Annales Mathematicae et Informaticae
61 (2025) pp. 186–201
DOI: 10.33039/ami.2025.10.004
URL: https://ami.uni-eszterhazy.hu

We are not afraid of the wolf! – AI usage attitudes among Hungarian informatics students

Gábor Kusper, György István Mátyás, Tamás Balla

Eszterházy Károly Catholic University
{kusper.gabor,matyas.gyorgy.istvan,balla.tamas}@uni-eszterhazy.hu

Abstract. We report a multi-instrument study of Hungarian informatics students' attitudes toward generative AI in programming education. Large Language Models (LLMs) are increasingly used to generate code, explain concepts, and support coursework, raising questions about reliability, skill development, and job security. Our study, We Are Not Afraid of the Wolf!, conducted at Eszterházy Károly Catholic University, combined six surveys across two waves with BSc students in Computer Science and Business Informatics. We tested three hypotheses: H1—students are not concerned that increasingly intelligent AI tools will hinder their job prospects; H2—better programmers use AI more effectively for programming tasks; H3—better programmers evaluate AI-generated code more critically. Results: H1 was partially supported—most view AI as a tool and express limited near-term concern, though medium-term uncertainty remains. H2 received partial support: programmer-quality proxies (course grade and self-assessment) showed weak-to-moderate positive associations with output in a 10-minute AI-assisted game development task. H3 was strongly supported: higher-competence students consistently review, debug, and seek to understand AI-generated code. Overall, students adopt a critical yet pragmatic stance: they leverage AI to increase efficiency while maintaining verification routines. The dataset is openly available and will be updated annually. So, there is new hope: higher education in informatics still makes sense, as in our results more skilled programmers outperform less skilled peers even when both use powerful AI tools.

Keywords: LLMs in Programming, AI in Computer Science Education, Student Attitudes Toward AI

AMS Subject Classification: 97P80 Artificial intelligence (educational aspects)

Accepted: October 8, 2025
Published online: October 28, 2025

1. Introduction

When we asked Roland, a 2nd-year Computer Science BSc student, whether he was afraid of AI, his immediate reply was: *"Not at all!"* This unexpected answer made us wonder: perhaps today's students approach AI with far less fear than our generation, which often views it with caution. This observation motivates our study: how do young computer science students perceive AI in their learning and future careers?

Roland's response is not just an isolated remark but a starting point for a broader inquiry. To situate this question, we must consider the wider educational and technological context in which generative AI is emerging.

The rapid development of generative artificial intelligence (AI) has created new opportunities and challenges in computer science education. Large Language Models (LLMs), such as ChatGPT, GitHub Copilot, or Gemini, are increasingly used by students to generate source code, explain programming concepts, or assist in software development projects. While these tools provide immediate support and inspiration, they also raise questions about reliability, dependence, and their long-term effect on programming skills and job security.

The title of this paper, *We are not afraid of the wolf!*, reflects this debate. The "wolf" symbolizes the fear that AI may replace human programmers or reduce the value of their skills. Our research investigates whether Hungarian informatics students share this fear, or whether they instead approach AI with curiosity and acceptance. More specifically, the study explores how students majoring in Computer Science and Business Informatics at Eszterházy Károly Catholic University (EKCU) use AI in programming tasks, exam preparation, and creative projects, and how these experiences influence their self-confidence and career expectations.

The contribution of this paper is threefold. First, we describe our study design and data-cleaning methodology across two survey waves (total $n = 72 + 110$, predominantly second- and third-year Computer Science BSc students). Second, we evaluate the pre-defined hypotheses and report exploratory findings on relationships among AI usage, self-efficacy, creativity, and labor market attitudes. Third, we outline how other researchers can reuse our openly released datasets and bilingual codebook to facilitate replication and secondary analysis. We do not engage in a sustained ethical analysis here, even though the questionnaires included several items on this topic.

This study investigates three hypotheses: H1 (The Wolf): Informatics students are not concerned that increasingly intelligent AI solutions will make it harder for them to find jobs on the labor market. H2 (A New Hope): Good programmers are more effective at using AI solutions to complete programming tasks than less skilled programmers. H3 (An Old Style): Good programmers evaluate AI-generated code more critically.

The results of these hypotheses are presented in Sections 3 and 4. To support reuse and replication, we also make our dataset openly available at: https://zenodo.org/records/17013486.

1.1. Related work

Early work on large language models for code highlighted both their promise and limitations for program synthesis. DeepMind's *AlphaCode* showed that models trained on public code can solve competition-level problems, yet still produce fragile solutions [6]. Following this, research on GitHub Copilot examined usability, correctness, and productivity. User studies found that developers value Copilot for speeding up tasks but can struggle with understanding and debugging longer snippets [13]. Empirical evaluations further assessed correctness across benchmarks [7, 10], and classroom experiments reported substantial efficiency gains in specific tasks [14].

In computer science education, a growing body of work investigates whether LLMs help novices learn to program. Classroom studies suggest that access to ChatGPT can support task completion and perceived understanding, though outcomes vary with scaffolding and assessment design [14]. Systematic reviews indicate generally positive short-term effects on performance and motivation, coupled with concerns about reduced cognitive effort and the need for responsible integration [1, 3]. Further studies analyze students' attitudes toward AI-generated code and the detectability of such submissions in coursework [2, 4]. Recent surveys synthesize open challenges around reliability, assessment, and academic integrity in AI-assisted education [11].

In the Hungarian context, several strands of research highlight the diverse ways in which generative AI is entering education. A recent pilot study explored retrieval-augmented AI tutoring in higher education, demonstrating feasibility and strong learner engagement [9]. Earlier work examined the integration of AI into electronic learning environments, emphasizing its potential to extend e-learning infrastructures [8]. More recently, Toldi investigated adaptive learning enhanced by generative models [12], and Király raised concerns about the erosion of algorithmic thinking in the context of LLM use among computer science students [5]. Our study adds to this emerging body of Hungarian research by providing empirical, multi-instrument evidence from informatics undergraduates on AI usage patterns.

2. Methodology

This study investigates how undergraduate informatics students use and perceive generative AI in programming-related contexts. We combine questionnaires with short, task-based activities to address three predefined hypotheses:

- labor market attitudes (H1 – The Wolf),
- programming proficiency and effective AI use (H2 – A New Hope), and
- critical evaluation of AI-generated code (H3 – An Old Style).

Data were collected in two survey waves during regular classes and lab sessions at EKCU in Spring 2025. Participation was voluntary; informed consent and institutional ethics approval were obtained. Across the two waves, we obtained 72 and

110 responses, respectively, from second- and third-year BSc students in Computer Science and Business Informatics.

We administered six questionnaires across the two waves:

1. **General AI Attitudes** (Wave 1): broad, multi-block survey on AI use attitudes, self-efficacy, perceived overreliance and reliability, ethics, and labor market views.
2. **Game Development Pre** (Wave 1): prior experience, tool familiarity, learning goals, and flow antecedents.
3. **Game Development Post** (Wave 1): perceived speed, reliability, error-fixing effort, satisfaction, and creativity after a **40-minute** *Snake* implementation with AI support.
4. **Rust Pre** (Wave 1): confidence and intended AI strategies before an AI-tutored Rust learning session.
5. **Rust Post** (Wave 1): perceived tutor quality, learning, motivation, and confidence after the session.
6. **Consolidated H2 Questionnaire** (Wave 2): a single-form instrument capturing programmer quality and task performance in a **10-minute** minimalist *Snake* game implementation. The full questionnaire appears in Appendix A.

Unless noted otherwise, items used 5-point Likert scales; several binary items (Yes/No) and free-text questions were also included. In task settings, students were allowed to use AI (e.g., ChatGPT or Copilot) to the extent specified in the activity instructions.

The Wave 1 game task required implementing a minimalist *Snake* game within **40 minutes** (core mechanics mandatory; optional features such as score counters allowed). The Wave 2 task used an intentionally **10-minute** time box to elicit rapid AI-assisted development: students implemented as many minimal functions as possible and reported them both as a checklist and as a single total. The Rust activity followed a *pre–post* design with an AI tutor: the pre-questionnaire captured prior confidence and plans for AI use; the post-questionnaire captured perceived learning and tutor experience.

3. Results of the first wave

In this paper we present the results of two data collection waves. This chapter presents the result of the first one.

3.1. Data collection and cleaning

Survey-based analyses are sensitive to careless responding and incomplete records. Since our goal was to capture *motivated* respondents, we implemented a two-stage

cleaning strategy applied to all questionnaires (actual removals occurred in the General AI Attitudes survey and in Rust Pre):

- **Motivation filter (missingness):** rows were retained only if at least **50%** of the *numerically convertible* items (grades, Likert, averages, binary answers (e.g., Yes/No)) were answered. Free-text fields did not count toward this ratio.
- **Uniformity filter (entropy):** we computed the effective number of unique responses as $\exp(H)$, where H is the Shannon entropy (natural logarithm) of the respondent's numeric answer distribution. Binary items were coded as 1/0; free-text and other non-numeric values were ignored for these calculations. If $\exp(H) \leq 1.3$, the row was excluded, indicating near-constant responding (e.g., almost all 4s with occasional deviations).

This two-step filtering left **36 valid responses** out of 71 raw entries in the General AI Attitudes survey. For the other questionnaires, the same checks were run; only *Rust Pre* required exclusions due to uniformity, while the game-related and *Rust Post* datasets remained essentially unaffected.

The rationale behind this procedure was to exclude respondents who provided low-effort answers, either by skipping large parts of the survey or by mechanically repeating the same response option. By focusing on motivated participants, the analysis better reflects the genuine attitudes and behaviors of students toward AI usage.

Table 1. Data cleaning outcomes per questionnaire.

Questionnaire	Raw	Removed	Final	Primary reason
General AI Attitudes	71	35	36	Missing data
Game Development Pre	38	0	38	–
Game Development Post	37	0	37	–
Rust Pre	34	7	27	Uniformity
Rust Post	28	0	28	–

3.2. Analysis plan and validity

For scale-level reliability we computed Cronbach's α on Likert-type items (integer responses in $[1, 5]$). To handle missingness consistently across questionnaires, we applied a simple inclusion rule for reliability: keep items with $\geq 80\%$ non-missing responses and rows with $\geq 80\%$ non-missing across the selected items, then compute α on complete cases.[1]

[1]This avoids unstable estimates due to sparse item coverage; exploratory factor analysis is outside this paper's scope.

Table 2. Reliability summary per questionnaire (Likert items only; 80% item/row rule).

Questionnaire	Final n	Items used	Rows used	α
General AI Attitudes	36	10	36	0.605
Game Development Pre	38	7	38	0.142
Game Development Post	37	11	37	0.678
Rust Pre	27	8	27	0.765
Rust Post	28	9	28	0.696

The *Rust* questionnaires exhibit satisfactory internal consistency ($\alpha \approx 0.70$–0.77). The *Game Development Post* questionnaire shows acceptable reliability for exploratory analysis ($\alpha \approx 0.68$). The *General AI Attitudes* questionnaire's reduced item count under the 80% rule still yields a moderate α (≈ 0.61), indicating heterogeneous but usable attitudinal indicators. The *Game Development Pre* block was intentionally diverse (experience, tools, motivation), which explains the low α; we therefore treat it as a descriptive questionnaire rather than a single latent scale.

We report descriptive statistics (means, SDs, distributions) for all main variables. For predefined hypotheses we use Pearson correlation on Likert scores (common in education research); as a robustness check we replicate key correlations with Spearman's ρ. For comparisons across AI-usage strata, we use two-sample tests: Welch t or Mann–Whitney, depending on normality/scale. All inferential analyses are computed on the cleaned datasets (Table 1).

Internal validity is limited by self-report and the correlational nature of several analyses. External validity is constrained by a single-institution Hungarian sample. Measurement validity is affected by short scales in some blocks and evolving AI tools. We mitigate these risks via transparent cleaning, a reproducible pipeline, reliability reporting, and robustness checks.

3.3. Results

All analyses use the cleaned General AI Attitudes sample ($n = 36$) and quality-checked task datasets. Before presenting detailed analysis for each hypothesis, we provide a brief overview.

- **H1 (The Wolf):** *Partially supported.* Students mostly view AI as a tool rather than a competitor, though some express medium-term labor market concerns.
- **H2 (A New Hope):** *Not decided in the first wave.* In the first wave, the planned pairing between programming quality and AI effectiveness could not be tested due to missing cross-questionnaire linkage.
- **H3 (An Old Style):** *Supported.* Higher-competence students critically evaluate AI-generated code through review, debugging, and understanding.

3.4. H1: The Wolf

The first hypothesis states: *H1 (The Wolf): Informatics students are not concerned that increasingly intelligent AI solutions will make it harder for them to find jobs on the labor market.*

H1 addresses labor market concerns: whether students fear that AI will reduce programming jobs, whether they see AI as a tool or a competitor, and whether they report direct experiences of job-search difficulties or dismissals.

On the item *"I see AI more as a tool than a competitor"*, students leaned strongly toward the tool perspective ($\bar{x} = 3.58$, $SD = 0.63$, $n = 26$). On the 5-point scale, **7.7%** disagreed (1–2), **26.9%** were neutral (3), and **65.4%** agreed (4–5). Expectations about AI "taking over" programming jobs varied by timeframe:

- **1–2 years:** $\bar{x} = 2.56$, $SD = 0.60$, $n = 18$; 1–2: **50.0%**, 3: **44.4%**, 4–5: **5.6%**.
- **3–4 years:** $\bar{x} = 3.09$, $SD = 0.83$, $n = 23$; 1–2: **30.4%**, 3: **30.4%**, 4–5: **39.1%**.
- **5–6 years:** $\bar{x} = 2.85$, $SD = 0.77$, $n = 26$; 1–2: **38.5%**, 3: **38.5%**, 4–5: **23.1%**.

Direct labor market experiences were rare: **15.2%** reported knowing a programmer who could not find employment ($n = 33$), and only **2.9%** reported knowing someone dismissed in the last 1–2 years ($n = 34$).

The frequency of AI use (e.g., asking an LLM to explain code) showed only weak, non-significant associations with attitudes: with the *tool vs. competitor* item $r = 0.09$ ($n = 24$), and with the *1–2 year takeover probability* item $r = -0.12$ ($n = 18$).

After entropy- and missingness-based filtering, the data indicate that students *do not generally fear* near-term job losses due to AI. Most frame AI as a tool, see a 1–2 year "takeover" as unlikely, and hold mixed expectations for 3–6 years. Direct negative experiences exist but remain in the minority. Overall, H1 is *partially supported*: students do not exhibit widespread short-term fears, yet medium-term uncertainty about AI's labor market impact persists.

3.5. H2: A New Hope

The second hypothesis states: *H2 (A New Hope): Good programmers are more effective at using AI solutions to complete programming tasks than less skilled programmers.*

We used two indicators of programmer quality: the *theoretical programming exam grade* (High-level Programming 2, lecture) and a self-assessment item on programming competence. For AI effectiveness, four questionnaires were designed (*Game Development Pre/Post*, *Rust Pre/Post*) to capture speed, reliability, bug-fixing effort, satisfaction, and tutor quality.

Our intended primary test was to pair each student's quality indicators with their task-level outcomes. However, the questionnaires were not linked by a common identifier: anonymity was prioritized, which prevented direct individual-level associations. As a result, only indirect proxies within single surveys could be analyzed (e.g., tool-vs-competitor framing, debugging practices, or post-task confidence). These showed weak and inconsistent relationships, making them insufficient for a decisive H2 test.

Rather than a simple limitation, this outcome highlights an important methodological lesson: anonymity safeguards must be balanced with the ability to test hypotheses across instruments. To address this, we designed a consolidated questionnaire; see Appendix A. This introduces an optional, privacy-preserving *call sign*. We used this questionnaire in the second wave; see Section 4.

3.6. H3: An Old Style

The third hypothesis states: *H3 (An Old Style): Good programmers evaluate AI-generated code more critically.*

H3 examines whether good programmers evaluate AI-generated code more critically.

Table 3 summarizes post-generation routines. A majority *review* AI code before running it ($\bar{x} = 3.45$, 55.3% select 4–5). Many report *fixing bugs themselves* ($\bar{x} = 3.15$), and an even larger share *try to understand* the generated code ($\bar{x} = 3.65$, 70.6% select 4–5). Consulting *documentation* is less frequent on average ($\bar{x} = 2.81$), and *restyling* to personal conventions sits in the mid-range ($\bar{x} = 3.35$). Systematic *cross-checking across multiple AI tools* is comparatively less common ($\bar{x} = 2.86$). Students report encountering errors in AI-generated code with moderate frequency ($\bar{x} = 3.37$), underlining the need for critical verification steps.

Table 3. Descriptive summary for H3 items (cleaned dataset). For 5-point items, percentages refer to response distribution within each item.

Item	n	Mean	SD	1–2%	3%	4–5%
Review AI code before running	38	3.45	0.69	10.5	34.2	55.3
I fix bugs myself after AI generation	52	3.15	0.75	21.2	42.3	36.5
Try to understand the generated code	17	3.65	0.61	5.9	23.5	70.6
Look up documentation to understand the code	43	2.81	0.73	37.2	44.2	18.6
Restyle the generated code to my conventions	46	3.35	0.71	13.0	39.1	47.8
Compare answers from multiple AI tools	42	2.86	0.75	35.7	42.9	21.4
How often do you find errors in AI-generated code?	68	3.37	0.90	13.2	42.6	44.1

Overall, students show a clear tendency toward *classical* quality assurance be-

haviors when working with AI-produced code: they review and actively try to understand it, often taking responsibility for bug fixing and, to a lesser extent, restyling. Documentation lookup and cross-tool triangulation are less common, which – together with the reported error frequency – suggests room for scaffolding (e.g., checklists or required code reviews) to ensure robust verification of AI outputs.

4. Results of the second wave: Is there a new hope?

This section presents the results of the second survey wave. It specifically addresses *H2 (A New Hope)* which was left undecided in the first wave, by using a single, consolidated questionnaire that enables direct pairing of programmer-quality indicators and task outcomes. The survey was completed by **110** participants, all **2nd- and 3rd-year Computer Science BSc** students. The full questionnaire appears in Appendix A.

4.1. Data collection and cleaning

The task was a time-boxed (10 minutes) *Snake* game implementation with AI support. We focused on two constructs: (i) **programmer quality** (A1, course grade; A2, self-assessed competence, both on 1–5 scales) and (ii) **task performance** recorded in Section C. Specifically, **C3** asked respondents to *tick which game features worked* at the end of the task – via a checklist that included the core mechanic (growth on dot; game over on wall/tail) and optional extras (e.g., score, accelerating snake, pick-up lives, obstacles, level switching, different dot types, sound/music, hall of fame), plus three other slots. We parsed this checklist into a numeric count, denoted *C3_count*. By contrast, **C4** requested a *single total*, the number of features implemented within ten minutes. Thus, *C3_count* derives from the marked features, while *C4* is the participant's self-reported total. The full wording of C3–C4 appears in Appendix A.

The questionnaire also contained two manual timestamps (B5 and C1), but they proved unusable. The *Snake* programming task was introduced in the header of Section C; B5 was the last item before Section C, and C1 the first item within it. Both B5 and C1 asked participants to enter the current time in `hh:mm` format. We intended to compute time-on-task as C1−B5; however, in almost all responses the difference is only one minute, indicating that most participants read the task and immediately filled out C1 before beginning the work. Consequently, we did not use B5/C1 for cleaning.

Instead, we relied on C4, which records the self-reported number of functions implemented for the *Snake* game within 10 minutes using the chosen AI tool. To balance inclusiveness with engagement control, we defined two analysis cohorts based on C4:

- **Permissive cohort:** rows with non-numeric C4 were removed; C4 = 0 was dropped as non-engagement; all cases with C4 ≥ 1 were retained.

- **Strict cohort ("engagement filter"):** same preprocessing but only cases with C4 ≥ 3 were retained, reflecting the expectation that 2nd–3rd year BSc students using an LLM can implement at least three minimal functions in ten minutes.

4.2. Results on H2

We will see the following result in this chapter. **H2 (A New Hope):** *Partially supported.* Higher programmer-quality proxies show weak to weak-to-moderate positive associations with task performance in a 10-minute AI-assisted setting.

Analysis plan: We report Pearson correlations with Spearman's ρ as a robustness check for ordinal/non-normal data. To bound the impact of the engagement filter, all primary associations are presented on both cohorts. Our focal question is whether A1 and A2 are positively associated with *C3_count*. All analyses use the consolidated second wave questionnaire and the cleaning rules above.

Analysis: Across the **Permissive cohort** (C4 ≥ 1; $N = 104$ valid pairs), A1 showed a *weak, positive* trend with C3_count (Pearson $r = 0.19$, $p = 0.052$; Spearman $\rho = 0.17$, $p = 0.081$), while A2 exhibited a *weak-to-moderate, positive* association (Pearson $r = 0.228$, $p = 0.020$; Spearman $\rho = 0.274$, $p = 0.0049$). In the **Strict cohort** (C4 ≥ 3; $N = 86$), the A1–C3_count trend persisted with similar magnitude (Pearson $r = 0.19$, $p = 0.078$; Spearman $\rho = 0.18$, $p = 0.099$), and A2–C3_count remained positive (Pearson $r = 0.170$, $p = 0.118$; Spearman $\rho = 0.212$, $p = 0.050$). In short, stronger programmers (by grade and self-assessment) tended to list more implemented functions in the short AI-assisted task; effects were small and more clearly detectable for self-assessment (A2), especially in the permissive cohort.

Table 4. Summary of H2-related associations.

Pair	Cohort	N	r / ρ	p	Note
A1 ↔ C3_count	Permissive (C4 ≥ 1)	104	$r = 0.19$ / $\rho = 0.17$	$p = 0.052$ / 0.081	Weak, positive trend
A1 ↔ C3_count	Strict (C4 ≥ 3)	86	$r = 0.19$ / $\rho = 0.18$	$p = 0.078$ / 0.099	Similar magnitude
A2 ↔ C3_count	Permissive (C4 ≥ 1)	104	$r = 0.228$ / $\rho = 0.274$	$p = 0.020$ / 0.0049	Weak-to-moderate, positive
A2 ↔ C3_count	Strict (C4 ≥ 3)	86	$r = 0.170$ / $\rho = 0.212$	$p = 0.118$ / 0.050	Borderline ρ

4.3. Interpretation

Taken together, these results offer *partial support* for H2. In a short, AI-assisted programming task students with higher programmer-quality proxies (A1, A2) tend to achieve more implemented functions. The pattern is clearest for the self-assessment proxy (A2), while the grade proxy (A1) shows a consistent but weaker trend hovering near conventional significance thresholds. A reasonable reading is that the two proxies capture partly different aspects of "being a good programmer":

the course grade reflects broader curricular achievement over time, whereas self-assessment may track task-immediate confidence and strategy use that matter when orchestrating AI assistance under time pressure.

Effect sizes are modest, which is unsurprising for at least three reasons. First, the task was intentionally brief (10 minutes), compressing performance variance. Second, our outcome measures trade precision for feasibility: *C3_count* aggregates a checklist of heterogeneous features, and *C4* is a single self-reported total. These two can diverge under time pressure, introducing measurement noise that typically attenuates observable associations. Third, differences in how students prompt, decompose, and verify AI outputs likely add further variance that our minimal instrument does not fully capture. Against this backdrop, the fact that the A2 signal remains detectable, especially in the permissive cohort, suggests a stable underlying tendency rather than a spurious fluctuation.

Reporting results on both a permissive and a strict cohort helps bound sensitivity to low-engagement cases. As expected, the strict cohort yields slightly higher average performance and reduced dispersion, yet the direction and relative strength of the A1/A2 associations persist. This stability implies that the observed tendencies are not driven solely by respondents with near-zero engagement. At the same time, we avoid over-interpreting the magnitude of the effects: the present design was optimized for short, classroom-feasible data collection rather than fine-grained performance measurement.

From a pedagogical perspective, the findings support a pragmatic view of AI in programming education. More skilled students appear to extract somewhat greater task-level benefits from the same class of AI tools, even in a tightly time-boxed setting. For instructors, this points to two complementary actions: (i) continue integrating AI workflows that reward strong problem-solving and code comprehension skills; and (ii) provide scaffolds (e.g., concise review checklists, debugging prompts, minimal test-driven steps) that help less skilled students translate AI outputs into working features more reliably.

In sum, within the constraints of a micro-task and minimalist measurement, we find that programmer-quality proxies relate positively – albeit modestly – to short-horizon AI-assisted output. This aligns with the broader narrative of the paper: AI tools can amplify productivity, but classical strengths in programming still matter.

4.4. Limitations and lessons

The principal instrumentation limitation was timing: because the time fields were positioned before task completion, we could not validate time-on-task. Future iterations should either place time entry at the end of the task block or, preferably, instrument an *automatic timer*. Likewise, manual enumeration of functions can diverge across C3 and C4 under time pressure; parsing and consistency flags help but do not eliminate noise. The consolidated instrument, available in Appendix A, will be deployed annually, and the datasets have been integrated into the *We are not afraid of the wolf!* open data release on Zenodo, enabling replication and

longitudinal analyses.

5. Open data release

Following ethics approval, we publish all datasets as *open-access* to support reuse, replication, and secondary analysis. Open data improves transparency, enables cumulative knowledge, and creates opportunities for new research beyond our own hypotheses (e.g., items on AI ethics, not analyzed here).

All materials are hosted on **Zenodo**, ensuring long-term preservation through OpenAIRE and CERN. The project landing page is at https://zenodo.org/records/17013486. Each release includes cleaned CSV files, metadata, and a bilingual (Hungarian–English) codebook. Updates will be issued annually with versioned DOIs, enabling both single-year and longitudinal analyses.

All data are anonymized and distributed under a CC BY 4.0 license. Users are asked to cite both the dataset DOI and this article.

Wave-specific releases:
First wave dataset: https://zenodo.org/records/17013486
Second wave dataset: https://zenodo.org/records/17218128

6. Conclusion and future work

This study offers a multi-instrument snapshot of how Hungarian informatics undergraduates perceive and use generative AI in programming. We find that students typically frame AI as a tool rather than a competitor and, consistent with classical software-engineering practice, higher-competence students tend to review, understand, and debug AI-generated code (supporting H3). While short-term labor market fears are limited, medium-term uncertainty remains (partial support for H1). A key limitation was our inability to decisively test H2 due to missing cross-form linkage; this was an anonymity-driven design error that we have remedied by introducing an optional, privacy-preserving call sign and a consolidated instrument, see Appendix A.

Beyond H1–H3, we formulated **Observation 1 (Prompt Productivity)**: issuing more prompts to the AI is associated with producing more working features within a tight, 10-minute time box. We observed positive associations between the number of prompts and task output, suggesting that rapid, iterative interaction can be beneficial in short-horizon, AI-assisted coding.

We also note **Observation 2 (Better Programmers Use Better Tools)** emerging from group-level patterns: more skilled programmers may gravitate toward, or extract more value from higher-yield AI tools. While suggestive, this requires controlled studies to disentangle self-selection from tool effects.

We plan to evaluate these observations in future work. We also plan to repeat the surveys in the upcoming academic years to build a larger dataset.

References

[1] Y. Albadarin, M. Saqr, N. Pope, M. Tukiainen: *A systematic literature review of empirical research on ChatGPT in education*, Discover Education 3.1 (2024), p. 60, doi: 10.1007/s44217-024-00138-2.

[2] C. K. Y. Chan, W. Hu: *Students' voices on generative AI: Perceptions, benefits, and challenges in higher education*, International Journal of Educational Technology in Higher Education 20.1 (2023), p. 43, doi: 10.1186/s41239-023-00411-8.

[3] R. Deng, M. Jiang, X. Yu, Y. Lu, S. Liu: *Does ChatGPT enhance student learning? A systematic review and meta-analysis of experimental studies*, Computers & Education 227 (2025), p. 105224.

[4] M. Hoq, Y. Shi, J. Leinonen, D. Babalola, C. Lynch, T. Price, B. Akram: *Detecting ChatGPT-generated code submissions in a CS1 course using machine learning models*, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, 2024, pp. 526–532, doi: 10.1145/3626252.3630826.

[5] S. Király, E. Troll: *Algorithmic Thinking at Risk? A Case Study on LLM Use Among Computer Science Students*, in: Proceedings of the International Conference on Formal Methods and Foundations of Artificial Intelligence (FMF-AI), Eszterházy Károly Catholic University, 2025, url: https://uni-eszterhazy.hu/fmf/m/abstracts.

[6] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. de Masson d'Autume, I. Babuschkin, X. Chen, P. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. S. Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, O. Vinyals: *Competition-level code generation with AlphaCode*, Science 378.6624 (2022), pp. 1092–1097, doi: 10.1126/science.abq1158.

[7] A. Mastropaolo, L. Pascarella, L. Ponzanelli, M. Tufano, G. Bavota: *On the Robustness of Code Generation Techniques: An Empirical Study on GitHub Copilot*, in: 45th IEEE/ACM International Conference on Software Engineering (ICSE), 2023, pp. 2149–2160, doi: 10.1109/ICSE48619.2023.00181.

[8] G. Molnár, Z. Szűts: *Use of artificial intelligence in electronic learning environments*, in: 2022 IEEE 5th International Conference and Workshop Óbuda on Electrical and Power Engineering (CANDO-EPE), IEEE, 2022, pp. 000137–000140.

[9] R. Németh, A. Tátrai, M. Szabó, P. T. Zaletnyik, Á. Tamási: *Exploring the use of retrieval-augmented generation models in higher education: A pilot study on artificial intelligence-based tutoring*, Social Sciences & Humanities Open 12 (2025), p. 101751.

[10] N. Nguyen, S. Nadi: *An Empirical Evaluation of GitHub Copilot's Code Suggestions*, in: Proceedings of the 19th International Conference on Mining Software Repositories (MSR), 2022, pp. 1–5, doi: 10.1145/3524842.3528470.

[11] N. Raihan, M. L. Siddiq, J. C. Santos, M. Zampieri: *Large language models in computer science education: A systematic literature review*, in: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1, 2025, pp. 938–944.

[12] L. Toldi: *Generative AI-Enhanced Adaptive Learning: Integrating GPT Models for Personalized Content and Feedback*, in: Proceedings of the International Conference on Formal Methods and Foundations of Artificial Intelligence (FMF-AI), Eszterházy Károly Catholic University, 2025, url: https://uni-eszterhazy.hu/fmf/m/abstracts.

[13] P. Vaithilingam, T. Zhang, E. L. Glassman: *Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models*, in: CHI Conference on Human Factors in Computing Systems Extended Abstracts, 2022, doi: 10.1145/3491101.3519665.

[14] Y. Xue, Y. Guo, Y. Jia, Y. Huang: *Does ChatGPT Help With Introductory Programming? An Experiment of Students Using ChatGPT in CS1*, in: Proceedings of the 46th International conference on software engineering: software engineering education and training, 2024, pp. 331–341, doi: 10.1145/3639474.3640076.

A. Questionnaire on attitudes toward AI use in programming

This single questionnaire captures (A) baseline programmer quality, (B) general AI use attitudes in programming, (C) task-specific outcomes after a short coding exercise, (D) post-generation quality assurance attitudes, (E) attitudes toward AI assistance in programming, and (F) vibe coding experience.

Administration and ethics. The survey is part of a research study conducted at the EKCU Faculty of Informatics. Participation is entirely voluntary and anonymous. An ethics approval was granted by the EKCU Scientific Committee. *Note: This form is not optimized for mobile. Please fill it on a laptop/desktop with a development environment available.*

Please choose a private *call sign* you can remember (e.g., `AI-User11`). Use the *same* call sign whenever you fill in any similar form. Do *not* include personal data (name, nickname, birth date/age).

Unless stated otherwise, items use a 5-point Likert scale: *1 = Strongly disagree*, *2 = Disagree*, *3 = Neutral*, *4 = Agree*, *5 = Strongly agree.*

Administrative header

ID1. Call sign (e.g., AI-User11): _____

A. Baseline programmer quality.

A1. *High-level Programming 2 (lecture) grade:* **1** (Fail) **2** **3** **4** **5** (Excellent) (select one)

A2. *By my own assessment, I am a good programmer.* (1–5)

B. General AI use attitudes in programming.

B1. *I use large language models (e.g., ChatGPT/Copilot) to generate code.* (1–5)

B2. *Using AI helps me complete programming tasks faster.* (1–5)

B3. *With AI, I finish more tasks within a fixed time.* (1–5)

B4. *AI use reduces the effort I spend on routine coding.* (1–5)

B5. *Please enter the current time (HH:MM), e.g., 11:23:* _____

C. Task-specific outcomes (to fill *after* the following short task). Short task (10 minutes, measure time precisely). *Develop a SNAKE game with AI assistance.*
Core function: The snake grows when it eats a dot; the game ends if it hits a wall or its own tail.
Possible extra features: score; accelerating snake; pick-up lives; obstacles; level switching; different dot types (e.g., speed-up/slow-down/bonus points); sound effects or background music; hall of fame; or any other *cool* feature.

C1. *Please enter the current time (HH:MM), e.g., 11:23:* _____

C2. *Which AI solution did you use for developing the game?* _____

C3. *Which features work (check all that apply)?*

- ☐ Core function (growth on dot; game over on wall/tail)
- ☐ Score
- ☐ Accelerating snake
- ☐ Pick-up lives
- ☐ Obstacles
- ☐ Level switching
- ☐ Different dot types (e.g., speed-up, slow-down, bonus points)
- ☐ Sound effects or background music
- ☐ Hall of fame
- ☐ Other cool feature #1.
- ☐ Other cool feature #2.
- ☐ Other cool feature #3.

C4. *Total number of features implemented in 10 minutes (select one):* 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

C5. *While developing the game, I fixed bugs myself.* (1–5)

C6. *Total number of prompts I issued to the AI:* _____

D. Post-generation quality assurance attitudes.

D1. *I review AI-generated code before running it.* (1–5)

D2. *I try to understand AI-generated code, not just execute it.* (1–5)

D3. *I fix bugs in AI-generated code myself.* (1–5)

D4. *I create (or generate) unit tests for AI-generated code (either before or after generation).* (1–5)

E. Attitudes toward AI assistance in programming.

E1. *I see AI more as a tool than a competitor.* (1–5)

E2. *After using AI, I feel more confident that my solution is correct and complete.* (1–5)

F. Vibe coding experience.

F1. *I enjoyed working with AI support during the Snake development task.* (1–5)

F2. *I experienced "vibe coding" (a creative flow state with AI) while working on the task.* (1–5)