motivation, and perceptions of the learning environment were assessed. The results indicate no significant differences in learning outcomes between the methods; however, generative AI improved perceptions of autonomy and reduced effort and pressure, while videos increased perceptions of competence. These findings suggest that both methods motivate students differently and complement each other to enhance programming instruction in university settings.

Despite the proliferation of research on GenAI in the programming field, contradictions and limitations that require attention persist. For example, Groothuijsen et al. [78] found that using AI chatbots in engineering education negatively influenced pair programming and collaboration among students, contrasting with studies reporting student engagement improvements [79]. Similarly, Frankford et al [80]. noted that while AI tutors in automated assessment systems provide timely feedback, they may also hinder autonomous learning by offering generic responses.

Regarding methodological limitations, Li et al. [81] indicated that their research on adaptive learning and GenAI lacks robust empirical evidence, limiting its applicability in real teaching environments. Similarly, Maphoto et al. [82] examined the incorporation of GenAI in distance education. Still, their conclusions are confined to a specific context, making it difficult to generalize their findings to other settings.

Although various studies analyze the acceptance and application of AI in programming, the motivation for programming with AI remains a relatively unexplored topic. While research such as that by Boguslawski et al. [76] and Yilmaz & Karaoglan Yilmaz [70] has approached motivation in programming education, there is still a need to investigate the factors that drive programmers to adopt AI as a development tool. Understanding these motivational drivers is essential for designing more intuitive and effective AI systems. Additionally, the relationship between self-efficacy and reliance on AI in programming requires more detailed analysis to determine when AI acts as a learning enhancer and when it may generate dependency or limit skill development.

Overall, recent literature agrees on the positive impact of generative AI on programming education and software development, although discrepancies persist in collaborative aspects and the assessment of learning outcomes. The main limitations stem from some studies' lack of generalization and robust empirical evidence. Finally, motivation for programming with AI emerges as a relevant gap in research, exploring which would contribute to the design of strategies and tools that promote more effective and responsible adoption of GenAI in programming. Ko et al. [83] emphasized that while GenAI tools can enhance learning experiences, they raise concerns regarding bias, dependency, and ethical dilemmas. The proposed framework encourages responsible use by guiding stakeholders to ensure GenAI contributes positively to student outcomes while addressing environmental and moral challenges in educational contexts.

3. Experimental Design

For the present study, a quasi-experimental design was chosen, following a quantitative research methodology based on the approach used by Mellado et al. [56]. A pretest-intervention-posttest scheme was implemented, where learning outcomes and affective variables were measured during the pre- and post-test stages, per the HMSAM model. Microsoft Copilot (Microsoft Copilot) was selected as the generative AI tool due to its free availability to students through their university-provided Office 365 accounts, ensuring accessibility, and its robust capabilities for real-time code generation and feedback, which align with the objectives of teaching PHP programming. While other AI tools, such as ChatGPT or Google Gemini (also accessible via university accounts), could offer user-friendly interfaces or education-specific features, Microsoft Copilot was preferred due to students’ prior familiarity with it from previous course activities, making it a practical choice for this context.

3.1 Participants

This study involved third-year students from an industrial engineering program at a university in Chile as participants. 71

students (53 men and 18 women), aged between 19 and 21, participated in the intervention. The activity was conducted during the first semester of 2024. Participants were randomly assigned to two groups: 35 students (23 men and 12 women) used Microsoft Copilot, and 36 (27 men and 9 women) utilized instructional videos as their study medium. The randomization was performed using the random group assignment feature in the Moodle platform, which automatically allocates participants into groups randomly.

To ensure the integrity of the random assignment, the Moodle group assignment function was configured to conceal group composition from the participants, thereby guaranteeing that students could not ascertain the assignment of their peers. Each student had access solely to the information about their group (Microsoft Copilot or instructional videos) via the platform, with no possibility of viewing the resources allocated to others. The assignment was executed automatically through the Moodle randomization feature before the commencement of the intervention, without manual intervention, which minimized the risk of bias in the allocation process. However, owing to the study design and logistical constraints, stratification strategies were not implemented, nor was the balance in additional baseline variables, such as gender, prior programming experience beyond the course modules, or digital literacy, verified. The pre-test focused exclusively on assessing knowledge of PHP, as this constituted the focus of the learning module, and all students possessed a similar exposure to the preceding Java and database modules, thereby minimizing initial variability within the context of this study. This process was completed before the intervention, ensuring each student had an equal probability of being assigned to either group. It was managed through Moodle, which facilitates access to the respective resources. Participation was voluntary; thus, this description excludes students who began but did not complete the activity.

We relied on the pre-test to assess initial PHP knowledge to control for potential pre-existing differences, as detailed in Section 4.1. An ANOVA applied to the pre-test scores confirmed no significant differences between the groups ($F(1,69) = 0.451$, p

= 0.50), indicating that the randomization effectively balanced prior knowledge. Additional variables such as age, gender, or previous academic performance were not analyzed in this study due to its scope and resource limitations. However, the random assignment via Moodle, combined with the pre-test equivalence, supports the comparability of the groups at baseline for this quasi-experimental design.

3.2 Curriculum

The activity was conducted as part of a software systems course, which includes a module on data structures in Java (8 weeks), another on databases (4 weeks), and a final module on web design with HTML and PHP (4 weeks). While the prior modules provided a foundation in programming logic and database management, the PHP module introduced a new language and web-specific concepts, representing a shift that required students to adapt their existing knowledge. The course consists of 3 weekly lecture sessions and 2 weekly laboratory workshop sessions. The learning objectives (LOs) considered in the activity correspond to the web module and were as follows:
LO1: The student analyzes basic PHP elements such as blocks, variables, loops, and decision-making in algorithmic solutions to simple problems.
LO2: The student correctly handles arrays, superglobal variables, and their concatenation with strings in algorithmic solutions.
LO3: The student correctly uses the `mysqli_` functions to interact with databases.

3.3 Process

Figure 1 illustrates the overall process used, which consists of four stages: (1) explanation of objectives and modality, (2) diagnostic assessment, (3) intervention with the exercise, and (4) final evaluation. In the first stage, during the lecture session, the objectives, modality, and deadlines for the exercise activity were explained. Later in the same week, during the two workshop sessions, stage 2 was carried out, which involved the application of diagnostic instruments to establish a baseline for comparison, both for initial knowledge and for affective perceptions about the upcoming activity (under the HMSAM model). During the same workshop sessions, stage 3 began, which consisted of the actual exercise, dividing participants into two groups: Microsoft Copilot and an instructional video. Figure 3 provides an example of an exercise from the PHP practice guide used specifically during stage 3 by the Microsoft Copilot group, highlighting the integration of follow-up questions to guide interaction with the AI tool.

The group division and activity sequence were managed through the Moodle platform assigned to the course. Participants were given 7 days to complete the exercise guide and the final questionnaires (stage 4), which included a final knowledge test and a perception questionnaire (HMSAM).

Although no explicit or formal training was provided immediately before the intervention, students assigned to the Microsoft Copilot group were assumed to be familiar with generative AI tools, including Microsoft Copilot specifically. This familiarity stemmed from their prior experiences within the course, where they had previously engaged informally with generative AI tools such as Microsoft Copilot, ChatGPT, and Google Gemini via their institutional Office 365 accounts. Consequently, only brief general instructions were given, allowing participants to interact freely with Microsoft Copilot during the exercises without additional detailed guidance or standardization protocols.

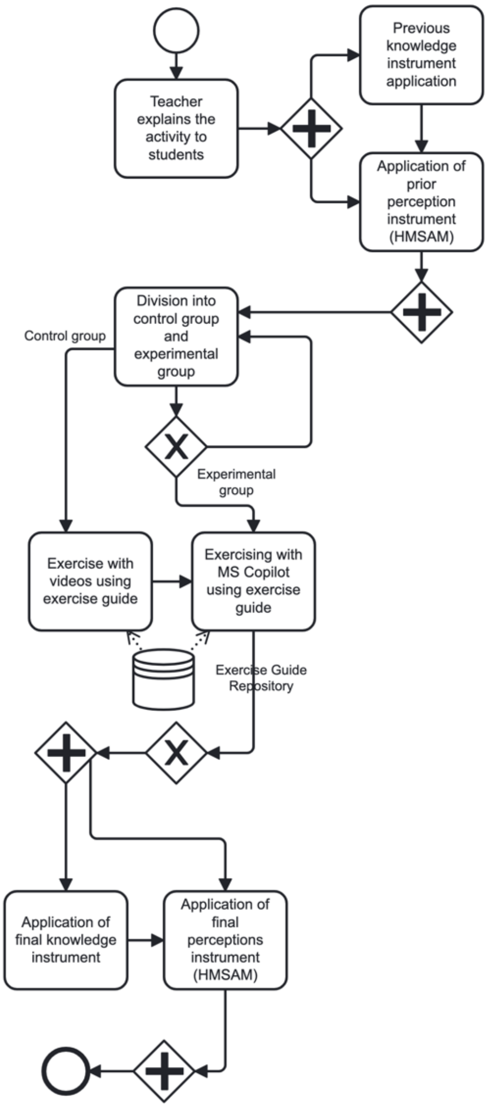
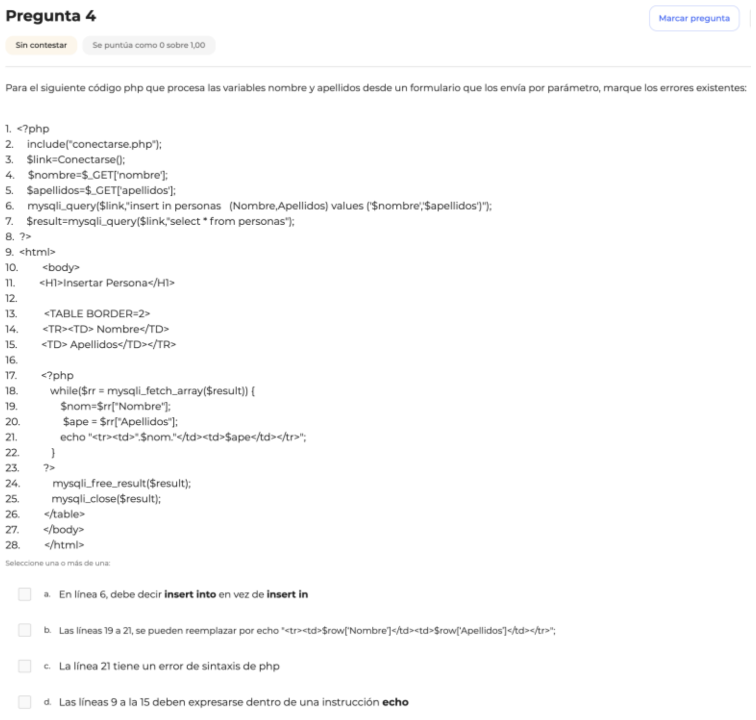

Teacher explains the activity to students
Previous knowledge instrument application
Application of prior perception instrument (HMSAM)
Division into control group and experimental group
Control group
Experimental group
Exercise with videos using exercise guide
Exercising with MS Copilot using exercise guide
Exercise Guide Repository
Application of final knowledge instrument
Application of final perceptions instrument (HMSAM)

Figure 1: Experimental process used.

Specific diagnostic instruments (pre- and post-tests) were used to evaluate learning, such as the example presented in Figure 2, which was drawn from a repository of questions and selected randomly. The example in Figure 2, extracted from the repository of questions used in the pretest and posttest, focuses on identifying and correcting errors in a PHP code that interacts with a MySQL database. The objective of the code is to process name and surname variables sent through a form, insert them into a table called "personas," and then display the results in an HTML table. However, the code contains several errors that must be corrected to function properly. These errors include incorrect use of SQL syntax in the INSERT statement, lack of data validation, and potential security issues such as SQL injection. The PHP instructions must also be reviewed to ensure the data is handled and displayed correctly in the browser. This exercise allowed for evaluating both the students' technical knowledge and their ability to apply programming concepts in practical contexts.


Pregunta 4

Para el siguiente código php que procesa las variables nombre y apellidos desde un formulario que los envía por parámetro, marque los errores existentes:

```
1. <?php
2.    include("conectarse.php");
3.    $link=Conectarse();
4.    $nombre=$_GET['nombre'];
5.    $apellidos=$_GET['apellidos'];
6.    mysqli_query($link,"insert in personas   (Nombre,Apellidos) values ('$nombre','$apellidos')");
7.    $result=mysqli_query($link,"select * from personas");
8. ?>
9. <html>
10.       <body>
11.      <H1>Insertar Persona</H1>
12.
13.       <TABLE BORDER=2>
14.       <TR><TD> Nombre</TD>
15.       <TD> Apellidos</TD></TR>
16.
17.       <?php
18.          while($rr = mysqli_fetch_array($result)) {
19.             $nom=$rr["Nombre"];
20.              $ape = $rr["Apellidos"];
21.             echo "<tr><td>".$nom."</td><td>$ape</td></tr>";
22.          }
23.       ?>
24.          mysqli_free_result($result);
25.          mysqli_close($result);
26.       </table>
27.       </body>
28.       </html>
```

Seleccione una o más de una:

- ☐ a. En línea 6, debe decir **insert into** en vez de **insert in**
- ☐ b. Las líneas 19 a 21, se pueden reemplazar por echo "<tr><td>$row['Nombre']</td><td>$row['Apellidos']</td></tr>";
- ☐ c. La línea 21 tiene un error de sintaxis de php
- ☐ d. Las líneas 9 a la 15 deben expresarse dentro de una instrucción **echo**

Figure 2: Example of the type of question used in pre and post-test.

Both groups used the same PHP practice guide, including a series of exercises with code snippets. Each exercise presents 3 to 4 non-exclusive alternatives containing statements about possible outcomes or consequences of the given code, specific lines of code, or potential replacements for certain lines of code. It is important to emphasize that this exercise guide is a compilation of various exercises used in previous semesters and captures the most frequent errors made by students who, despite having learned the material, begin coding in PHP.

Figure 3 presents an exercise from the PHP practice guide used in stage 3 by the Microsoft Copilot group, where a PHP block surrounds HTML code. Specifically, line 5 with $objetos = 2; is outside the PHP block, meaning it will not be preprocessed and will be displayed as plain text. The figure has been enlarged to enhance the legibility of the code and accompanying text.

To the Copilot software, in "precise" mode, add the following code prompts and questions:

```
<html>
<head>
  <title>Cálculos generales</title>
</head>
<body>
  $objetos = 2;
  <?php
    $peso = 10;
    echo "Peso total";
    echo $objetos * $peso;
    echo "<br>";
  ?>
</body>
</html>
```

Analysis Question:
1.1 The AI responded by assuming that the variable $objects is inside the PHP block. Is this correct? If not, correct the answer.

Alternatives proposed to the student:
a. The user will see, among other texts, the value 20.
b. The user will see, among other texts, the value 10 and the text <br>.

c. The user will see, among other texts, the string $objects=2;
d. The line containing $objects=2; will generate an error and will not be displayed.

Figure 3: Exercise 1 of the guide used.

Likewise, Figure 4 presents a PHP code snippet that utilizes the $POST superglobal variable to construct a database query in the $sql variable by concatenating strings with variables. Additionally, it uses Mysqli_ functions.

For the following PHP code, return the correct alternatives with their justification:

```
<?php
$nombre = $_POST["nombre"];
$apellido = $_POST["apellido"];
$sql = "INSERT INTO personas (nombre, apellido) VALUES
('$nombre', '$apellido')";

$conexion = mysqli_connect("localhost", "usuario",
"contraseña", "basededatos");
$resultado = mysqli_query($conexion, $sql);

if ($resultado) {
  echo "Registro insertado correctamente.";
} else {
  echo "Error al insertar: " . mysqli_error($conexion);
}

mysqli_close($conexion);
?>
```

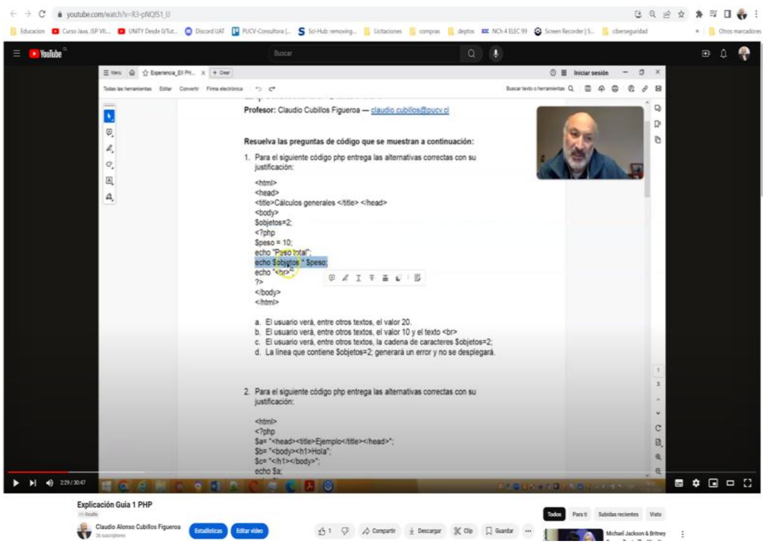
Analysis Question:
The AI generated a statement that directly concatenates the values of the variables $firstName and $lastName within the SQL statement.

Is this procedure correct? What observations would you make about this snippet?
Proposed alternatives (according to the original guide):

a. The code is well-written and should not generate errors.
b. Using double quotes with variables within the INSERT statement can cause interpretation errors.
c. The $sql variable should be declared within a function to avoid conflicts.
d. Using $_POST can allow the user to manipulate the query if there is no prior validation.

Figure 4: Exercise 5 of the guide used.

The control group used a video (see Figure 5) specifically created to review each exercise in the guide. The video explained why the different alternatives presented were either correct or incorrect.


Profesor: Claudio Cubillos Figueroa — claudio.cubillos@pucv.cl

Resuelva las preguntas de código que se muestran a continuación:

1. Para el siguiente código php entrega las alternativas correctas con su justificación:

```
<html>
<head>
<title>Cálculos generales </title> </head>
<body>
$objetos=2;
<?php
$peso = 10;
echo "Peso total";
echo $objetos * $peso;
echo "<br>";
?>
</body>
</html>
```

a. El usuario verá, entre otros textos, el valor 20.
b. El usuario verá, entre otros textos, el valor 10 y el texto
c. El usuario verá, entre otros textos, la cadena de caracteres $objetos=2;
d. La línea que contiene $objetos=2; generará un error y no se desplegará.

2. Para el siguiente código php entrega las alternativas correctas con su justificación:

```
<html>
<?php
$a= "<head><title>Ejemplo</title></head>";
$b= "<body><h1>Hola";
$c= "</h1></body>";
echo $a;
```

Figure 5: Explanatory video of the exercises is in the guide.

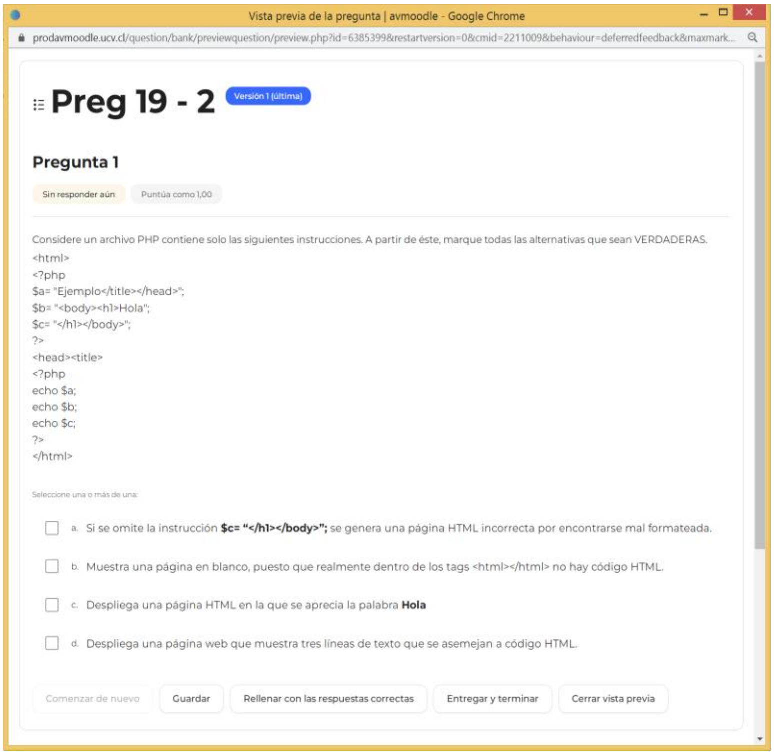
The experimental group used the same exercises but with the instruction to query Microsoft Copilot. Additionally, the exercises for this group included follow-up questions to guide students in evaluating the correctness of the AI's responses. For instance, in Exercise 1 (see Figure 3), a follow-up question 1.1 asks whether the AI considered the variable $objetos outside a PHP block. If not, the students were instructed to correct the AI.

To scaffold students' critical engagement with Microsoft Copilot, the exercise guide used in this group included follow-up questions accompanying specific tasks (see Figure 3). Given the context of each exercise, these questions prompted students to evaluate whether the AI's suggestions were syntactically and logically appropriate. For instance, one prompt asked: 'Did the assistant consider that the variable $objetos is outside the PHP block?' Students were encouraged to reformulate their queries or adjust the proposed code when discrepancies were detected. These prompts aimed to foster metacognitive awareness, reduce overreliance on AI suggestions, and support the iterative refinement of solutions.

3.4 Instruments

For the initial and final knowledge tests, exercises similar to those in the guide were used, following the format of code snippets with four non-exclusive alternatives. The pre-test and post-test included six exercises (two associated with each Learning Objective, LO), randomly selected from a pool of 18 exercises (distinct from the guide), with scores ranging from 0.0 to 10.0. These exercises were developed from materials used in prior semesters of the software systems course and aligned with the PHP module's learning objectives. Content validity was ensured through review by two instructors with over five years of PHP teaching experience. The pre- and post-test exercises were extracted from a broader item bank applied in the Software Systems course for over four academic years (seven semesters), primarily as practice and assessment tools aligned with the course's PHP module learning objectives. These items were developed by instructors with extensive experience in programming instruction and have undergone iterative refinement based on student performance and instructional feedback. While detailed item analysis was not conducted, the sustained application of these items across cohorts supports their empirical reliability and content validity. Post-hoc analysis of the post-test scores showed a Cronbach's alpha of 0.82, indicating good reliability. Figure 6 shows an example of a question used in the pre-test.

It is important to note that both experimental conditions—Microsoft Copilot and instructional video—worked with the same practice guide and were assessed using pre- and post-tests drawn from a shared item bank. This uniformity ensures that familiarity with item formats or content is applied equally across both groups, minimizing the risk of biased learning gains due to prior exposure.


Vista previa de la pregunta | avmoodle - Google Chrome
prodavmoodle.ucv.cl/question/bank/previewquestion/preview.php?id=6385399&restartversion=0&cmid=2211009&behaviour=deferredfeedback&maxmark...
Versión 1 (última)
Sin responder aún
Puntúa como 1,00
Comenzar de nuevo
Guardar
Rellenar con las respuestas correctas
Entregar y terminar
Cerrar vista previa

Preg 19 - 2

Pregunta 1

Considere un archivo PHP contiene solo las siguientes instrucciones. A partir de éste, marque todas las alternativas que sean VERDADERAS.

```
<html>
<?php
$a= "Ejemplo</title></head>";
$b= "<body><h1>Hola";
$c= "</h1></body>";
?>
<head><title>
<?php
echo $a;
echo $b;
echo $c;
?>
</html>
```

Seleccione una o más de una:

- ☐ a. Si se omite la instrucción **$c= "</h1></body>";** se genera una página HTML incorrecta por encontrarse mal formateada.
- ☐ b. Muestra una página en blanco, puesto que realmente dentro de los tags <html></html> no hay código HTML.
- ☐ c. Despliega una página HTML en la que se aprecia la palabra **Hola**
- ☐ d. Despliega una página web que muestra tres líneas de texto que se asemejan a código HTML.

Figure 6: Sample question in knowledge pretest.

On the other hand, for the initial and final perception tests, the HMSAM model was used with its eight dimensions: usefulness, enjoyment, ease of use, intention to use, control, focused immersion, temporal dissociation, and curiosity [16]. The total number of statements for both tests was 24 items, measured on a Likert scale from 1 to 7, where 1 corresponded to "Strongly Disagree," 4 to "Neutral," and 7 to "Strongly Agree".

Table 1: List with questions (statements) by construct of the HMSAM model (for pretest).

Construct	Question
Control (Ctrl)	• I will have little control over what I can do (Rev). • I expect to have control while performing the activities. • I expect to be able to freely choose what I want to see or do while performing the activities.
Curiosity (Cur)	• This experience will stimulate my curiosity. • This experience will spark my imagination. • This experience will make me curious.
Temporal Dissociation (TD)	• Time will seem to pass very quickly while doing the activity. • I will lose track of time while doing the activities. • Time will "fly" when I do the exercises.
Ease of use	• I believe navigating, writing questions, and reading answers will be easy. • I find that the activity with the ICT resource will be easy to use. • I believe that interacting with the ICT resource during the activity will be clear and understandable.
Focalized Immersion (FI)	• I will be focused and able to block out most distractions. • I will be absorbed/engaged in what I will be doing. • I will be immersed in the activity.
Enjoyment	• I will enjoy performing the activity. • I think it will be a fun activity. • The experience of the activity will be pleasant.
Behavioral Intention of Use (BIU)	• I believe I would plan to use it in the future to review. • I expect to continue using it in the future. • I believe I will intend to keep using this ICT resource during the semester.
Utility	• I expect that this activity will improve my knowledge of PHP. • I expect that this activity will help me with PHP programming. • I find that performing the activity will be useful.

4. Results

The statistical software SPSS 29 was used to analyze the results. Analysis of variance (ANOVA) and analysis of covariance (ANCOVA) tests were considered to measure the differences in initial and final learning outcomes. To measure the size of the effects found, partial eta squared (η^2) was used, with values of 0.01 indicating a small effect, 0.06 a medium effect, and 0.14 or greater a large effect [84].

4.1 Learning Effects

Table 1 presents the descriptive statistics for the knowledge tests administered at the beginning and end of the intervention, separated by experimental condition (video vs. Microsoft Copilot).

For assessing normality on pretest scores a Shapiro-Wilk test was performed for the Copilot ($W = 0.97$, $p = 0.36$) and the Video ($W = 0.98$, $p = 0.54$) groups, showing non significant differences from normality. On postest scores, Shapiro-Wilk test provided ($W = 0.95$, $p = 0.13$) for the Copilot and ($W = 0.95$, $p = 0.11$) for the Video conditions, indicative for normality. A Levene's test showed homogeneity of variances between conditions for pretest ($F = 0.38$, $p = 0.54$)and postest ($F = 3.18$, $p = 0.08$) scores.

Table 2: Descriptive statistics of pretest and posttest per condition.

Group	N	Pretest		Postest	
		Mean	**Standard deviation**	**Mean**	**Standard deviation**
Microsoft Copilot	35	4.46	2.42	5.69	2.32
Video	36	4.85	2.52	8.17	1.23
Total	71	4.66	2.46	6.94	2.22

When an ANCOVA test was applied to the post-test scores, using the pre-test as a covariate, to measure possible differences between groups, significant differences were found between the two groups. The group that used videos showed a higher post-test score than the group that used Microsoft Copilot, with $F(1,68) = 32.621$, $p < 0.001$, $\eta^2 = 0.32$ (see Figure 7).

Additionally, an analysis of variance (ANOVA) test was applied to the pre-test scores by the group to verify whether there were differences between the groups, yielding $F(1,69) = 0.451$, $p =$

0.50, and no significant differences were detected in the prior knowledge levels of the subjects in both groups. Similarly, an ANOVA was applied to the post-test scores by group, resulting in $F(1,69) = 31.920$, $p < 0.001$, $\eta^2 = 0.32$, indicating that the video group had a significantly higher post-test score than the Microsoft Copilot group.

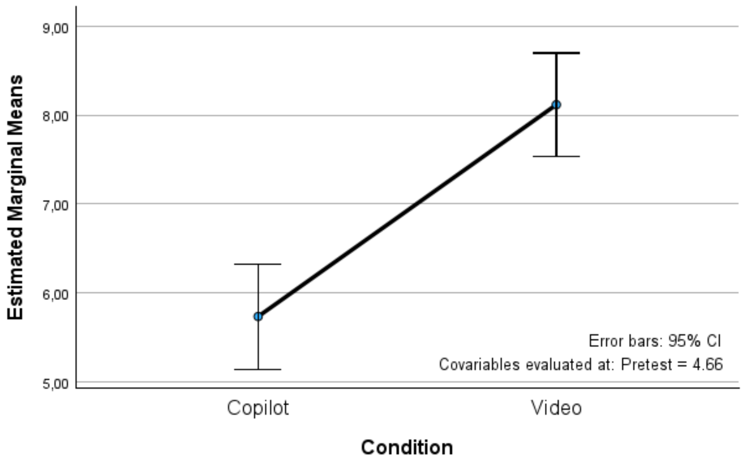


Figure 7: Estimated marginal means for posttest among conditions (video vs MSCopilot).

4.2 HMSAM Effects

Table 3 details the descriptive statistics for the pre-and post-intervention measures of the eight dimensions of the HMSAM model, broken down by condition. The constructs considered were usefulness, enjoyment, ease of use, intention to use, control, focused immersion, temporal dissociation, and curiosity. A Cronbach's alpha of 0.94 for the pre-IMI and 0.93 for the post-IMI perception tests was obtained.

On post perception values, homogeneity of variances between groups was assessed by the Levene's test for each of the eight constructs: Utility ($F = 2.69$, $p = 0.11$), Enjoyment ($F = 2.84$, $p = 0.70$), Ease of Use ($F = 0.71$, $p = 0.40$), BIU ($F = 1.60$, $p = 0.21$), Control (F = 0.96, p = 0.33), FI (F = 0.82, p = 0.37), TD (F = 2.25, p = 0.14), and Curiosity (F = 0.02, p = 0.90).

On pre-post perception values, the Levene's test showed equality of variances on Utility (F = 1.84, p = 0.18), Enjoyment (F = 0.24, p = 0.63), Ease of Use (F = 1.34, p = 0.25), BIU (F = 0.10, p = 0.75), Control (F = 0.74, p = 0.39), FI (F = 1.54, p = 0.22), TD (F = 3.57, p = 0.06), and Curiosity (F = 0.14, p = 0.71).

Table 3: Descriptive statistics of HMSAM pretest and posttest perceptions per condition.

Construct	**Condition**	**Pretest**		**Postest**	
		Mean	**Standard deviation**	**Mean**	**Standard deviation**
Utility	Copilot	5.78	0.99	5.48	1.14
	Video	5.72	0.94	5.92	0.87
	Total	5.75	0.96	5.70	1.03
Enjoyment (Joy)	Copilot	4.76	1.24	4.70	1.25
	Video	5.19	1.25	5.26	1.22
	Total	4.98	1.25	4.98	1.26
Ease of Use	Copilot	5.47	0.96	5.58	1.19
	Video	5.68	0.95	5.90	1.05
	Total	5.58	0.96	5.74	1.12
Behavioral Intention of Use (BIU)	Copilot	5.54	1.16	5.41	1.25
	Video	5.64	1.00	5.93	1.03
	Total	5.59	1.08	5.67	1.16
Control (Ctrl)	Copilot	5.24	0.93	5.11	1.05
	Video	5.46	0.89	5.43	1.16
	Total	5.35	0.91	5.27	1.11
Focalized Immersion (FI)	Copilot	5.13	1.02	5.12	1.04
	Video	5.41	1.13	5.54	1.22
	Total	5.28	1.08	5.33	1.15
Temporal Dissociation (TD)	Copilot	4.41	1.27	4.60	1.39
	Video	5.01	1.32	5.00	1.52
	Total	4.72	1.32	4.80	1.46
Curiosity (Cur)	Copilot	4.83	1.33	4.69	1.34
	Video	5.17	1.33	5.35	1.26
	Total	5.00	1.34	5.03	1.33

A two-way ANOVA was conducted considering the post-intervention measures, with the condition (video / Microsoft Copilot) and the HMSAM dimensions as factors on the scores obtained. The result was $F(1,69) = 3.820$, $p < 0.055$, $\eta^2 = 0.05$, indicating a marginally significant interaction. After performing pairwise comparisons with Bonferroni adjustment, it was found that only the construct of curiosity showed a significant difference in favor of the video condition, with $F(1,69) = 4.630$, $p < 0.035$, $\eta^2 = 0.06$.

This was followed by the dimensions of usefulness ($F(1,69) = 3.301$, $p < 0.074$, $\eta^2 = 0.05$), enjoyment ($F(1,69) = 3.716$, $p < 0.058$, $\eta^2 = 0.05$), and intention to use ($F(1,69) = 3.536$, $p < 0.064$, $\eta^2 = 0.05$), all showing differences in favor of the video condition, though not reaching conventional levels of significance. The remaining dimensions also showed differences favoring the video condition but were not statistically significant, as shown in Figure 8.

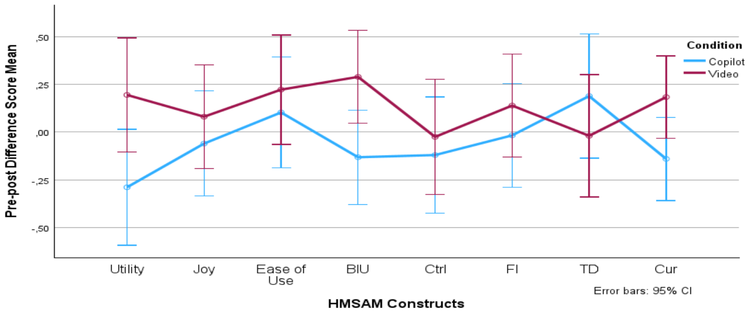


Figure 8: Post-perception scores of HMSAM constructs per condition.

An ANOVA was also conducted with the condition and HMSAM dimensions as factors, but this time on the score differences (post-test – pretest), yielding $F(1,69) = 2.912$, $p < 0.092$, $\eta^2 = 0.04$. Pairwise comparisons with Bonferroni adjustment revealed significant differences for the dimensions of usefulness ($F(1,69) = 5.135$, $p < 0.027$, $\eta^2 = 0.07$), intention to use ($F(1,69) = 5.867$, $p < 0.018$, $\eta^2 = 0.08$), and curiosity ($F(1,69) = 4.437$, $p < 0.039$, $\eta^2 = 0.06$), all favoring the video condition over the Microsoft Copilot condition (see Figure 9). It is important to note that, although eight separate ANOVA tests were conducted across the HMSAM constructs, no correction for family-wise error rate (FWER), such as Bonferroni or False Discovery Rate (FDR), was applied. This constitutes a limitation of the current analysis. However, the observed effects were consistent in direction and supported by medium effect sizes in key dimensions (e.g., $\eta^2 = 0.06$ for curiosity, $\eta^2 = 0.08$ for intention to use), suggesting that the findings are not solely due to random variation. Future studies should implement appropriate p-value correction techniques to ensure more conservative statistical inference and control for inflated Type I error risk when testing multiple constructs.

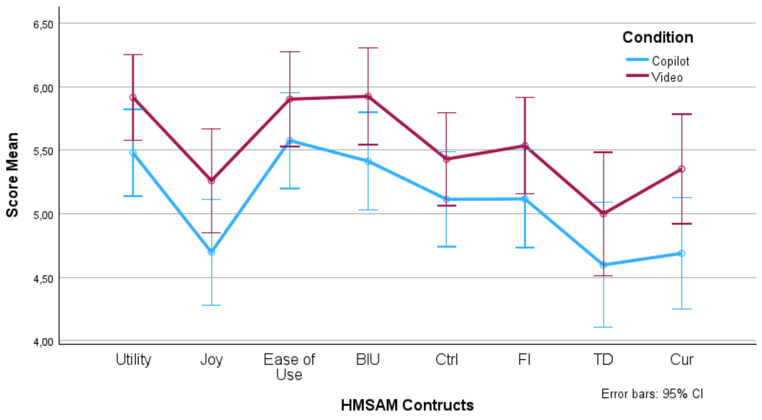


Figure 9: Pre-post perception differences of HMSAM constructs per condition.

5. Discussion

5.1 Learning Effects

The findings confirm that, although both methods promote improvements in student performance, those who followed the video-based instruction route achieved significantly greater progress. Several factors can explain this phenomenon.

From the cognitive load theory perspective [46,85], learning complex content (such as web programming logic) benefits from resources that optimize the distribution of mental load. In video-based instruction, students can process information visually and audibly simultaneously, facilitating the formation of integrated mental models [44,47]. This advantage is amplified by the fact that the instructional videos used in this study were specifically designed to teach PHP programming, with structured explanations and examples tailored to address common challenges in understanding complex concepts and algorithms. In contrast, while Microsoft Copilot provides immediate and adaptive feedback [86,87], it is a general-purpose GenAI tool not explicitly optimized for pedagogical purposes, requiring students to formulate questions and validate responses independently. Additionally, the ability to pause and rewind allows learners to control the pace of learning and focus on the most complex elements [28,48], further enhancing the effectiveness of these purposefully crafted audiovisual resources. This adaptive control over the learning pace may help reduce extraneous cognitive load, allowing students to understand better the code's logic and structures [46].

Although Microsoft Copilot provides immediate and adaptive feedback [86,87], students must formulate questions and validate the relevance of the generated responses. This process requires a higher level of metacognition and digital competencies to evaluate the quality of the feedback [38,42]. In contrast, videos present carefully sequenced examples and explanations, reducing uncertainty about appropriate practices. In this way, students perceive constant reinforcement that increases their perceived competence [39] and motivates them to continue exploring without fear of initial failure [33].

The inclusion of follow-up questions in the Copilot group (Figure 3) sought to support this evaluative process. These prompts encouraged learners to analyze the AI's output and identify potential inaccuracies critically. However, the impact of these questions likely varied depending on the student's level of engagement. Some participants may have used them effectively to guide iterative improvement, while others may have skipped them or answered superficially, leading to inconsistent benefits across the group.

In addition to these cognitive factors, it is important to consider that the quality and structure of the instructional content may have differed significantly between groups. The video provided a consistent, pedagogically sequenced explanation for each exercise, ensuring uniform content delivery across all participants. In contrast, the AI group relied on individually formulated prompts, resulting in variability in the responses' relevance, depth, and accuracy from Microsoft Copilot. This inconsistent content delivery may have contributed to the observed differences in learning outcomes. Future studies should control instructional structure across conditions to ensure more comparable and reliable evaluations of effectiveness.

While Levene’s test indicated homogeneity of variances between groups, the observed differences in post-test standard deviations (2.32 vs. 1.23) may suggest potential variance heterogeneity. However, given the substantial difference in post-test means, the coefficient of variation (CV) offers a more appropriate metric of relative dispersion, revealing that variability was not disproportionate when adjusted for group means. Moreover, ANCOVA is generally robust to moderate violations of the homogeneity of variances assumption, particularly in balanced designs [88,89]. Nonetheless, we acknowledge this as a potential limitation. As a future line of research, simulation studies [90] could be implemented to evaluate how deviations from normality or homoscedasticity might influence the robustness of ANCOVA results in educational contexts involving generative AI.

The Hedonic-Motivation System Adoption Model (HMSAM) establishes that the sense of enjoyment and perceived usefulness are key determinants in the adoption and effectiveness of learning systems [16]. On the other hand, audiovisual resources (with narrative, practical examples, and demonstrations) tend to generate greater affective and cognitive engagement by stimulating attention, curiosity, and interest [44,91]. In line with this, previous studies have shown that video-based learning can foster deeper emotional involvement, facilitating concept retention [29,31]. In contrast, interaction with generative artificial intelligence may be less engaging and require self-regulation strategies that not all students have developed, especially at the early stages of programming [30].

From an instructional perspective, Microsoft Copilot promotes a more active learning approach regarding exploration and constantly testing hypotheses within the code [82,92,93]. However, this constructivist approach may generate uncertainty when students lack a robust foundation in programming syntax and logic. On the other hand, instructional videos adopt an expository approach, where the teacher's guidance is explicit, and students assimilate problem-solving strategies more directly [28]. This difference becomes particularly noticeable in the early stages of learning when familiarity with basic concepts is essential to avoid cognitive overload [47,85].

Thus, despite the clear inclination of the results toward the effectiveness of videos, the findings do not diminish the value of generative artificial intelligence as a reinforcement tool or for advanced tutoring [80,81,94]. In particular, a mixed instructional strategy that combines the systematic and exemplified presentation of a video with the personalized feedback of Microsoft Copilot could maximize learning by providing a solid initial conceptual framework, followed by guided and immediate experimentation [29,33]. Future research could explore the optimal integration of both methods based on student profiles, such as their level of experience, learning styles, and intrinsic motivations.

The greater effectiveness of video in improving web programming learning can be explained by the reduced extraneous cognitive load, which facilitates sequential information reception through multiple modes; sustained attention and motivation derived from audiovisual resources; and the immediate and accessible support that reduces uncertainty for novice students. With this, we can answer the research question: Are there differences in learning between students who practice web programming using Microsoft Copilot versus instructional videos? Affirmatively, learning differences favor video-based instruction over generative artificial intelligence-based practice.
The profile of the participants (third-year industrial engineering students rather than computer science students) may further explain the observed learning effects. Unlike computer science students, who typically have extensive prior exposure to programming, industrial engineering students in this study had limited experience, primarily from prior course modules in Java and databases. This relative novelty of programming, particularly in PHP, likely heightened the importance of

affective factors such as curiosity, enjoyment, and immersion, as these students relied heavily on structured guidance (e.g., videos) to build confidence and competence. Consequently, the greater effectiveness of instructional videos may reflect their ability to meet the needs of learners with less programming expertise, suggesting that the comparative advantage of videos over Microsoft Copilot could vary with students who have stronger technical foundations.

An additional consideration that may limit the generalizability of our findings is the exclusive focus on PHP. While PHP was chosen for its curricular relevance and role in web development, different programming languages pose distinct syntactic, semantic, and conceptual challenges. For example, languages such as Python or JavaScript offer various levels of abstraction and readability, which could influence how learners interact with generative AI tools or benefit from structured instructional materials. Thus, the effectiveness observed in this study may not directly translate to other programming contexts. Future research should replicate this design using diverse programming languages to assess whether the comparative impact of instructional videos and AI tools remains consistent.

These conclusions open the door to the strategic use of both modalities, combining their strengths while mitigating their limitations. Thus, this integrated approach optimizes teaching and learning processes in web programming, databases, and Java. This integrated approach holds significant potential for enhancing educational outcomes and addressing the diverse needs of learners at different stages of their programming journey.

Nonetheless, it is important to underscore that the present findings are based solely on immediate post-intervention assessments. As such, they reflect short-term knowledge acquisition rather than long-term learning or retention. Without longitudinal follow-up data, we cannot determine whether the observed advantages of instructional videos persist over time. Nevertheless, it is important to note that the scope and design of this study were deliberately aligned with short-term, well-defined learning objectives. As established in instructional design theory and supported by Bloom’s taxonomy, limited interventions can be appropriate and effective when the targeted learning outcomes are specific, foundational, and skill-oriented [95-97]. Therefore, while long-term retention merits future study, the present design remains methodologically sound for its intended scope. Future studies should include delayed post-tests or follow-up assessments to examine the durability and transferability of these learning outcomes.

Although the pretest confirmed equivalence in baseline PHP knowledge between groups, it is important to note that this instrument did not directly assess broader aspects of digital literacy or prior experience with generative AI tools. While not measured, these factors could have contributed to the variability in how students interacted with Microsoft Copilot. For example, two students with similar programming knowledge may differ significantly in their ability to formulate prompts, interpret responses, or detect inaccuracies in AI outputs due to differences in their digital fluency. This represents a potential source of uncontrolled variance in the AI group. We recommend that future research incorporate explicit instruments to assess digital literacy and prior AI exposure, potentially including them as covariates or segmentation variables in experimental designs.

While the present study included a structured exercise guide with follow-up prompts designed to scaffold critical engagement with Microsoft Copilot, it did not incorporate a system for recording the number, content, or quality of student interactions with the tool. This limits the ability to assess the consistency or depth of engagement across participants, as actual usage behaviors (e.g., number of prompts submitted, adherence to correction instructions, or time on task) were not tracked. Although all participants in the AI condition received the same guided activities, designed to foster metacognitive reflection and verification of Copilot’s responses, there remains a gap between the intended instructional design and the unobserved execution of that design. Future studies should integrate interaction logging or usage analytics to more precisely evaluate the relationship between engagement patterns and learning outcomes.

Additionally, we did not record the number of queries or interactions each student had with Microsoft Copilot, which limits our ability to evaluate engagement consistency across participants. The absence of data on the quantity, quality, or nature of prompts (e.g., specificity, complexity, or frequency) precludes a detailed analysis of how student engagement with Microsoft Copilot influenced learning outcomes. Variability in prompt formulation and interaction patterns likely contributed to differences in the tool's effectiveness, as its performance heavily depends on the user's ability to craft effective prompts and critically assess AI-generated responses. Future studies should incorporate usage tracking or interaction logs to quantify engagement levels and explore their correlation with learning gains, thereby offering deeper insights into the role of AI tool proficiency in programming education.

Another critical factor potentially influencing the learning outcomes was the absence of explicit and standardized instructions for students interacting with Microsoft Copilot. While previous informal exposure to Microsoft Copilot and other generative artificial intelligence tools provided students with practical knowledge and competence in querying and interpreting AI outputs, individual variability in proficiency and approaches likely emerged. This variability might have influenced the consistency and effectiveness of interactions during the experimental activity. Future studies should explicitly standardize training sessions and develop structured interaction protocols for generative AI tools, thus ensuring greater methodological rigor, usage consistency, and improved results comparability across experimental conditions.

5.2 HMSAM Effects

5.2.1 Cognitive Absorption

The results of the present study focused on evaluating differences in the effects of cognitive absorption (enjoyment, control, focused immersion, temporal dissociation, and curiosity) between students who used Microsoft Copilot (Microsoft Copilot) and instructional videos to practice web programming, revealed that curiosity showed a significant difference in favor of the group that used videos over those who used generative artificial intelligence under the Microsoft Copilot model. This finding aligns with prior research highlighting the role of structured multimedia resources in stimulating intrinsic interest [35,37,47].

Most likely, curiosity, as the foundation of exploratory learning, was favored by instructional videos due to their ability to reduce extraneous cognitive load [85]. By integrating visual and auditory elements sequentially, videos facilitate the formation of coherent mental schemas, allowing students to focus on the logic of the code without informational overload [44]. This structured instructional design focuses attention and generates a sense of progressive competence, which is key to sparking curiosity [32]. For example, pausing and reviewing complex segments empowers students to explore concepts at their own pace, fostering self-directed curiosity [28].

In contrast, interaction with a generative artificial intelligence like Microsoft Copilot, while offering immediate feedback, may introduce beginner uncertainty by requiring the formulation of precise questions and the critical validation of responses generated by artificial models—processes that demand metacognitive skills still under development [38]. This dynamic could inhibit curiosity by being perceived as an obstacle to autonomous exploration.

Regarding the other constructs, such as enjoyment, control, focused immersion, and temporal dissociation, no significant differences were found between the groups. However, relevant trends were observed. In terms of enjoyment, videos scored slightly higher, suggesting that audiovisual storytelling might generate greater satisfaction by engaging emotional stimuli [91]. In terms of control, both methods showed similar levels, indicating that the flexibility of videos (pausing, rewinding) and the interactivity of Microsoft Copilot address different needs for autonomy. Although not statistically significant, focused immersion and temporal dissociation reflected that both approaches require comparable sustained attention, a critical aspect in programming environments [31].

These findings have significant pedagogical implications. Instructional videos emerge as effective tools for fostering curiosity in the early stages of learning, where clarity and structure are prioritized. However, Microsoft Copilot could be integrated into advanced stages, where students, already familiar with basic concepts, require personalized feedback for complex problems [42].

In response to the research question: Are there differences between students who practice web programming using Microsoft Copilot and those using instructional videos regarding cognitive absorption effects: enjoyment, control, focused immersion, temporal dissociation, and curiosity? We can state that only curiosity significantly differed in favor of the group that used videos. This result suggests that videos better foster intrinsic interest by structuring content clearly and through multisensory means. In contrast, no significant differences were found in the other constructs. This indicates that while structured multimedia resources enhance curiosity, different dimensions of cognitive absorption may depend on contextual factors or individual preferences.

Future research should explore hybrid models that combine both methodologies, adapting to student profiles and experience levels. It should also assess the long-term impact of curiosity on the retention of technical skills. Such investigations could provide deeper insights into optimizing learning strategies in web programming and related fields, ensuring that instructional approaches align with cognitive and motivational needs.

5.2.2 Dimensions of Technological Acceptance

The analysis of score differences (post-test – pre-test) revealed significant differences in perceived usefulness and intention to use, all favoring the group that used instructional videos compared to those that employed generative artificial intelligence. These findings suggest that videos are more effective in fostering curiosity, as previously discussed, and generate a higher perception of usefulness and a stronger intention to use the tool in the future.

First, perceived usefulness, defined as the extent to which students consider a tool to enhance their learning, was significantly higher in the video group. This aligns with the Technology Acceptance Model (TAM) [39], which posits that perceived usefulness is a key predictor of adopting educational tools. Videos provide structured and sequential explanations of programming concepts, allowing students to visualize how the content relates to their learning objectives and reinforcing their perception of usefulness [47].

On the other hand, while offering immediate feedback, Microsoft Copilot may generate uncertainty among students by requiring them to formulate precise questions and critically evaluate the generated responses. This process, which demands advanced metacognitive skills, could dilute their perception of usefulness, especially in the early stages of learning [38].
In the second place, intention to use, which reflects students' willingness to continue using a tool in the future, also showed significant differences in favor of videos. This result aligns with technology acceptance and usage studies that highlight the importance of ease of use and perceived usefulness in adopting technologies [28,41]. Being more intuitive and less demanding regarding metacognitive skills, videos may generate a more satisfying learning experience, increasing the intention to use them [28]. In contrast, interaction with Microsoft Copilot, while innovative and novel, may be perceived as more complex and less accessible for students at early programming levels, potentially reducing their intention to use it [42].

In response to research question RQ3: Are there differences in effects between students who practice using Microsoft Copilot and those using instructional videos regarding technology acceptance dimensions: ease of use, perceived usefulness, and intention to use? , there are significant differences in perceived usefulness and intention to use, both favoring the group that used videos. These results suggest that videos, by offering a more structured and accessible learning experience, generate a higher perception of usefulness and a stronger intention to use them in the future. In contrast, despite providing immediate feedback, Microsoft Copilot may not be perceived as equally useful or easy to use in early learning stages, limiting its adoption.

These findings underscore the importance of designing educational tools that balance innovation and accessibility, adapting to students' needs and experience levels. By addressing these factors, educators and developers can create solutions that maximize engagement and learning outcomes, ensuring that technological advancements are enablers rather than barriers to effective education.

6. Conclusions

This study evaluated the impact of using a generative artificial intelligence (Microsoft Copilot) compared to video-based instruction on university students' learning of web programming in PHP. Through a pretest-intervention-posttest experimental design, the effects on learning and affective perceptions of 71 participants were analyzed. Participants were randomly divided into two groups: one that used Microsoft Copilot and another that followed instructional videos. The Hedonic-Motivation System Adoption Model (HMSAM) was employed to assess perceptions.

The results revealed that students who received instruction through videos made greater progress in the post-test knowledge assessment than those who used Microsoft Copilot. This finding suggests that the sequential and multimodal structuring of information in videos, specifically designed to teach complex programming concepts and algorithms, facilitates the assimilation of concepts by reducing extraneous cognitive load. In contrast, Microsoft Copilot, as a general-purpose GenAI tool, lacks the tailored pedagogical focus of the videos, which may explain its relatively lower effectiveness for novice learners in this context.

These conclusions highlight the potential of instructional videos as a powerful tool for introductory programming education, particularly in contexts where clarity, structure, and reduced cognitive load are critical for student success, such as with industrial engineering students who may lack extensive programming experience. At the same time, they suggest that while generative AI tools like Microsoft Copilot offer innovative possibilities, their adoption may require careful scaffolding and integration, especially for novice learners. Notably, although the participants (third-year industrial engineering students) had prior programming experience from courses in Java and databases, their foundations might not have been sufficient to fully leverage Microsoft Copilot in this context. Learning PHP, a new language for them, alongside the autonomous use of Microsoft Copilot without specific training, likely demanded advanced metacognitive skills (e.g., crafting effective prompts and critically evaluating AI responses) that were not yet fully developed. This suggests that Microsoft Copilot could be more beneficial for students in advanced stages or with a computer science background, where a stronger, language-specific conceptual foundation and familiarity with AI interaction enable them to maximize their potential for real-time problem-solving. Future research should explore hybrid approaches that combine the strengths of both methods to optimize learning outcomes across different stages of programming education and learner profiles.

This study's main contribution provides empirical evidence on the differential impact of generative AI tools and traditional instructional methods on programming learning. These findings highlight the need to consider cognitive demands and students' affective perceptions when designing technology-based pedagogical strategies. Additionally, this study contributes relevant knowledge to the Latin American context, where research on the adoption of educational technologies is still scarce, and factors such as digital literacy and resource availability may influence the effectiveness of technological tools. While our results are rooted in the context of industrial engineering students learning PHP, they may hold broader implications. Due to their structured guidance and reduced cognitive load, the preference for instructional videos could generalize to other introductory programming courses (e.g., Python, JavaScript) or even non-programming domains (e.g., mathematics or engineering design) where novices benefit from clear, sequential instruction. Similarly, the potential of generative AI tools like Microsoft Copilot for advanced learners might extend to contexts requiring adaptive feedback. However, this would depend on learners' prior knowledge, the complexity of the subject matter, and the specific instructional design. However, such generalizations require caution, as differences in course objectives, disciplinary conventions, and student backgrounds could alter the observed effects.

Nevertheless, this research has some limitations. First, the sample was limited to students from a single university in Chile, which restricts the generalization of the findings to other educational and cultural contexts. Second, the intervention was a single-session activity conducted over seven days, with no follow-up assessments beyond the immediate post-test. This short duration means that the study primarily captures immediate learning outcomes rather than long-term retention or the ability to apply PHP skills in diverse contexts, potentially limiting the external validity of the results. The lack of follow-up assessments prevents us from determining whether the observed advantages of instructional videos or the potential of Microsoft Copilot persist over time. This is particularly relevant for programming education, where sustained practice is key to mastery. However, the study focused on comparing the immediate effectiveness of the two methods in a controlled setting, and the pre- and post-test design provides a valid measure of short-term learning gains within this scope. Finally, the study focused exclusively on learning PHP, so it would be relevant to examine whether the results are replicated in other programming languages and levels of complexity or entirely different study domains.

Another significant limitation is the lack of control over potential confounding variables, such as digital literacy, prior programming experience beyond the course modules, or gender, due to the absence of stratification or balance verification in the random assignment. Although the pre-test confirmed the equivalence in PHP knowledge between the groups, these unmeasured variables could have influenced the interaction with Microsoft Copilot, particularly considering that digital literacy may affect the ability to formulate prompts and validate AI responses. This limitation reflects the scope of the study, which prioritized evaluating the effects on learning and affective perceptions rather than technology adoption factors, which are often the primary focus in technology acceptance studies. Future research should include specific instruments to measure these variables and consider them as covariates or stratification criteria to enhance the robustness of the experimental design.

Future studies should address these limitations by expanding the sample to include diverse educational settings, extending the duration of interventions to evaluate long-term effects, and exploring the applicability of the findings across various programming languages (e.g., Python, C++) and non-programming disciplines (e.g., physics, statistics). Such efforts will help determine the extent to which the comparative advantages of instructional videos and generative AI tools can be generalized, providing a more comprehensive understanding of their role in technology-enhanced education across diverse contexts.

Future studies should address these limitations by expanding the sample to include diverse educational settings, extending the duration of interventions to evaluate long-term effects, and exploring the applicability of the findings across different programming languages and learner profiles. Such efforts will contribute to a more comprehensive understanding of how emerging technologies can be effectively integrated into programming education while addressing the unique needs of students in various contexts.

From a practical perspective, these findings provide valuable insights for curriculum designers and computer science educators. The results suggest that video-based instruction is an effective strategy for introductory programming teaching. At the same time, generative AI tools like Microsoft Copilot may be more useful in advanced stages when students already have solid conceptual foundations and require immediate feedback for problem-solving. At the technological level, these findings can guide the development of hybrid platforms that combine the pedagogical structure of videos with the adaptability of generative AI to optimize the learning experience.

As future lines of research, longitudinal studies to evaluate the long-term effects of these tools on learning are recommended. Such studies could include follow-up assessments at multiple intervals (e.g., one month, three months) to examine retention and skill application, addressing the limitation of the current short-term focus. Additionally, it would be valuable to explore combined strategies that integrate video-based instruction with generative AI, analyzing their effectiveness across different experience levels and learning profiles. Finally, expanding the sample to include diverse institutions and countries will help validate the generalization of these findings in varied educational contexts, strengthening the understanding of the impact of technology on education in Latin America. However, it is important to note that these conclusions are based on short-term post-intervention assessments. Future research should include delayed post-tests or follow-up studies to evaluate long-term retention and the sustainability of these instructional effects.

References

1. Bahroun Z, Anane C, Ahmed V, Zacca A. Transforming Education: A Comprehensive Review of Generative Artificial Intelligence in Educational Settings through Bibliometric and Content Analysis. Sustainability. 2023; 15: 12983.
2. Luckin R, Holmes W. Intelligence Unleashed: An argument for AI in Education. London: UCL Knowledge Lab. 2016; 849–851. Available onine at: https://www.pearson.com/content/dam/corporate/global/pearson-dot-com/files/innovation/Intelligence-Unleashed-Publication.pdf
3. Jackson V, Vasilescu B, Russo D, Ralph P, Izadi M, et al. The Impact of Generative AI on Creativity in Software Development: A Research Agenda. ACM Transactions on Software Engineering and Methodology. 2024.
4. Agarwal R, Karahanna E. Time flies when you're having fun: Cognitive absorption and beliefs about information

technology usage. MIS Quarterly: Management Information Systems. 2000; 24: 665–694.
5. Saadé R, Bahli B. The impact of cognitive absorption on perceived usefulness and perceived ease of use in on-line learning: an extension of the technology acceptance model. Information & Management. 2005; 42: 317–327.
6. Ling HC, Chiang H Sen. Learning Performance in Adaptive Learning Systems: A Case Study of Web Programming Learning Recommendations. Frontiers in Psychology. 2022; 13: 770637.
7. Liu IF, Hung HC, Liang CT. A study of programming learning perceptions and effectiveness under a blended learning model with live streaming: comparisons between full-time and working students. Interactive Learning Environments. 2024.
8. Park TH, Wiedenbeck S. Learning web development: Challenges at an earlier stage of computing education. ICER'11 - Proceedings of the ACM SIGCSE 2011 International Computing Education Research Workshop. 2011; 125–132.
9. Robins A, Rountree J, Rountree N. Learning and Teaching Programming: A Review and Discussion. Computer Science Education. 2023; 21: 137–172.
10. Guo PJ, Kim J, Rubin R. How video production affects student engagement: An empirical study of MOOC videos. L@S 2014 - Proceedings of the 1st ACM Conference on Learning at Scale. 2014; 41–50.
11. De La Torre A, Baldeon-Calisto M. Generative Artificial Intelligence in Latin American Higher Education: A Systematic Literature Review. 12th International Symposium on Digital Forensics and Security, ISDFS. 2024.
12. Guerrero-Quiñonez AJ, Bedoya-Flores MC, Mosquera-Quiñonez EF, Mesías-Simisterra ÁE, Bautista-Sánchez JV. Artificial Intelligence and its scope in Latin American higher education. Ibero-American Journal of Education & Society Research. 2023; 13: 264–271.
13. Hinostroza JE, Isaacs S, Bougroum M. Information and Communications Technologies for Improving Learning Opportunities and Outcomes in Developing Countries.

Learning and Education in Developing Countries. 2014; 42–57.

14. LahtinenEssi, Ala-MutkaKirsti, JärvinenHannu-Matti. A study of the difficulties of novice programmers. ACM SIGCSE Bulletin. 2005; 37: 14–18.
15. Gosavi CS, Arora S. Active Learning Strategies for Engaging Students in Higher Education. Journal of Engineering Education Transformations. 2022; 36: 2394–1707.
16. Lowry PB, Gaskin JE, Twyman NW, Hammer B, Roberts TL, et al. Taking "Fun and Games" Seriously: Proposing the Hedonic-Motivation System Adoption Model (HMSAM). Journal of the Association for Information Systems. 2013; 14: 2.
17. Davis FD, Bagozzi RP, Warshaw PR. Extrinsic and Intrinsic Motivation to Use Computers in the Workplace1. Journal of Applied Social Psychology. 1992; 22: 1111–1132.
18. Venkatesh V, Thong JYL, Xu X. Consumer acceptance and use of information technology: Extending the unified theory of acceptance and use of technology. MIS Quarterly: Management Information Systems. 2012; 36: 157–178.
19. Jennett C, Cox AL, Cairns P, Dhoparee S, Epps A, et al. Measuring and defining the experience of immersion in games. International Journal of Human-Computer Studies. 2008; 66: 641–661.
20. Ma JY, Xie JF, Chen CC. Exploring the Structural Relationships of Microinteractions in Perception and Behavior by the Hedonic Motivation System Adoption Model. International Journal of Human–Computer Interaction. 2025.
21. Qomarul Huda M, Aeni Hidayah N, Nur Hafizah Hersyaf T, Sujoko I, Asmawi. Analysis of Continuance Use of Video on Demand Applications by Using the Hedonic Motivation System Adoption Model. 2020 8th International Conference on Cyber and IT Service Management, CITSM 2020. 2020.
22. Chakraborty S. Generative AI in Modern Education Society. Computers and Society. 2024.
23. Hamari J, Koivisto J. Why do people use gamification services? International Journal of Information Management. 2025; 35: 419–431.

24. Zhang X, Guo X, Lai KH, Guo F, Li C. Understanding gender differences in m-health adoption: a modified theory of reasoned action model. Telemedicine Journal and E-Health : The Official Journal of the American Telemedicine Association. 2014; 20: 39–46.
25. Bratianu C, Bejinaru R. Knowledge dynamics: a thermodynamics approach. Kybernetes. 2020; 49: 6–21.
26. Bratianu C, Garcia-Perez A. Knowledge Dynamics and Expert Knowledge Translation: A Case Study. European Conference on Knowledge Management. 2023; 24: 140–147.
27. Qadhi S, Qadhi S. Knowledge Dynamics: Educational Pathways from Theories to Tangible Outcomes. From Theory of Knowledge Management to Practice. 2023.
28. Bishop JL, Verleger MA. The flipped classroom: A survey of the research. ASEE Annual Conference and Exposition, Conference Proceedings. 2013.
29. Sung YT, Chang KE, Liu TC. The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. Computers & Education. 2016; 94: 252–275.
30. Bernard RM, Abrami PC, Borokhovski E, Wade CA, Tamim RM, et al. A Meta-Analysis of Three Types of Interaction Treatments in Distance Education. 2009; 79: 1243–1289.
31. Gordillo A, Lopez-Fernandez D, Tovar E. Comparing the Effectiveness of Video-Based Learning and Game-Based Learning Using Teacher-Authored Video Games for Online Software Engineering Education. IEEE Transactions on Education. 2022; 65: 524–532.
32. Deci EL, Ryan RM. Intrinsic Motivation and Self-Determination in Human Behavior. Intrinsic Motivation and Self-Determination in Human Behavior. 1985.
33. Jena AK, Bhattacharjee S, Gupta S, Das J, Debnath R. Exploring the Effects of Web 2.0 Technology on Individual and Collaborative Learning Performance in Relation to Self-Regulation of Learners. Journal on School Educational Technology. 2018; 13: 20–35.
34. Huang J, Mizumoto A. Examining the effect of generative AI on students' motivation and writing self-efficacy. Digital Applied Linguistics. 2024; 1: 102324–102324.

35. Litman JA. Curiosity and the pleasures of learning: Wanting and liking new information. Cognition and Emotion. 2005; 19: 793–814.
36. Krouska A, Mylonas P, Kabassi K, Caro J, Sgouropoulou C, et al. Higher Education Students' Task Motivation in the Generative Artificial Intelligence Context: The Case of ChatGPT. Information. 2024; 15: 33.
37. Dousay TA. Multimedia Design and Situational Interest: A Look at Juxtaposition and Measurement. 2014; 69–82.
38. Abdelshiheed M, Maniktala M, Barnes T, Chi M. Assessing Competency Using Metacognition and Motivation: The Role of Time-Awareness in Preparation for Future Learning. 2023.
39. Davis FD. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly: Management Information Systems. 1989; 13: 319–339.
40. Ghimire A, Edwards J. Generative AI Adoption in Classroom in Context of Technology Acceptance Model (TAM) and the Innovation Diffusion Theory (IDT). 2024.
41. Venkatesh V, Morris MG, Davis GB, Davis FD. User acceptance of information technology: Toward a unified view. MIS Quarterly: Management Information Systems. 2003; 27: 425–478.
42. Steinert S, Avila KE, Ruzika S, Kuhn J, Küchemann S. Harnessing Large Language Models to Enhance Self-Regulated Learning via Formative Feedback. 2023.
43. Lin Z, Ng YL. Unraveling Gratifications, Concerns, and Acceptance of Generative Artificial Intelligence. International Journal of Human–Computer Interaction. 2024.
44. Clark R Colvin, Mayer RE. E-learning and the science of instruction : proven guidelines for consumers and designers of multimedia learning. 2016.
45. Al-Abdullatif AM. Modeling Teachers' Acceptance of Generative Artificial Intelligence Use in Higher Education: The Role of AI Literacy, Intelligent TPACK, and Perceived Trust. Education Sciences. 2024; 14: 1209.
46. Paas FGWC, Van Merriënboer JJG. Instructional control of cognitive load in the training of complex cognitive tasks. Educational Psychology Review. 1994; 6: 351–371.

47. Mayer RE. Multimedia Learning. Multimedia Learning, Second Edition. 2009; 1–304.
48. Martins DS, Cunha BCR, Yaguinuma CA, Zaine I, Pimentel M da GC. Effects of interactive video annotations on students' browsing behavior and perceived workload. ACM SIGAPP Applied Computing Review. 2019; 19: 44–57.
49. Bennedsen J, Caspersen ME. Failure rates in introductory programming. ACM SIGCSE Bulletin. 2007; 39: 32–36.
50. Suzuki R, Kato J, Yatani K. ClassCode: An Interactive Teaching and Learning Environment for Programming Education in Classrooms. 2020.
51. Yang C, Thung F, Lo D. Efficient Search of Live-Coding Screencasts from Online Videos. Proceedings - 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER. 2022; 73–77.
52. Khandwala K, Guo PJ. Codemotion: Expanding the design space of learner interactions with computer programming tutorial videos. Proceedings of the 5th Annual ACM Conference on Learning at Scale, L at S 2018. 2018.
53. Li W, Pea R, Haber N, Subramonyam H. Tutorly: Turning Programming Videos Into Apprenticeship Learning Environments with LLMs. 2024.
54. Buffardi K, Wang R. Integrating Videos with Programming Practice. Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE. 2022; 1: 241–247.
55. McGowan A, Anderson N, Trombino G, Sage P, Adhikari J, et al. Learning to code – investigating the adoption of video-based lab solutions for university-level novice programmers. ICERI2023 Proceedings. 2023; 1136.
56. Mellado R, Cubillos C. Gamification improves learning: Experience in a training activity of computer programming in higher education. Journal of Computer Assisted Learning. 2024; 40: 1959–1973.
57. Ferreira DJ, da Silva HC, Melo TFN, Ambrósio AP. Investigation of Continuous Assessment of Correctness in Introductory Programming. Educational Technology & Society. 2017; 20: 182–194.
58. Kadar R, Mahlan SB, Shamsuddin M, Othman J, Wahab NA. Analysis of Factors Contributing to the Difficulties in Learning Computer Programming among Non-Computer

Science Students. 2022 12th IEEE Symposium on Computer Applications and Industrial Electronics, ISCAIE. 2022; 89–94.
59. Esche S, Weihe K. Choosing a Didactic Basis for an Instructional Video: What Are the Implications for Novice Programmers? Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE. 2023; 1: 450–456.
60. Nguyen-Duc A, Cabrero-Daniel B, Przybylek A, Arora C, Khanna D, et al. Generative Artificial Intelligence for Software Engineering -- A Research Agenda. 2023.
61. Bilgram V, Laarmann F. Accelerating Innovation With Generative AI: AI-Augmented Digital Prototyping and Innovation Methods. IEEE Engineering Management Review. 2023; 51: 18–25.
62. Calegario F, Burégio V, Erivaldo F, Moraes D, Andrade C, et al. Exploring the intersection of Generative AI and Software Development. 2023.
63. Ebert C, Louridas P. Generative AI for Software Practitioners. IEEE Software. 2023; 40: 30–38.
64. Ulfsnes R, Moe NB, Stray V, Skarpen M. Transforming Software Development with Generative AI: Empirical Insights on Collaboration and Workflow. Generative AI for Effective Software Development. 2024; 219–234.
65. Nettur SB, Karpurapu S, Nettur U, Gajja LS. Cypress Copilot: Development of an AI Assistant for Boosting Productivity and Transforming Web Application Testing. IEEE Access. 2024.
66. Mahadevappa P, Muzammal SM, Tayyab M, Mahadevappa P, Muzammal SM,. Introduction to Generative AI in Web Engineering: Concepts and Applications. 2025; 15: 297–330.
67. Ho CL, Liu XY, Qiu YW, Yang SY. Research on Innovative Applications and Impacts of Using Generative AI for User Interface Design in Programming Courses. ACM International Conference Proceeding Series. 2024; 68–72.
68. Jayachandran D, Maldikar P, Love TS, Blum JJ. Leveraging Generative Artificial Intelligence to Broaden Participation in Computer Science. Proceedings of the AAAI Symposium Series. 2024; 3: 486–492.

69. Keuning H, Alpizar-Chacon I, Lykourentzou I, Beehler L, Köppe C, et al. Students' Perceptions and Use of Generative AI Tools for Programming Across Different Computing Courses. 2024.
70. Yilmaz R, Karaoglan Yilmaz FG. 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. 2023; 4: 100147.
71. Wilson SE, Nishimoto M. Assessing Learning of Computer Programing Skills in the Age of Generative Artificial Intelligence. Journal of Biomechanical Engineering. 2024; 146.
72. Shanshan S, Sen G. Empowering learners with AI-generated content for programming learning and computational thinking: The lens of extended effective use theory. Journal of Computer Assisted Learning. 2024; 40: 1941–1958.
73. Sajja A, Thakur D, Mehra A, Sajja A, Thakur D, et al. Integrating Generative AI into the Software Development Lifecycle: Impacts on Code Quality and Maintenance. International Journal of Science and Research Archive. 2024; 13: 1952–1960.
74. Yehia E. Developments on Generative AI. AI and Emerging Technologies: Automated Decision-Making, Digital Forensics, and Ethical Considerations. 2024; 139–160.
75. Liu J, Li S. Toward Artificial Intelligence-Human Paired Programming: A Review of the Educational Applications and Research on Artificial Intelligence Code-Generation Tools. 2024.
76. Boguslawski S, Deer R, Dawson MG. Programming education and learner motivation in the age of generative AI: student and educator perspectives. Information and Learning Science, ahead-of-print(ahead-of-print). 2024.
77. Cubillos C, Mellado R, Cabrera-Paniagua D, Urra E. Generative Artificial Intelligence in Computer Programming: Does it enhance learning, motivation, and the learning environment? IEEE Access. 2025.
78. Groothuijsen S, van den Beemt A, Remmers JC, van Meeuwen LW. AI chatbots in programming education: Students' use in a scientific computing course and

consequences for learning. Computers and Education: Artificial Intelligence. 2024; 7: 100290.
79. Pesovski I, Santos R, Henriques R, Trajkovik V. Generative AI for Customizable Learning Experiences. Sustainability. 2024; 16: 3034.
80. Frankford E, Sauerwein C, Bassner P, Krusche S, Breu R. AI-Tutoring in Software Engineering Education Experiences with Large Language Models in Programming Assessments. Proceedings - International Conference on Software Engineering. 2024; 309–319.
81. Li H, Xu T, Zhang C, Chen E, Liang J, et al. Bringing Generative AI to Adaptive Learning in Education. 2024.
82. Maphoto KB, Sevnarayan K, Mohale NE, Suliman Z, Ntsopi TJ, et al. Advancing Students' Academic Excellence in Distance Education: Exploring the Potential of Generative AI Integration to Improve Academic Writing Skills. Open Praxis. 2024; 16: 142–159.
83. Ko S, Chan SCH, Ko S, Chan SCH. A Framework for the Responsible Integration of Generative AI Tools in Learning. 2024; 5: 163–194.
84. Lakens D. Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for t-tests and ANOVAs. Frontiers in Psychology. 2013; 4: 62627.
85. Sweller J. Cognitive load during problem solving: Effects on learning. Cognitive Science. 1988; 12: 257–285.
86. Du X, Liu M, Wang K, Wang H, Liu J, et al. Evaluating Large Language Models in Class-Level Code Generation. Proceedings - International Conference on Software Engineering. 2024; 982–994.
87. Vaithilingam P, Zhang T, Glassman EL. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. Conference on Human Factors in Computing Systems - Proceedings. 2022. Available online at: https://doi.org/10.1145/3491101.3519665/SUPPL_FILE/3491101.3519665-TALK-VIDEO.MP4
88. Ch KP, Poremba KD, Rowell RK. Testing for Homogeneity of Slopes in Analysis of Covariance: A Tutorial. 1997.
89. Rheinheimer DC, Penfield DA. The Effects of Type I Error Rate and Power of the ANCOVA F-Test and Selected

Alternatives under Non-Normality and Variance Heterogeneity. 1998.
90. Ateş C, Kaymaz Ö, Tekindal MA, Erdoğan BD. Robustness of analysis of covariance (ancova) under the distributions assumptions and variance homogeneity. Eurasian Journal of Veterinary Sciences. 2020; 36: 58–65.
91. Huang YM, Chiu PS. The effectiveness of a meaningful learning-based evaluation model for context-aware mobile learning. British Journal of Educational Technology. 2015; 46: 437–447.
92. Hou I, Mettille S, Li Z, Man O, Zastudil C, et al. The Effects of Generative AI on Computing Students' Help-Seeking Preferences. ACM International Conference Proceeding Series. 2024; 39–48.
93. Kharrufa A, Johnson I. The Potential and Implications of Generative AI on HCI Education. ACM International Conference Proceeding Series. 2024.
94. Chen E, Lee JE, Lin J, Koedinger K. GPTutor: Great Personalized Tutor with Large Language Models for Personalized Learning Content Generation. L@S 2024 - Proceedings of the 11th ACM Conference on Learning @ Scale. 2024; 539–541.
95. Bloom BS. Learning for Mastery. Instruction and Curriculum. Regional Education Laboratory for the Carolinas and Virginia, Topical Papers and Reprints, Number 1. Evaluation Comment. 1968; 1.
96. Conklin J. Review of A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives Complete Edition. Educational Horizons. 2005; 83: 154–159.
97. Guskey TR. Closing Achievement Gaps: Revisiting Benjamin S. Bloom's "Learning for Mastery." Journal of Advanced Academics. 2007; 19: 8–31.