UDC 378.147:004.8

Yurii Kleban
National University of Ostroh Academy, Ostroh, Ukraine
ORCID ID 0000-0002-7070-5175
yuriy.kleban@oa.edu.ua

HUMAN–GENAI INTERACTION MODES IN PROGRAMMING EDUCATION

Abstract. Generative AI (GenAI) tools are becoming part of everyday learning in higher education, especially in programming courses where students can ask for explanations, debugging help, or complete code. This creates an important educational question: not only whether students use AI, but how they interact with it. This study compares three Human–GenAI interaction modes in an introductory object-oriented programming course: No AI, AI Tutor, and Direct AI. Thirty students completed three C# programming tasks under a within-subject classroom design with a balanced rotation of task order. The study examined completion time, code quality, measured understanding, perceived difficulty, and students' subjective evaluations. The Direct AI task–mode combination was associated with the strongest product-oriented outcomes: shorter completion time (Mdn = 10 min), higher code scores (M = 9.20/10), and the highest completion rate (96.7%). AI Tutor and No AI showed longer completion times and lower code scores. However, measured understanding varied little across modes (M = 3.57–3.70/4), with all medians at the maximum score, indicating a ceiling effect. The raw-score product–understanding gap was largest under Direct AI (M = 5.50), reflecting a stronger increase in code scores than in measured understanding. Student perceptions added a different perspective: 70.0% selected AI Tutor as the most effective mode for learning, while 90.0% selected Direct AI as the fastest. The findings suggest that unrestricted AI support can optimise task completion and product quality, but these gains should not be read as equivalent to learning gains. Guided AI Tutor interaction may be educationally valuable because it is designed to preserve more of the student's reasoning process. Because each mode was paired with a specific task, the results should be interpreted as task–mode combination differences rather than isolated causal effects of interaction mode. The study contributes empirical evidence for designing GenAI-supported programming activities and assessment practices that distinguish product-oriented performance from learning-oriented outcomes.

Keywords: Generative AI; human–AI interaction; programming education; object-oriented programming; AI tutor; learning outcomes.

1. INTRODUCTION

Generative AI has changed the practical conditions under which students learn programming. A novice programmer can now ask a tool such as ChatGPT or GitHub Copilot to generate a complete class, explain a compiler error, rewrite a method, or produce an alternative implementation. This can reduce frustration and make programming tasks more accessible. It also creates a difficult assessment problem: a correct code submission no longer necessarily shows that the student understands the concept implemented in the code [1, p. 2523]. This issue is especially visible in programming education because code is both the learning product and the usual evidence of competence.

At the same time, Generative AI (GenAI) is not only a challenge for assessment but also a potentially powerful learning aid. It can personalise learning, provide immediate feedback, and make tutor-like support available to large groups of students [2]. In programming education, the main goals of using AI include explaining concepts and error messages, providing personalised feedback on code, scaffolding problem solving through hints and guiding questions, supporting debugging, generating examples and practice tasks, and helping teachers prepare learning materials and feedback [3]. Empirical evidence, however, shows that the educational value of AI depends on how its support is designed. In a large field experiment,

© Yurii Kleban, 2026.

unrestricted access to a GPT-based assistant improved students' performance during practice but harmed their subsequent performance once access was removed, whereas a tutor-like version with pedagogical safeguards largely mitigated this negative effect [4]. Conversely, a carefully designed AI tutor produced greater learning gains than in-class active learning [5]. Therefore, the learning value of AI is determined less by access to the tool itself than by the goal it serves in a learning task, that is, by the mode of Human–GenAI interaction.

Statement of the problem.

The need to clarify how students should interact with GenAI is also reflected in international regulatory and policy documents on the ethical and responsible use of AI in education. The UNESCO Recommendation on the Ethics of Artificial Intelligence establishes human oversight, transparency, and fairness as basic principles for AI use [6]. UNESCO's Guidance for Generative AI in Education and Research calls for a human-centred approach that protects learners' agency and warns against uncritical reliance on AI-generated content [7]. The UNESCO AI competency frameworks define the knowledge, skills, and values that students and teachers need to use AI ethically and responsibly [8], [9], while the European Commission's ethical guidelines help educators decide when and how AI and data can be used in teaching and learning [10]. The EU Artificial Intelligence Act classifies AI systems used to evaluate learning outcomes or steer the learning process as high-risk and requires AI literacy from those who deploy AI systems [11]. Together, these documents agree that AI should support rather than replace students' cognitive work. However, they formulate general principles and do not specify how students' interaction with GenAI should be organised in concrete learning tasks.

Much of the current discussion about GenAI in higher education still treats AI use as a binary condition: students either use AI or they do not. For teaching and assessment, this distinction is too coarse. Asking an AI tool to generate a complete solution, asking it for hints, and working without AI are different learning situations. They differ in how much cognitive work remains with the student, how much of the final product is generated externally, and how much opportunity the student must explain, test, and revise their own reasoning. The central issue is therefore not only whether students use GenAI, but what kind of interaction with GenAI is built into the learning task. Accordingly, the research problem addressed in this study is the lack of empirical evidence on how different modes of Human–GenAI interaction affect the balance between product quality and learning in programming education. Without such evidence, teachers have limited grounds for translating the international principles of responsible AI use into concrete decisions about which kind of AI support to allow in a particular learning task.

Analysis of recent studies and publications.

GenAI is increasingly viewed as both a learning support tool and a challenge to assessment in higher education. Kasneci et al. [2] outline opportunities such as personalised learning alongside risks related to bias, misuse, and AI literacy. Chan and Hu [12] report generally positive student attitudes, but also concerns about accuracy, privacy, ethics, and personal development. Francis et al. [13] emphasise tensions between innovation, academic integrity, equity, and assessment validity, while Liu and Bates [14] call for systematic institutional integration and assessment redesign. These issues are especially important in programming, where executable code may appear successful even when students only partly understand the underlying concepts.

Evidence from programming education remains mixed. Xue et al. [15] found no significant improvement in assignment or post-evaluation performance, although students

relied less on traditional resources when ChatGPT was available. Sun et al. [16] likewise found no significant performance difference, but observed changes in programming behaviour and more positive perceptions after use. In a counterbalanced study, Andleeb et al. [17] reported higher code-quality scores and shorter completion times, while effects on conceptual understanding varied by topic. Garcia's rapid review of 107 documents [3] identified benefits such as tutoring, code generation, feedback, and assessment support, but also risks of overdependence, reduced critical thinking, academic dishonesty, and technical limitations. Prather et al. [18] observed that novice programmers could either guide Copilot effectively or become lost after following incorrect suggestions, with concerns about understanding and dependence. Kazemitabaar et al. [19] found improved code-authoring performance without weaker manual code-modification performance, although delayed retention gains were not significant. In a software-engineering study rather than an educational experiment, DePalma et al. [20] showed that ChatGPT could support Java code refactoring, but its inconsistency and weaker performance on complex changes still required human oversight.

However, it remains insufficiently understood how distinct Human–GenAI interaction modes, rather than AI access alone, shape the trade-off between product-oriented and learning-oriented outcomes in programming education. This gap is also practically important: international guidelines expect AI to support rather than replace students' reasoning, yet empirical studies in programming education rarely compare interaction modes that differ precisely in how much of this reasoning is delegated to AI.

The research goal and the research objectives.

Given this, the purpose of the article is to compare product quality, completion time, and measured understanding across three Human–GenAI task–mode combinations in an introductory programming course.

This article reports a classroom experiment comparing three interaction modes in an introductory object-oriented programming (OOP) course. In the No AI mode, students solved a programming task independently. In the AI Tutor mode, students could ask GenAI for hints, explanations, and guiding questions, but not for a complete solution. In the Direct AI mode, students could use GenAI without restrictions, including complete code generation. The study focuses on the trade-off between product-oriented outcomes, such as code quality and completion time, and learning-oriented outcomes, especially measured understanding.

To achieve this goal, the following research objectives were set, formulated as four research questions (RQ1–RQ4):

1. How do the No AI, AI Tutor, and Direct AI task–mode combinations differ in completion time, code quality, measured understanding, and perceived difficulty?
2. To what extent do product-oriented outcomes and learning-oriented outcomes diverge across the three task–mode combinations?
3. How do students' subjective judgements of effectiveness, ease, and speed align with their observed task-level outcomes?
4. What reflection themes contextualise students' experiences of the three interaction modes?

The article contributes to the literature in three ways. First, it treats Human–GenAI interaction mode as a central construct rather than reducing the problem to AI access versus no access. Second, it compares multiple outcome dimensions within the same classroom setting, including time, code quality, measured understanding, perceived difficulty, and student reflection. Third, it offers practical implications for programming assessment in AI-rich contexts, where product quality alone may overstate what students have learned.

2. THE THEORETICAL BACKGROUNDS

2.1. From AI access to interaction modes

The distinction between Direct AI and AI Tutor interaction is pedagogically important. Direct AI allows students to request complete or near-complete solutions. It is likely to optimise task completion, because the tool can generate code, repair errors, and suggest structures that satisfy the rubric. AI Tutor interaction has a different goal: it provides scaffolding without directly replacing the learner's reasoning process. A tutor-like AI may ask questions, suggest steps, explain concepts, and guide debugging while leaving the student responsible for writing and understanding the code.

This distinction connects to work on scaffolding, intelligent tutoring systems, and cognitive engagement. The ICAP (Interactive, Constructive, Active, Passive) framework distinguishes passive, active, constructive, and interactive forms of engagement, with deeper learning expected when students construct explanations or interact around ideas rather than merely receive information [21, pp. 220–221]. Applied to programming with GenAI, Direct AI may support a more product-centred workflow, while AI Tutor may preserve more constructive or interactive engagement. However, this is not automatic. Poorly designed tutor-like interactions may still over-disclose answers, while unrestricted Direct AI can support learning if students actively interrogate, modify, and explain the generated code. Preliminary evidence also suggests that the quality of cognitive engagement during AI tutor conversations may predict near-transfer performance [22, p. 40].

2.2. Product-oriented and learning-oriented outcomes

A core premise of this study is that product quality and learning are related but not identical. The learning-versus-performance distinction shows that conditions producing strong immediate performance may not produce durable understanding or transfer [23, pp. 176–177]. Similarly, cognitive load theory suggests that reducing task friction can help learners manage complex work, but if support removes too much of the necessary reasoning, it may also reduce the cognitive activity that contributes to learning [24]. In AI-assisted programming, this creates a possible trade-off: Direct AI may improve code quality and speed, but the student's measured understanding may not increase to the same degree.

This article therefore adopts a trade-off framework. It distinguishes product-oriented outcomes (code score, completion rate, efficiency), learning-oriented outcomes (measured understanding), process outcomes (time and perceived difficulty), and subjective outcomes (students' judgements of effectiveness, ease, speed, and reflection themes). This framework aligns with the special issue focus on the nature of Human–GenAI interaction, its risks and opportunities, and possible trade-offs between educational outcomes.

3. RESEARCH METHODS

3.1. Research design

The study used a classroom-based within-subject design. Each participant completed three C# object-oriented programming tasks under three Human–GenAI interaction modes: No AI, AI Tutor, and Direct AI. The design prioritised ecological validity: the experiment was conducted in a real course context, with students completing tasks under classroom conditions rather than in a laboratory.

The main methodological limitation is that task and mode were paired. Task A was always completed without AI, Task B under AI Tutor conditions, and Task C under Direct AI conditions. The order of these task–mode combinations was rotated across three groups so that each mode appeared in the first, second, and third position exactly once. This reduced order effects and within-session learning effects, but it did not separate task effects from mode effects. For this reason, the findings are interpreted as task–mode combination differences rather than isolated causal effects of the interaction mode.

3.2. Participants and group rotation

The final dataset included 30 students enrolled in an introductory OOP course. Students were divided into three groups of 10. Baseline characteristics were similar across groups: most participants reported beginner or intermediate C# proficiency, most reported regular AI use, and the median self-rated confidence in OOP tasks was 3 on a five-point scale (Table 1). The high prevalence of regular AI use is relevant because students were not AI-naive; Direct AI condition likely reflected a familiar workflow for many participants.

Table 1

Participant characteristics by experimental group

Characteristic	All (N = 30)	G1 (N = 10)	G2 (N = 10)	G3 (N = 10)
Beginner C# level	13 (43.3%)	5 (50.0%)	4 (40.0%)	4 (40.0%)
Intermediate C# level	14 (46.7%)	4 (40.0%)	5 (50.0%)	5 (50.0%)
Upper-intermediate C# level	3 (10.0%)	1 (10.0%)	1 (10.0%)	1 (10.0%)
AI use: sometimes	8 (26.7%)	3 (30.0%)	3 (30.0%)	2 (20.0%)
AI use: regularly	22 (73.3%)	7 (70.0%)	7 (70.0%)	8 (80.0%)
OOP confidence, M (SD)	3.00 (0.74)	3.00 (0.82)	3.10 (0.74)	2.90 (0.74)
OOP confidence, Mdn [IQR]	3.00 [3.00–3.00]	3.00 [3.00–3.00]	3.00 [3.00–3.75]	3.00 [2.25–3.00]

Note. OOP confidence was measured on a five-point self-report scale. Values for categorical variables are reported as n (%). M = mean; SD = standard deviation; Mdn = median; IQR = interquartile range.

To control for order effects, the three task–mode combinations were counterbalanced across the three groups (G1–G3) using a Latin-square design. Each combination – No AI with Encapsulation, AI Tutor with Inheritance, and Direct AI with Polymorphism – was completed once in every ordinal position, so that no combination was consistently attempted first, second, or last. Figure 1 summarises the resulting rotation.

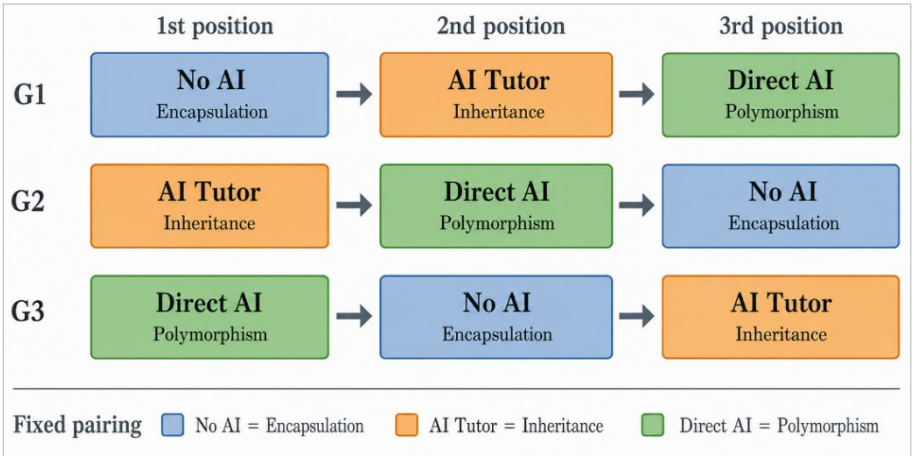

Figure 1. Task–mode rotation across groups (Latin-square design)

As Figure 1 shows, every task–mode combination appears once in each row and once in each column, which balances the sequence in which the modes were experienced and limits the risk that fatigue, practice, or warm-up effects favour any single mode. It is important to note, however, that this design counterbalances only the order of the combinations, not the pairing of task and mode itself: because each mode was always tied to the same task, the observed differences should be interpreted as task–mode combination effects rather than isolated effects of interaction mode – a constraint discussed further in the Limitations.

3.3. Interaction modes and programming tasks

In the No AI condition, students solved the task independently without GenAI assistance. In the AI Tutor condition, students were allowed to use GenAI only for tutor-like support: hints, explanations, guiding questions, and conceptual clarification. Requesting a full or near-complete solution was prohibited. Students were instructed either to use a provided prompt configuring the model as a C# tutor or, where available, to select a built-in AI Tutor mode. The prompt instructed the model to provide hints, explanations, and guiding questions without generating a complete solution. In the Direct AI condition, students could use GenAI without restrictions, including complete code generation, debugging, and solution revision.

The three tasks targeted core OOP concepts in C#. Task A required students to implement a SmartLight class using encapsulation, including a private brightness field and validation of values from 0 to 100. Task B required an inheritance structure for delivery types, with a base Delivery class and derived delivery classes overriding a virtual method. Task C required a polymorphic payment processing system using an abstract base class and derived classes for different payment methods. After each task, students answered understanding questions without AI assistance.

3.4. Measures

Code score was assessed using a rubric with four components: correctness (4 points), OOP principles (3 points), code quality (2 points), and validation/error handling (1 point), for a maximum of 10. Explanation quality was scored separately and used in component-level analysis but not included in the code score total. Understanding score was based on four post-task items answered without AI: conceptual, applied, reasoning, and transfer questions, each worth 1 point, for a maximum of 4. Total score was calculated as code score plus understanding score (maximum 14). Because code score contributed 10 of the 14 available points, the total score was weighted primarily toward product quality. The raw-score product–understanding gap was calculated as code score minus understanding score and was used only for within-study comparisons across the task–mode combinations.

Completion time was calculated from student-reported start and end times. Values above 60 minutes were prespecified for exclusion as potentially including non-active time; no observations met this criterion. Score-based analyses retained all observations. Perceived difficulty was measured after each task on a five-point scale. It is interpreted as the perceived difficulty of completing the task under a specific interaction mode, not as a pure measure of intrinsic task difficulty.

Student perceptions were collected in a final reflection section. Students selected which mode felt most effective, easiest, and fastest. Open-ended reflections were coded using binary thematic indicators following a thematic analysis approach [25, p. 87]. Themes included learning, understanding, speed, ease, AI helpfulness, AI dependence, value of AI Tutor, value of Direct AI, value of No AI, and academic integrity concerns.

3.5. Statistical analysis

The analysis combined descriptive statistics with exploratory inferential checks. Means, standard deviations, medians, interquartile ranges, frequencies, and percentages were calculated by interaction mode. Because each student completed all three task–mode combinations and several variables were non-normally distributed, Friedman tests [26] were used for repeated-measures comparisons, followed by pairwise Wilcoxon signed-rank tests with Holm correction across the three comparisons within each outcome [27, pp. 65–66], where appropriate. Kendall's W [28] and rank-biserial correlations [29] were used as effect size indicators. All inferential tests are treated as exploratory because of the task–mode pairing and small sample size.

4. FINDINGS

4.1. Main task outcomes

The descriptive results show a clear product-oriented advantage for the Direct AI task–mode combination (Table 2). Direct AI had the highest mean code score (9.20/10), the highest mean total score (12.90/14), the highest completion rate (96.7%), and the shortest median completion time (10 minutes). No AI and AI Tutor had similar time profiles, with medians of 26.0 and 23.5 minutes respectively, while AI Tutor showed a higher mean code score than No AI.

Table 2

Primary task outcomes by AI interaction mode

Metric	No AI (N = 30)	AI Tutor (N = 30)	Direct AI (N = 30)
Code score, M (SD)	6.50 (3.54)	7.77 (2.90)	9.20 (1.24)
Code score, Mdn	8.5	9	10
Understanding, M (SD)	3.63 (0.85)	3.57 (0.82)	3.70 (0.60)
Total score, M (SD)	10.13 (3.89)	11.33 (3.31)	12.90 (1.35)
Time, min, Mdn [IQR]	26.00 [20.00–30.00]	23.50 [18.25–29.75]	10.00 [6.00–15.00]
Time, min, M (SD)	26.13 (11.91)	24.17 (10.15)	11.93 (8.21)
Perceived difficulty, M (SD)	2.73 (1.14)	2.73 (1.05)	1.90 (1.06)
Fully completed	20 (66.7%)	21 (70.0%)	29 (96.7%)

Note. Code score max = 10; understanding score max = 4; total score max = 14. Time statistics are based on all 30 observations per mode.

Figure 2 shows the distribution of completion time across the three interaction modes.

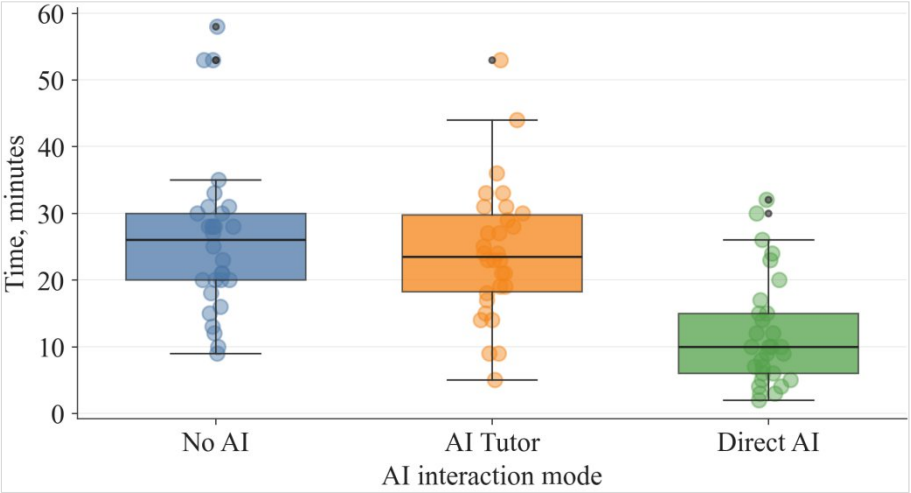

Figure 2. Distribution of completion time by interaction mode

Figure 3 makes the central contrast between code score and measured understanding visible. Code scores increased from No AI to AI Tutor and Direct AI, while understanding scores stayed almost unchanged. Understanding means ranged only from 3.57 to 3.70 on a four-point scale, and all medians were at the maximum score. This indicates a strong ceiling effect and means that the understanding measure was not very sensitive to differences between modes. The result should therefore not be interpreted as proof that all modes produced the same learning, but rather as evidence that the short post-task understanding items did not detect meaningful differences.

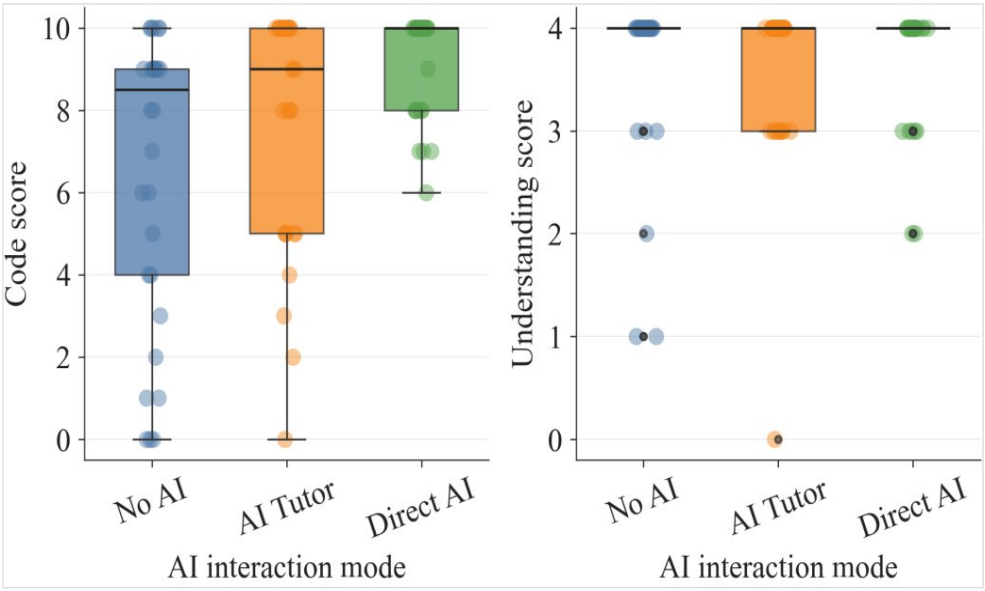

Figure 3. Code score and measured understanding by interaction mode

4.2. Speed–quality–understanding trade-offs

Table 3 summarises the main trade-off pattern. Direct AI produced the strongest efficiency profile: median code score per minute was 1.00, compared with 0.35 for AI Tutor and 0.30 for No AI. The product–understanding gap was also largest under Direct AI, indicating that the code product improved more strongly than measured understanding.

Table 3

Speed–quality–understanding trade-offs by AI interaction mode

Metric	No AI	AI Tutor	Direct AI
Time, min, Mdn [IQR]	26.00 [20.00–30.00]	23.50 [18.25–29.75]	10.00 [6.00–15.00]
Code score, M (SD)	6.50 (3.54)	7.77 (2.90)	9.20 (1.24)
Understanding, M (SD)	3.63 (0.85)	3.57 (0.82)	3.70 (0.60)
Product–understanding gap, M (SD)	2.87 (3.37)	4.20 (2.67)	5.50 (1.41)
Code per min, Mdn [IQR]	0.30 [0.12–0.45]	0.35 [0.22–0.47]	1.00 [0.61–1.42]
Understanding per min, Mdn [IQR]	0.14 [0.12–0.20]	0.16 [0.12–0.22]	0.33 [0.23–0.64]
Total per min, Mdn [IQR]	0.46 [0.24–0.64]	0.52 [0.41–0.67]	1.40 [0.85–2.12]
Fully completed	20 (66.7%)	21 (70.0%)	29 (96.7%)
Perceived difficulty, M (SD)	2.73 (1.14)	2.73 (1.05)	1.90 (1.06)

Note. Product–understanding gap is the raw-score difference between the 10-point code score and the 4-point understanding score. Efficiency metrics divide the relevant score by completion time.

Figures 4 and 5 summarise the product–understanding gap and efficiency pattern across modes.

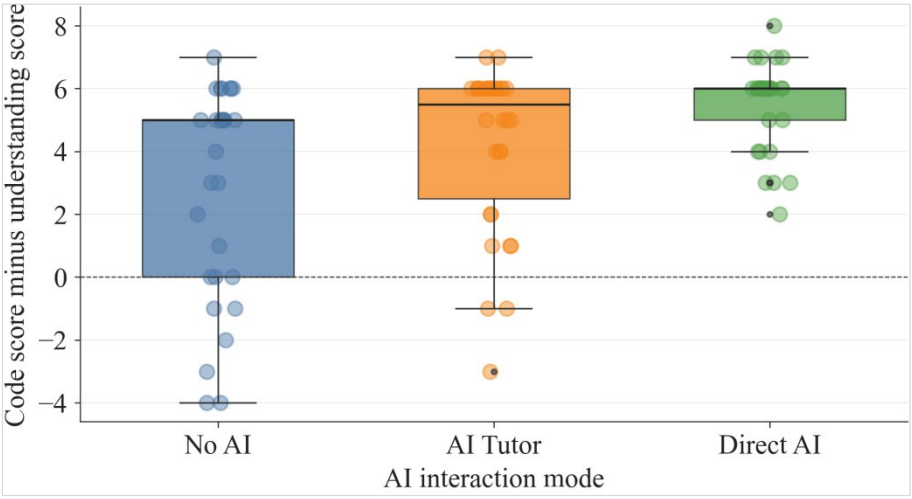

Figure 4. Product–understanding gap by interaction mode

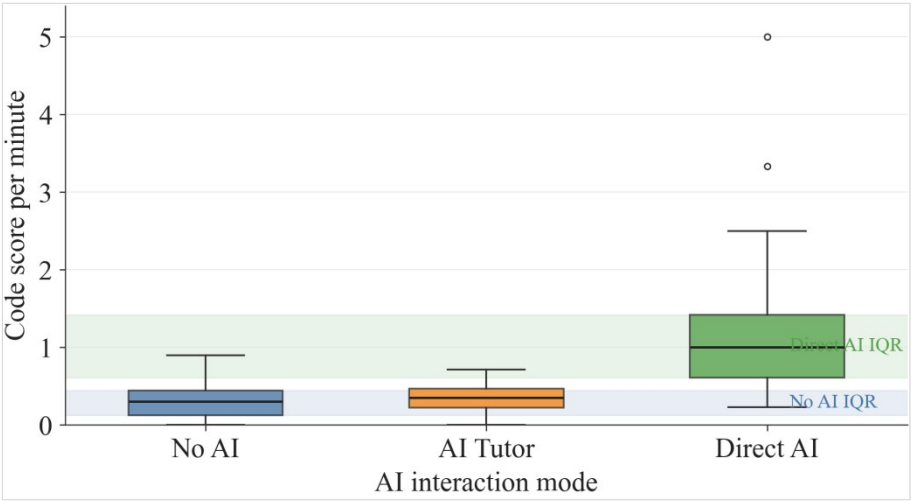

Figure 5. Code score per minute by interaction mode

Component-level results, reported in full in Table A1, show that differences were concentrated in code-related components. Direct AI had the highest scores for correctness, OOP principles, code quality, validation/error handling, and explanation quality. Two components reached zero variance under Direct AI: all students achieved the maximum score for OOP principles and validation/error handling. In contrast, understanding components were close to the maximum across all modes, again supporting the interpretation of a ceiling effect.

4.3. Student perceptions and reflections

Student perceptions complicated a simple performance-based interpretation. Direct AI was clearly experienced as the fastest and easiest mode, but AI Tutor was most often selected as the most effective mode for learning (Table 4). This suggests that students did not define effectiveness only as speed or final code quality.

Table 4

Student perceptions and reflection themes

Section	Item	N (%)
Perceived most effective mode	No AI	1 (3.3%)
	AI Tutor	21 (70.0%)
	Direct AI	8 (26.7%)
Perceived easiest mode	No AI	1 (3.3%)
	AI Tutor	9 (30.0%)
	Direct AI	20 (66.7%)
Perceived fastest mode	No AI	1 (3.3%)
	AI Tutor	2 (6.7%)
	Direct AI	27 (90.0%)
Reflection themes	Understanding	20 (66.7%)
	Learning	19 (63.3%)
	AI as help	14 (46.7%)
	AI dependence / overreliance	10 (33.3%)
	Speed / time saving	10 (33.3%)
	Value of AI Tutor	9 (30.0%)
	Value of No AI	7 (23.3%)
	Academic integrity / ethics	4 (13.3%)

Note. Abridged table for the main manuscript. The complete perception and reflection theme table is provided in Table A2.

Figures 6 and 7 show how students evaluated the modes and what themes appeared in their final reflections.

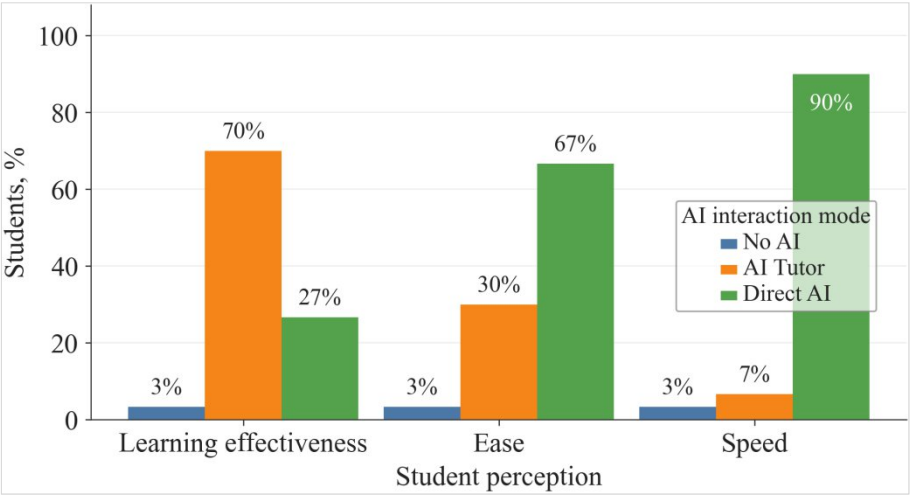

Figure 6. Student perceptions of effectiveness, ease, and speed

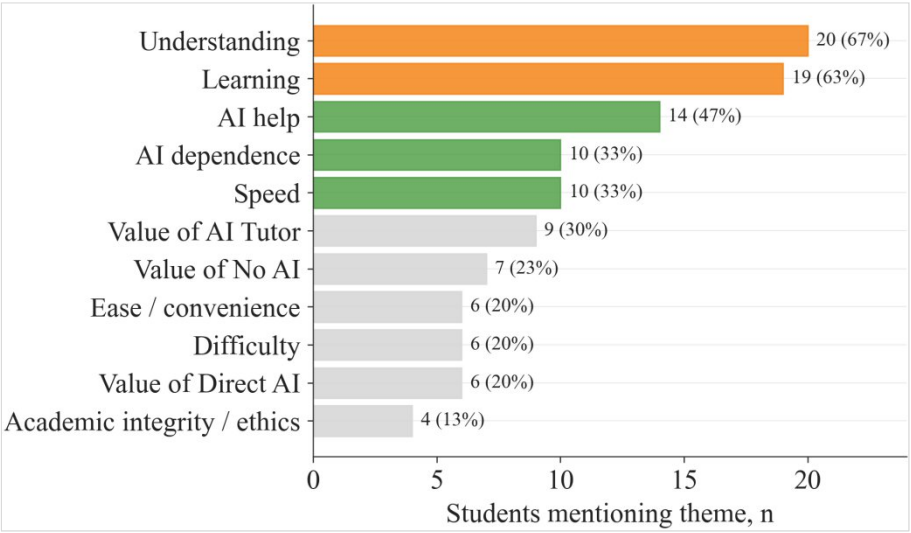

Figure 7. Reflection themes in students' final comments

The reflection data explains why AI Tutor could be perceived as most effective even though Direct AI had stronger task scores. The most common themes were understanding and learning, not only speed or convenience. One third of students also mentioned AI dependence or overreliance, which suggests that students were aware of the risks of using AI as a shortcut. In other words, many students seemed to recognise a distinction between "getting the task done" and "learning through the task."

4.4. Objective–subjective alignment and exploratory tests

The alignment analysis compared student judgements with their own task-level outcomes. Alignment was strongest for speed: 80.0% of students identified a fastest mode that matched their objectively fastest mode. Alignment was lower for perceived effectiveness: 60.0% of students' selected most effective mode matched the mode with their highest total score, and 63.3% matched the mode with their highest code score. This reinforces the interpretation that students' meaning of "effective" was broader than score maximisation. Full alignment results are provided in Table A3.

Exploratory Friedman tests were consistent with the descriptive patterns. Significant differences across task–mode combinations were found for time, code score, total score,

perceived difficulty, product–understanding gap, and all efficiency metrics. No significant difference was found for understanding score ($p = 0.537$; Kendall's $W = 0.021$). The largest effects were for efficiency metrics, especially code score per minute ($W = 0.568$) and total score per minute ($W = 0.491$). Pairwise Wilcoxon tests showed that Direct AI was significantly faster than both No AI and AI Tutor and had significantly higher code and total scores than No AI. AI Tutor did not differ significantly from No AI after correction. The full Friedman and Wilcoxon tables are provided in Tables A4 and A5 (Appendix A).

Figure 8 reports Kendall's W effect sizes for the exploratory Friedman tests.

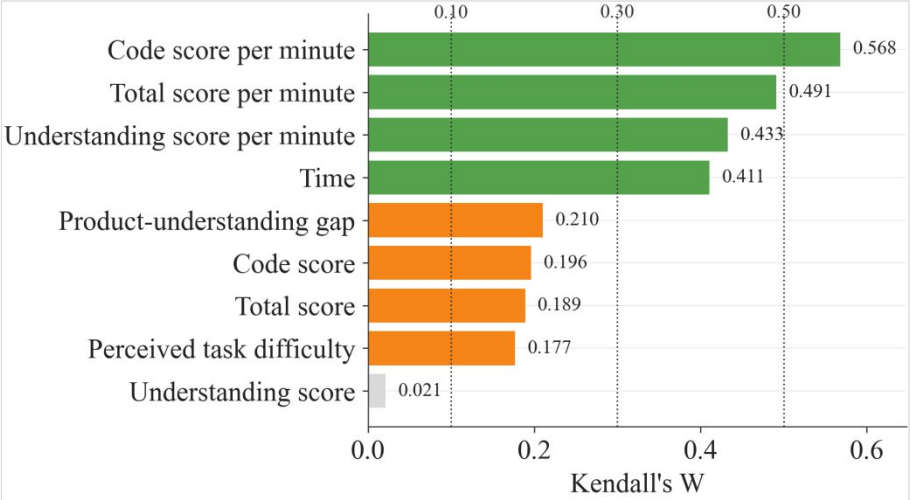

Figure 8. Kendall's W effect sizes from Friedman tests

4.5. Discussion

This study compared three Human–GenAI interaction modes in a classroom programming experiment. The findings show a coherent trade-off. Direct AI was associated with faster completion, stronger code outcomes, higher completion rates, and lower perceived difficulty. However, these product-oriented gains did not correspond to a comparable improvement in measured understanding. At the same time, students most often perceived AI Tutor as the most effective mode for learning, despite its weaker productivity profile. This pattern is important because it shows that "better output" and "better learning experience" are not the same thing.

4.5.1. Direct AI optimises task completion and product quality

The Direct AI task–mode combination produced the strongest immediate performance profile. Students completed the task much faster, achieved higher code scores, and almost all fully completed the assignment. This is consistent with prior work showing that AI code assistants can support productivity, code generation, debugging, and task completion in programming education [15]–[17], [19], [30]. In practical terms, Direct AI can be useful when the instructional goal is rapid prototyping, exposure to solution patterns, or support during complex implementation work.

However, the same pattern creates an assessment problem. Under Direct AI, code scores were not only higher but also less variable. At the component level, some rubric criteria lost discriminating power because all students achieved maximum scores. In an AI-rich environment, code correctness and code structure may reflect the quality of AI support as much as the student's independent competence. This does not make code assessment irrelevant, but it means code should no longer be treated as a sufficient proxy for understanding.

4.5.2. Product gains do not automatically imply learning gains

The most important educational finding is the divergence between code quality and measured understanding. Direct AI improved product-oriented outcomes, but understanding scores remained similar across modes. This fits the learning-versus-performance distinction: conditions that improve immediate performance do not necessarily produce stronger learning or transfer [23, pp. 176–177]. It also fits the concern that AI assistance can reduce cognitive effort in a way that makes task completion easier without requiring the student to construct the solution.

This result should be interpreted cautiously because the understanding measure had a ceiling effect. Most students scored at or near the maximum, so the instrument could not strongly differentiate deeper levels of understanding. The conclusion is therefore not that Direct AI has no learning effect. Rather, the evidence shows that the productivity gains observed in this classroom setting were clearer than the measured understanding gains. Future work should use more demanding understanding measures, including debugging tasks, code modification tasks, oral explanation, delayed retention tests, and transfer problems.

4.5.3. AI Tutor is valued as learning support

Students' subjective evaluations are especially informative. Direct AI was recognised as the fastest and easiest mode, but AI Tutor was selected as the most effective mode by 70.0% of students. This suggests that many students judged effectiveness through a learning lens rather than a productivity lens. Tutor-like interaction may have felt more educationally meaningful because it offered guidance while still requiring students to think, decide, and write code.

This finding aligns with the ICAP framework, which emphasises the value of constructive and interactive engagement [21, p. 229]. AI Tutor interaction can encourage students to ask why, compare alternatives, and reason through a solution. It may therefore support the kind of engagement that students recognise as learning, even when it does not produce the fastest or highest-scoring product. At the same time, perceived effectiveness is not the same as objectively measured learning. The result should be treated as an important signal for instructional design, not as proof that AI Tutor caused deeper learning in this study.

4.5.4. Perceived difficulty is part of the task–mode experience

Direct AI was perceived as easier than the other modes. This should not be interpreted as evidence that the polymorphism task was intrinsically easier. In this design, perceived difficulty reflects the whole experience of completing a task under a particular interaction mode: understanding the task, generating or writing code, debugging, and preparing an explanation. Direct AI reduced the friction of implementation, so students experienced the task as easier. No AI and AI Tutor preserved more of the work of problem solving, so they were experienced as more difficult.

This matters because difficulty can be educationally ambiguous. Lower perceived difficulty may be beneficial when it reduces unnecessary frustration or extraneous cognitive load. But it can also indicate that the learner has offloaded too much of the reasoning process. The negative association between perceived difficulty and the product–understanding gap suggests that easier task–mode experiences were linked to larger divergences between code quality and understanding (Table A6). This is consistent with the idea that productive cognitive friction may sometimes be necessary for learning.

4.5.5. Implications for teaching and assessment

The findings suggest that different AI interaction modes should be selected according to pedagogical purpose. Direct AI may be appropriate for demonstration, rapid prototyping, debugging support, or comparison of alternative designs. AI Tutor may be more appropriate when the goal is to preserve reasoning, explanation, and conceptual engagement. No AI remains

valuable when instructors need to observe independent problem solving or diagnose students' baseline understanding.

For assessment, the implication is direct: code correctness alone is not enough. Programming assessment in AI-rich contexts should include explanation tasks, reasoning questions, debugging tasks, code modification tasks, transfer to new problems, and, where feasible, oral defence or live explanation. Programming instructors are already adapting their approaches in response to AI tools, shifting emphasis toward explanation and process evidence [31, p. 114]. Students may also be asked to document how AI was used, what was accepted or rejected, and why. These practices do not require banning AI; they require aligning the allowed interaction mode with the learning objective and the evidence needed for valid assessment.

4.6. Limitations

The main limitation is task–mode pairing. Encapsulation was always completed without AI, inheritance with AI Tutor, and polymorphism with Direct AI. The balanced rotation distributed these combinations across task positions, but it did not fully separate task effects from mode effects. The findings should therefore be read as differences between task–mode combinations. A future study should use a fully counterbalanced design in which each task appears under each interaction mode across groups.

Second, the sample was small and drawn from one classroom context. The results are useful as exploratory classroom evidence but should not be generalised as stable estimates of AI effects in programming education. Third, the understanding measure showed a ceiling effect, limiting sensitivity to mode differences. Because the product–understanding gap is based on raw scores with different maximum values, its magnitude is scale-dependent and should be interpreted only comparatively within this study. Although the AI Tutor interaction was standardised through a provided prompt or a built-in tutor mode, compliance was not independently verified through interaction logs. Finally, reflection data were coded as thematic indicators rather than analysed through a full qualitative design. Future research should combine richer interaction logs, more sensitive learning measures, delayed post-tests, and deeper qualitative analysis of student reasoning and dependence.

5. CONCLUSIONS AND PROSPECTS FOR FURTHER RESEARCH

The goal of the study, to compare product quality, completion time, and measured understanding across three Human–GenAI task–mode combinations in an introductory programming course, has been achieved. The research objectives were addressed as follows.

1. Regarding the differences between the task–mode combinations (RQ1), Direct AI was associated with the shortest completion time (Mdn = 10 min, compared with 26.0 min for No AI and 23.5 min for AI Tutor), the highest code score (M = 9.20, compared with 6.50 and 7.77, respectively), the highest completion rate (96.7%), and the lowest perceived difficulty (M = 1.90, compared with 2.73 in both other modes). Measured understanding did not differ significantly across modes (M = 3.57–3.70/4; p = 0.537), reflecting a ceiling effect.
2. Regarding the divergence between product-oriented and learning-oriented outcomes (RQ2), the product–understanding gap increased from No AI (2.87) to AI Tutor (4.20) and Direct AI (5.50), and Direct AI showed the highest efficiency (median code score per minute = 1.00, compared with 0.35 for AI Tutor and 0.30 for No AI).
3. Regarding the alignment between subjective judgements and observed outcomes (RQ3), 90.0% of students named Direct AI the fastest mode and 70.0% named AI Tutor the most effective for learning. Alignment with objective results was high for speed (80.0%)

but lower for effectiveness (60.0%), indicating that students understood effectiveness more broadly than score maximisation.

4. Regarding reflection themes (RQ4), students most often referred to understanding (66.7%) and learning (63.3%), while one third mentioned AI dependence or overreliance (33.3%).

Taken together, these results show that different modes of GenAI interaction produce different educational profiles in programming education: unrestricted AI support mainly improves the product, whereas guided interaction is valued by students because it preserves their own reasoning. The central implication is that GenAI should not be integrated into programming education as a single generic tool, but designed as a set of interaction modes, each matched to a specific pedagogical purpose. In AI-rich classrooms, better code is not automatically better learning, and valid assessment must look beyond the final product.

Based on these findings, the following methodological suggestions for using AI in programming education are proposed. When a new concept is introduced, students should first work without AI or with an AI tutor configured to give hints, explanations, and guiding questions rather than complete solutions. Direct AI can be introduced once basic understanding has been demonstrated, mainly for prototyping, debugging, or comparing alternative designs, and should be combined with tasks that require students to explain, modify, or test the generated code. Assessment should cover not only the final code but also explanations, reasoning, and, where feasible, oral defence, and students may be asked to document how they used AI. Finally, the permitted interaction mode should be stated explicitly for each task, in line with international guidance on the responsible use of AI [7].

Prospects for further research follow directly from these findings and the study's limitations. Future work should decouple task and interaction mode using a fully counterbalanced design, so that the effect of interaction mode can be separated from task difficulty. Larger and multi-context samples would allow these patterns to be tested for generalisability. Finally, because measured understanding showed a ceiling effect, future studies should use more sensitive, delayed, or transfer-based measures of learning to test whether the product–understanding gap observed under Direct AI persists over time.

DECLARATION OF GENERATIVE AI AND AI-ASSISTED TECHNOLOGIES IN THE WRITING PROCESS

During the preparation of this manuscript, the author used ChatGPT 5.5 for survey instrument design and experiment planning, and Claude Fable 5 for manuscript revision and checking the consistency of sections between parts of the article. AI tools were not used to generate or analyse research data. The author has reviewed and edited the output and takes full responsibility for the accuracy and integrity of the published work.

ETHICS APPROVAL AND CONSENT

The research was conducted as part of regular classroom activities at the National University of Ostroh Academy. Participation in the study was voluntary. Before proceeding with the study, students were asked in the questionnaire to confirm whether they voluntarily agreed to participate; only those who selected "Yes" could continue. The study was anonymous: no personally identifiable information was collected at any stage, and individual participants cannot be identified from the dataset. Participation had no bearing on course grades or academic standing. Consent for publication is not applicable because the study collected no individually identifiable data, images, or personal details.

CONFLICT OF INTEREST

The author declares that there is no financial or non-financial conflict of interest that could influence the results of the study or their interpretation.

REPRODUCIBILITY AND DATA AVAILABILITY

The anonymised survey dataset supporting the findings of this study is openly available in the Zenodo repository [32] (https://doi.org/10.5281/zenodo.20364465). The dataset contains participant-level records for all 30 students, including task scores, completion times, perceived difficulty ratings, understanding scores, and coded reflection themes; no personally identifiable information is included. A separate reproducibility package [33] (https://doi.org/10.5281/zenodo.20367558) provides the analysis code used to compute all descriptive statistics, exploratory Friedman and Wilcoxon tests, and effect sizes reported in this article, together with the code used to generate Figures 2–8. Both records are published under a Creative Commons Attribution 4.0 International (CC BY 4.0) license. Together, they allow independent verification of all results reported in Tables 1–4 and Tables A1–A6.

REFERENCES (TRANSLATED AND TRANSLITERATED)

[1] A. K. Kofinas, C. H.-H. Tsay, and D. Pike, “The impact of generative AI on academic integrity of authentic assessments within a higher education context,” Br. J. Educ. Technol., vol. 56, no. 6, pp. 2522–2549, 2025, doi: 10.1111/bjet.13585. (in English)

[2] E. Kasneci et al., “ChatGPT for good? On opportunities and challenges of large language models for education,” Learn. Individ. Differ., vol. 103, 2023, Art. no. 102274, doi: 10.1016/j.lindif.2023.102274. (in English)

[3] M. B. Garcia, “Teaching and learning computer programming using ChatGPT: A rapid review of literature amid the rise of generative AI technologies,” Educ. Inf. Technol, vol. 30, pp. 16721–16745, 2025, doi: 10.1007/s10639-025-13452-5. (in English)

[4] H. Bastani, O. Bastani, A. Sungu, H. Ge, Ö. Kabakcı, and R. Mariman, “Generative AI without guardrails can harm learning: Evidence from high school mathematics,” Proc. Natl. Acad. Sci. U.S.A., vol. 122, no. 26, 2025, Art. no. e2422633122, doi: 10.1073/pnas.2422633122. (in English)

[5] G. Kestin, K. Miller, A. Klales, T. Milbourne, and G. Ponti, “AI tutoring outperforms in-class active learning: An RCT introducing a novel research-based design in an authentic educational setting,” Sci. Rep., vol. 15, no. 1, 2025, Art. no. 17458, doi: 10.1038/s41598-025-97652-6. (in English)

[6] UNESCO, “Recommendation on the ethics of artificial intelligence,” UNESCO, Paris, France, 2021. Accessed: Sep. 26, 2026. [Online]. Available: https://unesdoc.unesco.org/ark:/48223/pf0000381137 (in English)

[7] F. Miao and W. Holmes, “Guidance for generative AI in education and research,” UNESCO, Paris, France, 2023, doi: 10.54675/EWZM9535. (in English)

[8] F. Miao, K. Shiohira, and N. Lao, “AI competency framework for students,” UNESCO, Paris, France, 2024, doi: 10.54675/JKJB9835. (in English)

[9] F. Miao and M. Cukurova, “AI competency framework for teachers,” UNESCO, Paris, France, 2024, doi: 10.54675/ZJTE2084. (in English)

[10] European Commission, Directorate-General for Education, Youth, Sport and Culture, “Ethical guidelines on the use of artificial intelligence (AI) and data in teaching and learning for educators,” Publ. Office EU, Luxembourg, 2022, doi: 10.2766/153756. (in English)

[11] “Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence (Artificial Intelligence Act),” Off. J. Eur. Union, L series, Jul. 12, 2024. Accessed: Sep. 26, 2026. [Online]. Available: http://data.europa.eu/eli/reg/2024/1689/oj (in English)

[12] C. K. Y. Chan and W. Hu, “Students’ voices on generative AI: Perceptions, benefits, and challenges in higher education,” Int. J. Educ. Technol. Higher Educ., vol. 20, no. 1, 2023, Art. no. 43, doi: 10.1186/s41239-023-00411-8. (in English)

[13] N. J. Francis, S. Jones, and D. P. Smith, “Generative AI in higher education: Balancing innovation and integrity,” Br. J. Biomed. Sci., vol. 81, 2025, Art. no. 14048, doi: 10.3389/bjbs.2024.14048. (in English)
[14] D. Y. T. Liu and S. Bates, “Generative AI in higher education: Current practices and ways forward,” Assoc. Pacific Rim Univ. (APRU), Jan. 2025. Accessed: Jul. 11, 2026. [Online]. Available: https://www.apru.org/our-work/university-leadership/generative-ai-in-education/ (in English).
[15] Y. Xue, H. Chen, G. R. Bai, R. Tairas, and Y. Huang, “Does ChatGPT help with introductory programming? An experiment of students using ChatGPT in CS1,” in Proc. 46th Int. Conf. Softw. Eng.: Softw. Eng. Educ. Training, 2024, pp. 331–341, doi: 10.1145/3639474.3640076. (in English)
[16] D. Sun, A. Boudouaia, C. Zhu, and Y. Li, “Would ChatGPT-facilitated programming mode impact college students’ programming behaviors, performances, and perceptions? An empirical study,” Int. J. Educ. Technol. Higher Educ., vol. 21, 2024, Art. no. 14, doi: 10.1186/s41239-024-00446-5. (in English)
[17] S. Andleeb, B. Kantorski, and J. Carver, “ChatGPT in introductory programming: Counterbalanced evaluation of code quality, conceptual learning, and student perceptions,” in Proc. 26th ACM Annu. Conf. Cybersecurity Inf. Technol. Educ. (SIGCITE ’25), 2025, pp. 180–186, doi: 10.1145/3769694.3771128. (in English)
[18] J. Prather et al., “‘It’s weird that it knows what I want’: Usability and interactions with Copilot for novice programmers,” ACM Trans. Comput.-Hum. Interact., vol. 31, no. 1, 2023, Art. no. 4, doi: 10.1145/3617367. (in English)
[19] M. Kazemitabaar, J. Chow, C. K. T. Ma, B. J. Ericson, D. Weintrop, and T. Grossman, “Studying the effect of AI code generators on supporting novice learners in introductory programming,” in Proc. 2023 CHI Conf. Human Factors Comput. Syst., 2023, Art. no. 455, doi: 10.1145/3544548.3580919. (in English)
[20] K. DePalma, I. Miminoshvili, C. Henselder, K. Moss, and E. A. AlOmar, “Exploring ChatGPT’s code refactoring capabilities: An empirical study,” Expert Syst. Appl., vol. 249, 2024, Art. no. 123602, doi: 10.1016/j.eswa.2024.123602. (in English)
[21] M. T. H. Chi and R. Wylie, “The ICAP framework: Linking cognitive engagement to active learning outcomes,” Educ. Psychol., vol. 49, no. 4, pp. 219–243, 2014, doi: 10.1080/00461520.2014.965823. (in English)
[22] K. Weatherholtz et al., “Cognitive engagement in GenAI tutor conversations: At-scale measurement and impact on learning,” in Proc. Artif. Intell. Meas. Educ. Conf. (AIME-Con): Works in Progress, 2025, pp. 40–48. [Online]. Available: https://aclanthology.org/2025.aimecon-wip.6/ (in English)
[23] N. C. Soderstrom and R. A. Bjork, “Learning versus performance: An integrative review,” Perspect. Psychol. Sci., vol. 10, no. 2, pp. 176–199, 2015, doi: 10.1177/1745691615569000. (in English)
[24] F. Paas, A. Renkl, and J. Sweller, “Cognitive load theory and instructional design: Recent developments,” Educ. Psychol., vol. 38, no. 1, pp. 1–4, 2003, doi: 10.1207/S15326985EP3801_1. (in English)
[25] V. Braun and V. Clarke, “Using thematic analysis in psychology,” Qual. Res. Psychol., vol. 3, no. 2, pp. 77–101, 2006, doi: 10.1191/1478088706qp063oa. (in English)
[26] J. Richardson, “Friedman test,” in The SAGE Encyclopedia of Educational Research, Measurement, and Evaluation, vol. 4, B. B. Frey, Ed. Thousand Oaks, CA, USA: SAGE, 2018, pp. 709–712, doi: 10.4135/9781506326139.n277. (in English)
[27] S. Holm, “A simple sequentially rejective multiple test procedure,” Scand. J. Statist., vol. 6, no. 2, pp. 65–70, 1979. [Online]. Available: https://www.jstor.org/stable/4615733 (in English)
[28] M. G. Kendall and B. Babington Smith, “The problem of m rankings,” Ann. Math. Statist., vol. 10, no. 3, pp. 275–287, 1939, doi: 10.1214/aoms/1177732186. (in English)
[29] D. S. Kerby, “The simple difference formula: An approach to teaching nonparametric correlation,” Compr. Psychol., vol. 3, 2014, Art. no. 11, doi: 10.2466/11.IT.3.1. (in English)
[30] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos, “Programming is hard – or at least it used to be: Educational opportunities and challenges of AI code generation,” in Proc. 54th ACM Tech. Symp. Comput. Sci. Educ. V. 1, 2023, pp. 500–506, doi: 10.1145/3545945.3569759. (in English)
[31] S. Lau and P. Guo, “From ‘ban it till we understand it’ to ‘resistance is futile’: How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot,” in Proc. 2023 ACM Conf. Int. Comput. Educ. Res., vol. 1, 2023, pp. 106–121, doi: 10.1145/3568813.3600138. (in English)
[32] Y. Kleban, “Survey data from a classroom experiment on Generative AI interaction modes in object-oriented programming education,” Zenodo, 2026, doi: 10.5281/zenodo.20364465. (in English)
[33] Y. Kleban, “Reproducibility package for a classroom study on Human–Generative AI interaction in programming education,” Zenodo, 2026, doi: 10.5281/zenodo.20367558. (in English)
[34] J. Cohen, “A power primer,” Psychol. Bull., vol. 112, no. 1, pp. 155–159, 1992, doi: 10.1037/0033-2909.112.1.155. (in English)

Text of the article was accepted by Editorial Team 11.07.2026

РЕЖИМИ ВЗАЄМОДІЇ ЛЮДИНИ З ГЕНЕРАТИВНИМ ШТУЧНИМ ІНТЕЛЕКТОМ У НАВЧАННІ ПРОГРАМУВАННЯ

Юрій Клебан
Національний університет «Острозька академія», м.Острог, Україна
ORCID ID 0000-0002-7070-5175
yuriy.kleban@oa.edu.ua

Анотація. Інструменти генеративного штучного інтелекту (GenAI) поступово стають частиною щоденного навчання у вищій освіті, особливо в курсах програмування, де студенти можуть попросити пояснення, допомогу з виправленням помилок або готовий код. Тому постає важливе освітнє питання: як саме використовують студенти ШІ?. У дослідженні порівняно три режими взаємодії людини з GenAI у вступному курсі об'єктно-орієнтованого програмування: без ШІ (No AI), ШІ-тьютор (AI Tutor) і прямий ШІ (Direct AI). Тридцять студентів виконали три завдання мовою C# за дизайном, у якому кожен студент проходив усі три режими, а порядок завдань було збалансовано між групами. Вимірювали час виконання, якість коду, виміряне розуміння, суб’єктивну складність і суб’єктивні оцінки студентів. Найсильніші продуктоорієнтовані результати – тобто ті, що стосуються якості готового продукту, – показала комбінація завдання й режиму Direct AI: найкоротший час виконання (Mdn = 10 хв), найвищі бали за код (M = 9,20/10) і найбільша частка повністю виконаних завдань (96,7%). У режимах AI Tutor і No AI час був довшим, а бали за код – нижчими. Водночас виміряне розуміння майже не відрізнялось між режимами (M = 3,57–3,70/4), а всі медіани були на максимумі, що вказує на ефект стелі. Розрив між сирими балами за якість коду й розуміння був найбільшим у режимі Direct AI (M = 5,50), що відображає сильніше зростання балів за код порівняно з виміряним розумінням. Суб'єктивні оцінки студентів додали інший погляд: 70,0% назвали AI Tutor найефективнішим режимом для навчання, а 90,0% визнали Direct AI найшвидшим. Отже, необмежена підтримка ШІ може оптимізувати виконання завдання та якість продукту, але ці переваги не слід сприймати як рівні навчальним здобуткам. Робота з ШІ-тьютором може бути цінною для навчання, оскільки цей режим передбачає збереження більшого простору для власних міркувань студента. Оскільки кожен режим був поєднаний з окремим завданням, результати слід розуміти як відмінності пар «завдання–режим», а не як ізольований причинний вплив самого режиму взаємодії. Дослідження дає емпіричні докази для створення навчальних завдань із GenAI та практик оцінювання, які розрізняють продуктоорієнтовану результативність і результати навчання.

Ключові слова: генеративний штучний інтелект; взаємодія людини і ШІ; навчання програмування; об’єктно-орієнтоване програмування; ШІ-тьютор; результати навчання.

(cc) BY-NC-SA

This work is licensed under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.