The Clash of Codes: From Peer-to-Peer Duplication to AI-Generation in Introductory Programming Assignments

Jose Maria Zuzarte-Reis
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
j.m.zuzarte.reis.claver@student.vu.nl

Mahbod Tajdini*
Universität Zürich
Zürich, Switzerland
mahbod.tajdini@uzh.ch

Mauricio Verano Merino
Vrije Universiteit Amsterdam
Amsterdam, The Netherlands
m.verano.merino@vu.nl

Abstract

Generative AI tools such as ChatGPT, GitHub Copilot, and Gemini have rapidly influenced programming education by providing instant code generation and problem-solving support. This study investigates the potential impact of these tools on an introductory Python programming course by conducting an observational analysis of two cohorts in 2022 and 2024. In total, we analyzed a corpus of 1,614 submissions across three assignments. To study the possible influence of generative AI tools, we created baseline solutions for each assignment using ChatGPT and Gemini. We then analyzed code similarity between the baseline solutions and the students' submissions, as well as syntax errors across both cohorts. For Assignment 2, peer-to-peer duplication decreased by 60.11% between 2022 and 2024. Among the 2024 submissions, we observe substantial similarity to fixed AI baselines, depending on task complexity. Although this pattern is consistent with the convergence toward AI-generated solutions, we lack direct measures of tool use and make no causal claims. The number of syntax errors did not differ significantly between the two cohorts. These findings suggest a shift in programming education, from direct student copying toward solutions that increasingly resemble AI-generated outputs. This shift emphasizes the need for updated assessments and policies to promote fairness, integrity, and meaningful learning in introductory programming courses.

CCS Concepts

• **Applied computing** → **Education**; • **General and reference** → **Empirical studies**; Metrics.

Keywords

AI in Education, Code Similarity, Generative AI, Plagiarism Detection, Programming Education, Syntax Correctness

ACM Reference Format:
Jose Maria Zuzarte-Reis, Mahbod Tajdini, and Mauricio Verano Merino. 2026. The Clash of Codes: From Peer-to-Peer Duplication to AI-Generation in Introductory Programming Assignments. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-SEET '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3786580.3786955

*Both authors contributed equally to this research.

ICSE-SEET '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2423-7/2026/04
https://doi.org/10.1145/3786580.3786955

1 Introduction

The use of generative artificial intelligence (GenAI) tools such as ChatGPT, GitHub Copilot, and Google Gemini is transforming modern life. In educational settings, these tools are changing how students approach learning. For instance, Qadir [31] discusses how ChatGPT can provide instant feedback and personalized support, while raising concerns about fairness and integrity in engineering education. Particularly in introductory programming courses, these tools can offer immediate support with exercises and problem-solving assignments. Their use has become increasingly common among university students [34], alongside the growing popularity and easy access to AI-powered code editors such as Cursor and Windsurf [3, 14].

Beyond supporting coding tasks, generative AI has the potential to create personalized and adaptive learning experiences by tailoring content to individual learners, taking into account their needs, habits, and preferences. At the same time, its use in education raises critical concerns regarding bias, data privacy, and accountability [26]. The rise of generative AI tools such as ChatGPT raises questions about the sustainability of traditional assessment methods. Generative AI tools have altered how students approach education and learning. Recent large-scale surveys confirm its widespread adoption in higher education. Respondents report frequent use for tasks such as information retrieval and paraphrasing, while also emphasizing ethical concerns and risks of academic dishonesty [45].

Introductory programming courses are recognized as challenging for many students, who often struggle with misconceptions related to syntax, concepts, and problem-solving strategies [32]. Despite these difficulties, such courses play a foundational role in developing an understanding of how computers and software work, while simultaneously training cognitive skills such as logical and abstract thinking and reasoning [25]. In these courses, practical programming exercises are key learning activities and are often used as formative or summative assessment mechanisms [25]. Recent research highlights that AI tools can improve student learning by supporting study habits, time management, and feedback processes, while also contributing to better academic outcomes [42]. However, these benefits are accompanied by challenges such as over-reliance and difficulties in integrating AI into traditional teaching approaches, suggesting that AI should complement rather than replace conventional strategies [42]. Meanwhile, teachers and examination boards face the growing challenge of detecting AI-assisted cheating, which threatens fairness, integrity, and the credibility of academic assessment. Recent work highlights not only the risks posed by AI misuse but also emphasizes the need for extended research on detection strategies, ethical education, and institutional

responses [44]. Although recent work discusses the risks and opportunities of generative AI tools in higher education [6], there is little empirical evidence comparing two groups of students before and after the widespread availability of generative AI tools under similar conditions.

To achieve this goal, this paper investigates the impact of generative AI tools in computer programming courses, focusing on individual assignments, through an observational comparison. We analyze two groups of students from a Dutch university: the 2022 class, with limited access to GenAI tools, and the 2024 class, with widespread access to GenAI tools. Both groups completed the same assignments under the same conditions, eliminating variation due to course design. Our analysis focuses on the similarity of student code to AI-generated solutions, the syntax correctness of submissions, and the similarity between student submissions using JPlag [38]. The main contributions of this paper can be summarized as follows:

- Data-driven comparison of student behavior and performance across two different year cohorts, focusing on student-to-student code similarity (JPlag) and syntax correctness, to assess potential shifts in programming practices with the rise of generative AI.
- Analysis of student similarity to fixed AI-generated baselines using two popular LLMs (ChatGPT and Gemini). This provides a reference for quantifying the similarity to generative AI outputs.
- Observational analysis of distributional shifts in code similarity across student-to-student and student-to-AI, along with syntax error rates, enabling a comparative view of the 2022 and 2024 cohorts.
- Discussion of implications for assessment design and academic integrity in introductory programming courses in light of the availability of GenAI.

2 Related Work

2.1 Generative AI in Education

The adoption of generative AI tools in higher education has accelerated rapidly. Simaremare et al. [34] surveyed 1,157 students in Indonesia and found that while 70% of the respondents were aware of generative AI, nearly all had used such tools to support their learning, with 88.8% reporting use for source code generation. In addition, a large-scale cross-cultural survey confirms widespread awareness and adoption of generative AI in higher education worldwide, while also highlighting variations in perceived benefits and ethical concerns across cultural contexts [45]. Our study extends this line of work by analyzing students' assignment submissions, offering evidence of how generative AI availability might influence similarity to AI-generated baselines.

However, policy responses lag behind adoption. In 2023, UNESCO [40] reported that fewer than 10% of higher education institutions had formal AI policies, although more recent data indicate that this figure has increased to 19%, with another 42% developing policies as of 2025 [41]. A detailed analysis of the top 50 U.S. universities shows that institutions are now responding quickly, with 94% of faculty guidelines emphasizing course-specific rules, consistently highlighting academic integrity and privacy concerns, while also calling for flexible, stakeholder-specific approaches to balance generative AI opportunities and risks [2]. Our findings provide empirical evidence showing how the availability of generative AI coincides with shifting patterns of code similarity in student submissions. This understanding contributes to the broader discussion of academic integrity and can guide the design of fair and reliable assessments.

In parallel, generative AI is reshaping assessment practices. Sol et al. [36] noted that the widespread availability of AI challenges traditional forms of evaluation, raising concerns about academic integrity while also creating opportunities for personalized assessment, automated feedback, and AI-driven collaboration. To guide sustainable integration, they propose the AI Assessment Scale. Similarly, Smolansky et al. [35] surveyed students and educators and found agreement on which assessments are the most affected, with educators highlighting the need to adapt tasks to assume AI use by students and to foster critical thinking and authentic applications. Our study complements these perspectives by empirically demonstrating how assessment outcomes, specifically code similarity and syntax correctness, shift across cohorts with differing access to generative AI tools. These findings help identify which assignment types remain less affected by AI use and where redesign is necessary to preserve fairness and learning value.

2.2 Generative AI in Programming/CS/AI

Introductory programming courses have long been recognized as challenging, with persistent difficulties related to problem-solving skills, mathematical knowledge, abstraction, debugging, and the lack of personalized feedback [25]. These baseline issues form an essential backdrop for evaluating how generative AI tools may alter learning trajectories and reshape the challenges traditionally encountered in programming education. To understand the scope of AI's influence, we track whether long-standing difficulties such as syntax errors have shifted, observing if these challenges persist despite the widespread availability of GenAI tools.

Building on this context, recent surveys highlight that tools such as ChatGPT, Gemini, and GitHub Copilot are reshaping programming education by offering educational support while simultaneously posing new risks to academic integrity [5]. Our study investigates this duality through code similarity analysis, syntax parsing, and performance metrics, aiming to better understand the influence of generative AI on student work.

One of the most notable challenges for institutions in this area is the difficulty in detecting GenAI-generated code. Pham et al. [28] introduce *MageCode*, a method that combines semantic features extracted from pre-trained models such as CodeT5+ with metric-based techniques. Evaluated on a dataset of more than 45,000 LLM-generated code samples across Python, Java, and C++, it achieved up to 98.46% accuracy with less than 1% false positives, outperforming existing baselines and highlighting promising directions for source code detection. Complementary approaches have also been proposed, such as the use of AI-generated *pseudo-submissions*, which make generative output transparent and comparable to student work, thus improving detection accuracy [4]. Although these methods propose possible detection approaches, our study contributes a classroom perspective by using JPlag [38] and fixed AI baselines to

compare similarity between human-generated and AI-generated code in real student submissions.

Beyond the classroom, adoption studies in software engineering illustrate a similar duality. A survey conducted on Stack Overflow reports that over 70% of developers already use or intend to use tools such as ChatGPT and GitHub Copilot, citing productivity gains, testing support, and improved code comprehension [15]. However, concerns remain about code quality, security, licensing, and the impact on junior developer roles [15]. These findings reinforce the broader implications of generative AI for programming practice, suggesting that while technology can provide significant support, it also raises unresolved challenges in detection, integrity, and professional development. Our work situates these adoption patterns within the educational setting, showing how student practices in introductory programming may anticipate the opportunities and challenges later faced in a professional environment.

3 Study Design

3.1 Goals and Research Questions

The goal of this study is to analyze *the potential influence of generative AI tools in an introductory programming course*, aiming to understand shifting student behaviors and their implications for performance and academic integrity (e.g., plagiarism). We adopt an observational stance, without inferring causality or direct tool usage, and focus on three signals derived from the collected data: (i) Code similarity among students' submissions; (ii) Resemblance to AI-generated baseline solutions; (iii) Syntactic correctness. To guide this analysis, we define the following three research questions.

RQ1: For each assignment, how did the distribution of code similarity among student submissions change between 2022 and 2024?

- *Null hypothesis (H_0):* For each assignment, the distribution of code similarity scores is the same in 2022 and 2024.
- *Alternative hypothesis (H_1):* For each assignment, the distribution of code similarity scores differs between 2022 and 2024.

While RQ1 compares peer-to-peer patterns across both cohorts, the following analysis of AI resemblance is restricted to the 2024 cohort. We exclude 2022 from this specific analysis because the course took place in September 2022, prior to the initial release of ChatGPT (November 2022).

RQ2: Across assignments in 2024, how do students' maximum similarity scores to AI-generated answers differ?

- *Null hypothesis (H_0):* In 2024, for the same students, the distribution of the maximum similarity scores does not differ between assignments (A1, A2, A3).
- *Alternative hypothesis (H_1):* For the same students, at least one assignment has a different similarity score distribution compared to the others in 2024.

RQ3: For each assignment, how do the proportions of syntax failures (AST errors) among student submissions compare between 2022 and 2024?

- *Null hypothesis (H_0):* For each assignment, the proportion of syntax failures is the same in 2022 and 2024.
- *Alternative hypothesis (H_1):* For each assignment, the proportion of syntax failures differs between 2022 and 2024.

3.2 Methodology

An observational study was conducted comparing two independent student groups enrolled in the same *Introduction to Python* course at a Dutch university: class 2022 (presumably low exposure to AI tools) and class 2024 (presumably high exposure to AI tools). Both groups completed the same three assignments under the same instructor and course design, reducing variation due to possible instructional differences. This observational analysis combines the use of JPlag to obtain the distribution of similarity among student submissions, the resemblance to fixed AI-generated solutions from two of the most popular LLMs at the time (ChatGPT and Gemini), and syntax validation. Figure 1 provides an overview of the methodology.

3.3 Dataset

We use data from an introductory Python programming course taught to first-year students in an AI Bachelor's program. In the course, students' progress is assessed through summative (e.g., assignments, a midterm, and a final exam) and formative assessments (e.g., in-class exercises and exercises with automated feedback). In this context, we focus on assignments designed to reinforce the lecture material while developing programming skills through practice. They account for 10% of the final grade, and are completed individually at home. As the course progresses, tasks grow in complexity: early ones have predictable solutions, while later ones admit multiple correct implementations, supporting creativity and reducing code similarity. Appendix A outlines the core topics covered in each assignment. In total, we collected *1,762 raw submissions*, including 2022 and 2024. We pre-processed this raw data to remove duplicate entries [1] and anonymize all personal data.

Data Cleaning. To ensure statistical independence of the observations between the two-year cohorts, any student who appeared in 2022 and 2024 was removed from the 2024 dataset. For example, if a student submitted Assignment 1 in both years, these two submissions would be inherently dependent. Removing such cases ensures that comparisons reflect truly independent student cohorts.

Anonymization. Student submissions came in two forms: Python scripts (`.py` files) or Jupyter notebooks (`.ipynb` files). Jupyter notebooks were converted into Python scripts, ensuring that both the original logic and the order of the code cells were preserved. Moreover, all the resulting scripts were anonymized; personal data was removed and the filenames were replaced with hashed IDs.

JPlag Token Threshold. To assess syntactic and semantic similarity, we used *JPlag*, an open-source tool for source code plagiarism detection [38]. JPlag has a minimum token threshold to exclude empty or trivial submissions from the analysis [30].

Final Dataset. After removing duplicates and cleaning the raw data, the resulting dataset contains 1,614 submissions (Table 1).

[1] Students who were enrolled in the course in both years

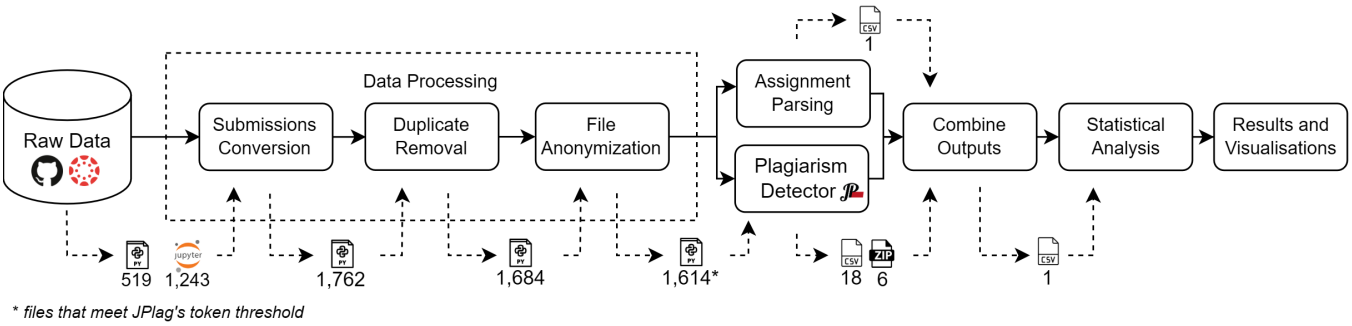

Figure 1: Overview of the study design.

Baseline. For the baseline, we generated the AI solutions for each assignment in March 2025 using *ChatGPT-4* and *Gemini 2.0*. We use the same problem descriptions provided to students.

Assignment	Submissions		Duplicates		JPlag Filter		Total
	2022	2024	2022	2024	2022	2024	
A1	306	389	-	48	3	8	636
A2	254	336	-	30	-	-	560
A3	197	280	-	-	54	5	418
Total	**757**	**1005**	**0**	**78**	**57**	**13**	**1614**

Table 1: Number of submissions used for the analysis.

3.4 Statistical Analyses

All the statistical analyses are conducted using *Python* (version 3.10.11) and various libraries (i.e., NumPy, SciPy, and statsmodels). A fixed random seed (2) was established to maintain reproducibility in all resampling procedures [8]. Because similarity scores (to both student and AI submissions) were non-normally distributed, bounded between 0 and 1, and clustered at the boundary values, we used nonparametric, resampling-based methods throughout analysis. For each research question, the selection of statistical tests was guided by three main considerations: the study design (independent or repeated measures), the scale of the data (continuous bounded [0,1], similarity scores, or binary syntax-failure outcomes), and the presence of non-normal heavily tied distributions. To control the family-wise error rate within each group of related tests, Holm's step-down adjustment was applied to the p-values [18].

4 Code Similarity Between Students in 2022 and 2024 (RQ1)

4.1 Approach

To evaluate whether the distributions of code similarity scores differed between the 2022 and 2024 students for each assignment, all hypothesis tests were performed using a two-sided significance level of $\alpha = 0.05$. The maximum similarity among student submissions was defined as the highest pairwise JPlag similarity score obtained by a student against any other submission within the same assignment and year, bounded in $[0, 1]$. AI-generated baselines were excluded from this analysis to strictly focus on similarity patterns among students.

Primary analysis. To determine whether the similarity-score distributions for 2022 and 2024 are equal in distribution, we apply the k-sample Anderson–Darling test [33]. Since standard asymptotic approximations may be unreliable due to heavy ties and the bounded nature of the data, we derive significance levels using a nonparametric permutation procedure ($B = 10,000$) that randomly shuffles year labels within each assignment [11, 17]. The final p-values are computed using the add-one adjustment $p = \frac{k+1}{B+1}$ to avoid zero values, where k denotes the number of permuted statistics at least as extreme as the observed value.

Multiplicity control. Holm adjustments were applied separately to the families of tests using a family-wise significance level of $\alpha = 0.05$.

4.2 Descriptive Statistics

Table 2 presents the descriptive statistics of the code similarity scores for each assignment per year. The most prominent pattern was observed in A2 for 2022, where 61.42% of submissions had similarity scores of 1.0, compared to only 1.31% in 2024. In contrast, similarity scores of 0.0 were rare in all assignments, never exceeding 1.5%. A1 and A3 exhibited more balanced distributions, although both showed modest increases in similarity scores of 1.0 from 2022 to 2024.

Given that many similarity scores were concentrated at the exact boundaries, representing either completely unique code (0.0) or identical submissions (1.0), we decomposed the analysis into interior-only comparisons and separate tests for boundary proportions to better characterize distributional differences.

	Median		IQR		Proportion = 0 (%)		Proportion = 1 (%)	
	2022	2024	2022	2024	2022	2024	2022	2024
A1	0.947	1.000	0.192	0.191	0.99	1.50	48.18	51.95
A2	**1.000**	0.439	0.656	0.156	0.00	0.00	**61.42**	1.31
A3	0.416	0.692	0.164	0.241	0.00	0.00	0.00	1.45

Table 2: Descriptive statistics for student similarity scores by assignment, 2022 and 2024.

4.3 Primary Analysis

To differentiate between general shifts in solution structure and direct copying, the primary analysis included the Anderson–Darling test to compare overall similarity distributions, alongside boundary proportion tests to specifically monitor the prevalence of structurally independent solutions (0.0) and exact peer-to-peer duplication (1.0).

Based on the results (Table 3), we observe clear differences between cohorts for A2 and A3 after Holm adjustment. For A2, the Anderson-Darling statistic was 135.043 with Holm-adjusted $p < 0.001$, indicating a large and highly significant shift in distribution that points to a structural convergence in student code. Similarly, A3 showed a very large statistic of 88.999 with Holm-adjusted $p < 0.001$, reflecting a substantial distributional change where partial code matches became more frequent or intense. In contrast, A1 showed no significant difference, with a statistic of 0.771 and Holm-adjusted $p = 0.519$.

Interior-only Distributional tests. To test the interior-only distribution, we removed all scores at exactly 0.0 or 1.0. Then, we tested the distributional equality again in the remaining samples (Table 3). After removing the boundary values, all three assignments showed significant differences between 2022 and 2024. A2 remained highly significant (Statistic = 35.543, Holm $p < 0.001$), A3 continued to show a strong effect (Statistic = 87.393, Holm $p < 0.001$), and A1 revealed a smaller but statistically significant difference (Statistic = 3.665, Holm $p = 0.010$). This indicates that, beyond changes at the extreme similarity values, there were also notable shifts in the interior portions of the similarity score distributions.

	Full dataset			Interior-only				
	Statistic	***p* (raw)**	***p* (Holm)**	n_{2022}	n_{2024}	**Statistic**	***p* (raw)**	***p* (Holm)**
A1	0.771	0.519	0.519	154	155	3.665	0.010	0.010
A2	**135.043**	< 0.001	< 0.001	98	302	35.543	< 0.001	< 0.001
A3	**88.999**	< 0.001	< 0.001	143	271	**87.393**	< 0.001	< 0.001

Table 3: Anderson–Darling tests on full dataset (including boundaries) and Interior-only dataset for similarity scores.

Boundary proportion tests. The proportions of submissions with a similarity score of one (1.0) or zero (0.0) are compared using Pearson's χ^2 independence test without Yates continuity correction [1, 24, 27]. When the expected cell count was less than five, or when zeros invalidated the χ^2 test assumptions, Fisher's exact test was applied instead [1, 12, 13]. Also, if both years did not have observations at a given boundary value, the comparison was reported as not applicable ('-').

Table 4 presents the comparison of proportions between years. Based on the results, A2 showed a significant decrease in the proportion of scores equal to 1.0 from 2022 to 2024. After Holm adjustment, *no other boundary effects were significant.*

The results show that significant changes in code similarity patterns between 2022 and 2024 occurred at the same time as the emergence of generative AI tools. Assignment 2 (A2) experienced the most change, with the proportion of perfectly identical submissions dropping sharply from 61.42% to 1.31%. In addition, the overall shape of the similarity distribution was restructured, suggesting a fundamental change in how students approached this mid-level task. For A2, we **reject** H_0, indicating a strong difference between years.

	Boundary	**Test**	**2022 (%)**	**2024 (%)**	***p* (raw)**	***p* (Holm)**
A1	1.0	χ^2	48.18	51.95	0.343	0.608
A2	1.0	χ^2	**61.42**	**1.31**	< 0.001	< 0.001
A3	1.0	Fisher	0.00	1.45	0.304	0.608
A1	0.0	Fisher	0.99	1.50	0.727	1.000
A2	0.0	-	0.00	0.00	-	-
A3	0.0	-	0.00	0.00	-	-

Table 4: Comparison of proportions at similarity scores 0.0 and 1.0 by assignment, 2022 and 2024.

Assignment 3 (A3) also showed a considerable change, reflected in a higher median similarity score from 0.416 to 0.692. Unlike A2, this change occurred primarily within the interior of the distribution, indicating greater convergence among student solutions without increases at the boundaries. For A3, we also **reject** H_0.

In contrast, Assignment 1 (A1) remained largely stable across years, showing no major shift in similarity patterns at the boundaries. However, a modest interior change was detected. This stability suggests that either generative AI tools were not heavily used for this introductory task or that the nature of the task limited variation in solutions. For A1, we **reject** H_0 in the interior-only analysis.

Take Away RQ1

Code similarity patterns shifted markedly between 2022 and 2024, coinciding with the rise of generative AI tools. The largest transformation occurred in A2, where identical submissions nearly disappeared, and the distribution itself was reshaped, signaling a fundamental change in how students tackled the task. A3 also changed substantially, with submissions converging toward higher similarity without extreme clustering. In contrast, A1 remained stable, showing only minor interior shifts. We therefore **reject** H_0 for A2 and A3, whereas for A1 we **fail to reject** H_0, noting only a minor interior redistribution.

5 Comparing Student–AI Similarity in 2024 (RQ2)

5.1 Approach

To evaluate whether students in the 2024 cohort showed similarity to AI-generated answers across assignments, we analyze repeated measures for each student across all assignments (A1, A2, and A3). For each submission, AI similarity was computed by comparing the student's code to fixed baseline solutions generated using ChatGPT and Gemini for the corresponding assignment, and defining a student's AI similarity as the element-wise maximum of the *GPT-based* and *Gemini-based* similarity scores. The analysis is limited to complete cases ($n = 222$), which are students who submitted all three assignments in 2024.

5.1.1 Primary analysis. The primary analysis treated assignments as a within-subject factor and evaluated whether the distributions of code similarity scores differed across assignments (A1, A2, and

A3). We apply the Friedman test as a nonparametric omnibus test for related samples [16, 37]. The effect size is assessed using Kendall's coefficient of concordance (W), which is calculated directly from the Friedman statistic χ^2 [19]. If the omnibus test shows a significant difference, we perform pairwise Wilcoxon signed-rank tests between assignments (A1–A2, A1–A3, A2-A3) to determine which pairs accounted for the overall difference [9, 43]. These pairwise tests are explicitly performed as two-sided tests, evaluating higher and lower similarity differences. The Pratt method was used to handle zero differences, so exact ties remained in the analysis but were assigned a zero rank [29]. Moreover, for each pairwise comparison, we calculated the paired Rank-Biserial correlation as a measure of effect size with bootstrap percentile 95%, and confidence intervals based on 10,000 resamples [10, 20].

5.1.2 Sensitivity analysis. Restricting complete-cases can reduce the sample size, and within-subject methods may lose power when values are highly concentrated at the boundary, where 1.0 indicates a perfect match; we performed a between-subjects sensitivity analysis [21].

Thus, we first apply a Kruskal-Wallis test as an omnibus comparison of A1, A2, and A3 [22]. Effect sizes were reported as epsilon-squared (ε^2), with 95% confidence intervals estimated using percentile bootstrapping based on 10,000 stratified resamples [10, 39]. When the Kruskal-Wallis test is significant, pairwise two-sided Mann-Whitney U tests are performed between assignments, with Cliff's delta and bootstrap percentile 95% confidence intervals as effect sizes [7, 10, 23].

5.1.3 Multiplicity control. To control the family-wise error rate, we used Holm's step-down adjustment at $\alpha = 0.05$ for the three pairwise Wilcoxon signed-rank tests from the primary repeated-measures analysis [18]. Sensitivity analysis is reported with raw two-sided p-values, and it is treated as descriptive, without multiplicity adjustment.

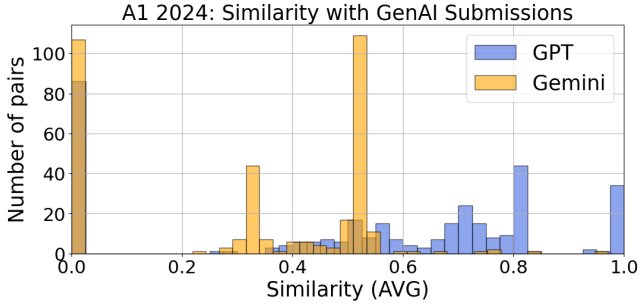

Figure 2: Similarity distribution with AI-generated baselines of A1 in 2024.

5.2 Descriptive Statistics

Figures 2 to 4 show the AI baseline similarity distributions between assignments, providing context for the analyses that follow. Table 5 summarizes the maximum similarity to AI-generated answers in 2024. In this context, a similarity score of 1.0 denotes maximal resemblance to at least one AI baseline, whereas 0.0 indicates no

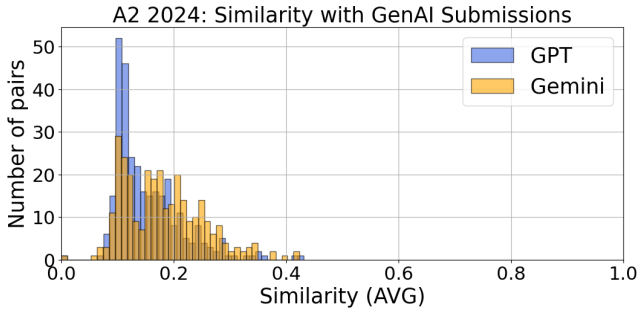

Figure 3: Similarity distribution with AI-generated baselines of A2 in 2024.

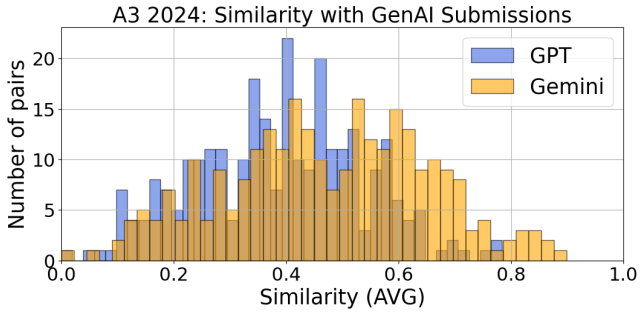

Figure 4: Similarity distribution with AI-generated baselines of A3 in 2024.

detectable resemblance under JPlag scoring. With these definitions established, we observe that A1 shows the highest central tendency, with Q2 (50%) = 0.643 on the raw 0-1 similarity scale, and substantial clustering at both boundaries, with 24.93% of scores at 0.0 and 9.91% at 1.0. In contrast, A2 and A3 had almost no boundary clustering, with proportions at 0.0 below 0.40% and no scores at 1.0.

Given these characteristics, we used rank-based, nonparametric tests and a between-subject sensitivity analysis to account for boundary effects.

	AVG	STD	Q1 (25%)	Q2 (50%)	Q3 (75%)	IQR	Proportion = 0 (%)	Proportion = 1 (%)
A1	0.530	0.342	0.267	0.643	0.800	0.533	24.93	9.91
A2	0.197	0.073	0.145	0.190	0.241	0.096	0.33	0.00
A3	0.492	0.175	0.373	0.507	0.626	0.253	0.36	0.00

Table 5: Descriptive statistics for students' maximum similarity to AI-generated baselines.

5.3 Primary Analysis

The Friedman omnibus test revealed a statistically significant difference in the distributions of maximum AI similarity scores across the three assignments ($\chi^2(2) = 167.177, p < 0.001, n = 222$). With $k = 3$ conditions (A1, A2, A3), the $df = 2$ reflects $k - 1$. The associated Kendall's $W = 0.377$ quantifies effect magnitude (0 = no systematic within-student differences; 1 = perfect ordering) and is typically interpreted as a moderate effect. Given the significant omnibus, we conducted three Wilcoxon signed-rank tests with Holm

adjustment, as shown in Table 6. A1-A2 and A2–A3 were significant (Holm $p < 0.001$) with large Rank-Biserial effects ($r = 0.814$ and $r = -0.979$, respectively), while A1–A3 was not significant (Holm $p = 0.163$; $r = 0.107$). Here, positive r values indicate that the scores in the first assignment tend to be higher than in the second. Collectively, these results imply the pattern A1 ≈ A3 > A2 for maximum similarity to AI baselines in 2024.

	Zero Differences (%)	W	p (raw)	p (Holm)	Rank-Biserial r [95% CI]
A1-A2	0.00	2305.000	< 0.001	< 0.001	0.814 [0.734, 0.878]
A1-A3	0.45	11040.000	0.163	0.163	0.107 [-0.047, 0.259]
A2-A3	0.00	262.000	< 0.001	< 0.001	-0.979 [-0.998, -0.950]

Table 6: Wilcoxon signed-rank pairwise comparisons of maximum similarity between assignments, 2024.

Sensitivity Analysis. To assess robustness, we performed a between-subject analysis, treating the three groups as independent. The Kruskal–Wallis test showed a statistically significant difference in maximum AI similarity across assignments ($H(2) = 269.960, p < 0.001$), with a large effect size ($\epsilon^2 = 0.294$ [0.250, 0.335]). Practically, this result confirms that A1 and A3 tend to show higher AI similarity than A2. Supporting this, the pairwise Mann-Whitney tests (Table 7) show that the similarity scores of A1>A2 and A3>A2 (both $p < 0.001$; positive Cliff's δ means that the first group tends to have higher scores), with medians A1= 0.643, A2= 0.190, A3= 0.507. The A1-A3 contrast is smaller (yet significant in the sensitivity analysis), yielding the overall pattern A1 ≳ A3 ≫ A2, which *aligns with* the repeated-measures results and suggests that the results are robust rather than being driven by the restriction to complete cases.

	U	p (raw)	Cliff's δ [95% CI]	Median A	Median B
A1-A2	76 363.500	< 0.001	0.499 [0.408, 0.590]	0.643	0.190
A1-A3	54 884.500	< 0.001	0.199 [0.106, 0.292]	0.643	0.507
A2-A3	5 638.000	< 0.001	-0.866 [-0.907, -0.819]	0.190	0.507

Table 7: Mann–Whitney U sensitivity analysis of maximum similarity between assignments, 2024.

The within-student distributions of maximum similarity to AI baselines *differ across assignments* in 2024 (Friedman $\chi^2(2) = 167.177$, $p < 0.001$, $W = 0.377$, $n = 222$); we therefore **reject** H_0 that A1, A2, and A3 share the same distribution. Holm-adjusted Wilcoxon signed-rank tests show **A1 > A2** and **A3 > A2** (both $p < 0.001$; large Rank-Biserial effects, $r = 0.814$ for A1-A2 and $r = -0.979$ for A2-A3), whereas the **A1-A3 contrast** is *not* significant (Holm $p = 0.163$; $r = 0.107$). Medians mirror this ordering (A1 = 0.643, A2 = 0.190, A3 = 0.507), indicating *A1 ≈ A3 > A2.*

Boundary clustering is most pronounced in A1 (24.93% at 0.0; 9.91% at 1.0) and minimal in A2 and A3 (proportion at 0.0 < 0.40%; none at 1.0), consistent with a higher resemblance in A1 and A3.

A between-subjects Mann-Whitney U sensitivity analysis with Cliff's δ yields the same qualitative ordering (A1 > A2, A3 > A2; see Table 7), indicating that the pattern is *robust* to the complete-cases restriction.

Take Away RQ2

The degree of similarity to AI baselines is highly assignment-dependent. Specifically, A1 and A3 show substantially higher resemblance than A2. A1 also exhibits stronger clustering at the boundaries; however, this pattern may reflect the constrained solution space of a relatively simple task with limited implementation variability, rather than stronger AI influence. In contrast, A2 shows neither pronounced boundary clustering nor high similarity to AI baselines. Accordingly, we **reject** H_0, as the distributions of maximum AI similarity differ across assignments in 2024.

6 Syntax Correctness Between Submissions in 2022 and 2024 (RQ3)

6.1 Approach

This research question examines whether the proportion of parsing errors differs between years for each assignment. For this goal, each submission was parsed using Python version 3.10.11. To evaluate the difference in the proportion of parsing errors between 2022 and 2024 for each assignment, we analyze whether a submission resulted in a parsing error (1 = failure, 0 = no failure). Each assignment was analyzed separately. To preserve the independence of the samples, the students belong to a single cohort (either 2022 or 2024). All hypothesis tests were conducted using two-sided inference with a family-wise significance level set at $\alpha = 0.05$.

6.1.1 Primary analysis. We create a 2×2 contingency table for each assignment, with the years (2022 and 2024) as rows and the parsing outcomes as columns. We used Pearson's χ^2 independence test without Yates continuity correction when all cell counts [2] are at least five [1, 24, 27]. If this condition was not satisfied or zero counts were present, we used Fisher's exact test [1, 12, 13]. For Pearson's χ^2 test, we check the *expected* cell counts under the null hypothesis of independence,

$$E_{ij} = \frac{(\text{row}_i \text{ total}) \times (\text{column}_j \text{ total})}{\text{grand total}},$$

and apply the χ^2 test only when all $E_{ij} \geq 5$; otherwise (including zeros or sparse cells) we use Fisher's exact test.

Two-sided p-values were reported for both Pearson's χ^2 and Fisher's exact tests. The Holm step-down procedure was then applied in all three assignments to control the family-wise error rate [18].

Multiplicity control. To account for multiple tests across assignments, we applied Holm's step-down procedure to the three *primary* p-values (one per assignment from Pearson's χ^2 or Fisher's exact test), controlling the family-wise error rate at $\alpha = 0.05$ [18].

6.2 Descriptive Statistics

Table 8 provides the descriptive statistics for syntax failure rates, showing the counts and percentages of failures by year and assignment. Rates are the proportion of submissions with parsing errors (1 = failure, 0 = no failure). We observe that the failure rates were *low in Assignment 1* for both years (1.32% in 2022 and 2.10% in 2024).

[2] Observed frequencies in each of the four cells of the 2×2 table (year × parsing outcome)

Assignment 2 showed the *highest overall rates*, although there was a decrease from 15.75% in 2022 to 10.78% in 2024. Assignment 3 showed intermediate rates, with a slight decline from 8.39% to 6.55% over the years.

6.3 Primary Analysis

For each assignment and year, syntax failure rates are computed using the number of submissions included in the final dataset after preprocessing, as summarized in Table 1.

A1. Syntax failure rates were low in both years (2022: 4/303 = 1.32%; 2024: 7/333 = 2.10%). The difference was not significant ($\chi^2 = 0.571$, $p = 0.450$, Holm $p = 0.900$), indicating no detectable change in A1.

A2. Failure rates decreased from 40/254 = 15.75% (2022) to 33/306 = 10.78% (2024). Although suggestive, this difference did not reach significance after multiplicity control ($\chi^2 = 3.017$, $p = 0.082$, Holm $p = 0.247$), so we fail to reject H_0 for A2.

A3. Rates were 12/143 = 8.39% (2022) versus 18/275 = 6.55% (2024), a small decline that was not statistically significant ($\chi^2 = 0.481$, $p = 0.488$, Holm $p = 0.900$).

Taken together, none of the assignments showed a statistically significant change from year-to-year in syntactic error rates after the Holm adjustment; therefore, we **fail to reject** H_0 for A1, A2, or A3.

	Number of Failures		Failure Rate (%)		Test	Statistic	p (raw)	p Holm
	2022	2024	2022	2024				
A1	4	7	1.32	2.10		0.571	0.450	0.900
A2	**40**	**33**	**15.75**	**10.78**	χ^2 (df = 1)	3.017	0.082	0.247
A3	12	18	8.39	6.55		0.481	0.488	0.900

Table 8: Descriptive statistics for parsing errors and primary tests by assignment.

Take Away RQ3

Syntax correctness rates were stable over time. Although raw failure rates in A2 and A3 decreased (A2: 15.75% → 10.78%; A3: 8.39% → 6.55%), A1 remained low (2022: 1.32%, 2024: 2.10%). None of these differences were statistically significant after Holm adjustment of the χ^2 tests. We therefore **fail to reject** H_0 for A1, A2, or A3. In short, while similarity patterns changed across years, there is no statistically significant evidence of a cohort-level change in basic syntactic correctness.

7 Discussion

7.1 Shifts in Code Similarity Patterns Across Assignments

Our findings reveal a shift in how students approach programming assignments after the widespread availability of generative AI tools. The most significant change occurred in Assignment 2 (A2). In 2022, 61.42% of submissions had a maximum similarity of 1.0, compared to only 1.31% in 2024, while similarity scores of 0.0 remained rare in both years. At the same time, the distribution of scores within the interior range (between 0.0 and 1.0) became more concentrated in 2024, with the interquartile range decreasing from 0.656 in 2022 to 0.156 in 2024. Together, these results indicate a sharp reduction in perfect matches and a shift toward a tighter clustering of similarity among non-identical submissions, pointing to a transition away from traditional forms of plagiarism (e.g., code sharing between classmates) toward more diverse and individualized solutions, likely influenced by the use of generative AI.

For Assignment 3 (A3), the most significant change was an increase in the median similarity score, rising from 0.416 in 2022 to 0.692 in 2024. Importantly, this change was not accompanied by an increase in perfect 1.0 matches at the boundary, indicating that the year-to-year difference occurred primarily within the interior of the distribution. This could suggest that while students' submissions became more alike overall, they were not identical. Instead, students appeared to converge on similar solution strategies without directly copying one another's code.

Assignment 1 (A1) remained largely stable between 2022 and 2024. Boundary proportions at 0.0 and 1.0 showed no meaningful differences, and only a modest redistribution was detected within the interior scores. This stability could suggest that generative AI tools had limited influence on this introductory task, likely because the problem structure already constrained the range of possible correct solutions.

7.2 Task Complexity and Alignment with AI-Generated Solutions

When examining resemblance to fixed AI baselines within the 2024 cohort, we observed distinct patterns across assignments. Both A1 and A3 exhibited high similarity to AI-generated baselines, while A2 was notably lower. This suggests that AI tools are well suited to simple tasks, where their outputs closely match what students produce independently, and to complex, open-ended tasks, where students may adopt AI-generated guidance to structure their solutions. By contrast, mid-level tasks such as A2 appear to encourage more diverse approaches, resulting in lower direct AI resemblance.

7.3 No Significant Change in Syntax Error

Despite these shifts in similarity patterns, syntax correctness did not differ significantly between 2022 and 2024. Although raw failure rates decreased slightly in A2 and A3, these differences were not statistically significant after correcting for multiple comparisons. This finding supports the interpretation that while generative AI may influence the structure and originality of code, it has not yet produced measurable improvements or declines in basic syntactic accuracy across the cohort.

Summary

Overall, these results highlight a change in the nature of academic integrity risks. Where earlier concerns centered on student-to-student duplication, our data show a move toward student-to-AI convergence, with solutions increasingly resembling those generated by automated systems. This transition has important implications for how educators interpret similarity metrics and assess originality in the age of generative AI.

8 Implications

These findings have several implications for educators, institutions, and policy makers as they adapt to the widespread use of generative AI in higher education.

From Peer-to-Peer Duplication to Student-AI Convergence. The marked decrease in perfect peer matches in A2 along with the increase in similarity to AI-generated baselines highlights a fundamental shift in the meaning of similarity scores. Traditional plagiarism detection tools such as JPlag remain valuable for identifying direct copying among students, but they are no longer sufficient for detecting academic integrity violations when the source of similarity is an external AI system. This calls for a rethinking of integrity frameworks and the development of new detection strategies that account for student-AI convergence rather than peer-to-peer duplication.

Rethink Programming Assignments in the Age of AI. The strong alignment of A1 and A3 with AI baselines suggests that the design of programming assignments must evolve. Predictable, straightforward tasks (A1) are particularly vulnerable to full automation by generative AI, raising questions about their effectiveness as authentic assessments of student learning. At the same time, the convergence seen in open-ended tasks (A3) indicates that AI tools may influence how students structure their solutions even when creativity is encouraged. These patterns imply that future assessments should emphasize problem-solving processes, reflection, and reasoning, rather than final code alone, to better differentiate between AI-generated and human written work.

Code Production vs. Conceptual Mastery. The lack of significant improvement in syntax correctness over the years underscores that AI does not necessarily improve foundational programming skills such as debugging or error identification. This finding suggests that, while generative AI can accelerate code production, it does not guarantee deeper learning or mastery of core concepts. Educators must therefore integrate explicit instructional strategies to support debugging, comprehension, and algorithmic thinking, rather than assuming that AI assistance will address these persistent challenges.

GenAI Policies. At the institutional level, the transition from student-to-student plagiarism to student-AI convergence highlights the need to update policies and guidelines. Institutions need clear frameworks that define acceptable and unacceptable uses of generative AI, coupled with pedagogical practices that encourage responsible engagement with these tools. Without such guidance, both students and instructors face uncertainty about ethical boundaries, and assessments risk losing their ability to meaningfully measure individual achievement. Our results provide a data-driven foundation for these conversations and point to the need for proactive course and curriculum redesign in the era of AI-driven programming support.

Assignments in Controlled Environments. To mitigate the usage of generative AI tools in the introductory programming course, assignments could be conducted in a controlled environment.

This would guarantee that students would have limited access to materials and tools available to develop assignments. Even though this is an interesting direction, additional considerations are needed. For example, physical facilities are limited; this puts extra pressure and stress on teaching staff and students, additional overhead in terms of personnel to supervise those sessions, and the length and scope of the assignments must be more concise given the time constraints.

9 Limitations

Internal validity. In A3, the 2022 submissions had inconsistent file-naming conventions. Thus, students who submitted in 2022 and 2024 were not detected as duplicates, potentially inflating similarity levels for A3.
In addition, despite maintaining the same instructor and assignments, uncontrolled variables may introduce bias. These include differences in cohort preparation, motivation, and informal support, as well as logistical factors such as class size, timing, or grade stakes. However, JPlag token thresholds and permutation-based p-values reduce trivial matches and ties, but cannot eliminate underlying design biases.

External validity. We studied only three assignments from a single introductory programming course at a single institution. Although these assignments represent an important assessment method, they do not capture the diversity of assessment formats (e.g., larger projects or written exams). Therefore, the generalization of the results to other contexts, courses, or programming languages is limited. Although the period (2022-2024) coincides with rapid model upgrades, other timelines may not replicate the magnitudes observed in this study.

Construct validity. The influence of AI was assessed through similarity to fixed ChatGPT and Gemini baselines generated under specific prompts and model versions. However, this approach offers only a single representation of AI assistance. Since different prompts or model versions could produce varying outputs, our similarity scores should be interpreted as reflecting resemblance to these specific baselines rather than to the full range of solutions that GenAI can produce.

Separately, parsing errors focus on syntax only; they do not capture logic, semantics, runtime, or code quality errors. Additionally, this metric may be insensitive to AI usage among struggling students. For instance, if a student lacks the competency to interpret and resolve basic error messages, they are likely to have high error rates even when provided with AI support.

10 Conclusion and Future Work

Conclusion. This study examined the impact of generative AI on introductory programming assignments by comparing two cohorts of students (2022 and 2024) who completed the same three Python assignments under the same instructor and course structure. We used three complementary measures: student–student similarity measured with JPlag (RQ1-Section 4), resemblance to AI-generated solutions compared against baseline solutions (RQ2-Section 5), and syntax correctness assessed via parsing errors (RQ3-Section 6).

Across the three assignments, the most significant change occurred in Assignment 2 (A2). Perfect peer-to-peer matches dropped sharply from 2022 to 2024, and the non-boundary scores (those not exactly 0.0 or 1.0) moved from a wide spread to a narrow cluster.

Assignment 3 (A3) also changed, but only in the middle range of scores (excluding exact 0.0 and 1.0): students' submissions were generally similar, without an increase in perfectly identical matches. In contrast, Assignment 1 (A1) remained stable across years.

These patterns suggest a move away from traditional forms of plagiarism (e.g., code sharing between students) toward more diverse, individualized solutions, likely influenced by generative AI tools. Within the 2024 cohort, the similarity to AI-generated baselines was highest for A1 and A3, and considerably lower for A2. This might indicate that both simple and complex tasks are more susceptible to AI influence, while mid-level tasks such as A2 encourage a wider range of approaches. Despite these changes, syntax error rates did not differ significantly between years, indicating that AI use has not yet led to measurable improvements or declines in basic syntactic accuracy.

In general, these findings might imply a change in academic integrity concerns. Earlier challenges focused on student-to-student duplication; however, the current challenge has become differentiating between human-written and AI-influenced code. This change might have important implications in how educators design and create assignments, evaluate similarity scores, and maintain integrity in programming education.

Future work. Future research should expand the scope of this study by examining additional courses, programming languages, and institutions. Replicating the analysis across diverse contexts will help determine whether the observed patterns can be generalized or are context-specific. Tracking future cohorts over time could also reveal how these trends evolve as GenAI tools continue to advance.

Another important direction is strengthening causal inference. Because this study was observational, it cannot establish definitive causal relationships between GenAI use and the changes observed in code similarity patterns. Future work could employ experimental designs, such as randomized assignment versions, or natural experiments such as mid-course policy changes, to better isolate the effects of AI on student behavior and learning outcomes.

Future work should also develop broader measures of AI influence. While this study used similarity to fixed baselines, incorporating multiple AI models or prompts, as well as runtime correctness checks, would provide a richer and more robust understanding of AI involvement in student work.

Finally, linking similarity patterns with learning outcomes is essential. Examining debugging skills, conceptual understanding, and transfer performance would clarify whether AI resemblance reflects meaningful learning or only surface-level code generation. This could be further strengthened by combining similarity analyses with behavioral or survey data, such as self-reported AI use, as well as individual IDE interaction logs capturing development patterns (e.g., incremental editing or large code insertions) to better interpret the processes underlying observed similarity trends.

Acknowledgments

The authors acknowledge the use of OpenAI's ChatGPT (version GPT-5) for assistance in structuring and formatting tables. The generated tables were manually checked, revised, and edited by the authors.

References

[1] Alan Agresti. 2007. *An Introduction to Categorical Data Analysis* (2nd ed.). John Wiley & Sons, Hoboken, NJ.

[2] Yunjo An, Ji Hyun Yu, and Shadarra James. 2025. Investigating the higher education institutions' guidelines and policies regarding the use of generative AI in teaching, learning, research, and administration. *International Journal of Educational Technology in Higher Education* 22, 1 (2 2025). doi:10.1186/s41239-025-00507-3

[3] Anysphere. 2025. Cursor (AI-powered IDE). https://www.cursor.com/. Latest release version 1.0, released 2025-06-04; accessed 2025-09-10.

[4] S. Bashir. 2025. Using pseudo-AI submissions for detecting AI-generated code. *Frontiers in Computer Science* 7 (2025). doi:10.3389/fcomp.2025.1549761

[5] K. Bittle and O. El-Gayar. 2025. Generative AI and Academic Integrity in Higher Education: A Systematic Review and Research Agenda. *Information* 16, 4 (2025), 296. doi:10.3390/info16040296

[6] Cecilia Ka Yuk Chan and Wenjie Hu. 2023. Students' voices on generative AI: perceptions, benefits, and challenges in higher education. *International Journal of Educational Technology in Higher Education* 20, 1 (7 2023). doi:10.1186/s41239-023-00411-8

[7] Nora Cliff. 1993. Dominance Statistics: Ordinal Analyses to Answer Ordinal Questions. *Psychological Bulletin* 114, 3 (1993), 494–509. doi:10.1037/0033-2909.114.3.494

[8] M. Crane and W. Dempsey. 2018. Reproducibility and Variability of Published Results. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics. https://aclanthology.org/Q18-1018.pdf

[9] Janez Demsar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine Learning Research* 7 (2006), 1–30. http://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf

[10] Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, New York, NY. doi:10.1201/9780429246593

[11] Mark D. Ernst. 2004. Permutation Methods: A Basis for Exact Inference. *Statist. Sci.* 19, 4 (2004), 676–685. doi:10.1214/088342304000000396

[12] R.A. Fisher. 1922. On the Interpretation of χ^2 from Contingency Tables, and the Calculation of P. *Journal of the Royal Statistical Society* 85, 1 (1922), 87–94. https://www.jstor.org/stable/2340521

[13] Ronald A. Fisher. 1935. *The Design of Experiments*. Oliver and Boyd, Edinburgh and London.

[14] Windsurf (formerly Codeium). 2025. Windsurf Editor (AI-native IDE with Cascade agent). https://windsurf.com/editor. Used by over 1 million developers in four months; acquired by Cognition AI in 2025; access date 2025-09-10.

[15] S. L. France. 2024. Navigating software development in the ChatGPT and GitHub Copilot era. *Business Horizons* (2024). doi:10.1016/j.bushor.2024.05.009

[16] Milton Friedman. 1937. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *J. Amer. Statist. Assoc.* 32, 200 (1937), 675–701. doi:10.1080/01621459.1937.10503522

[17] Philip I. Good. 2000. *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypotheses*. Springer, New York, NY. doi:10.1007/978-1-4757-2346-5

[18] Sture Holm. 1979. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6, 2 (1979), 65–70. https://www.jstor.org/stable/4615733

[19] Maurice G. Kendall and Jean Dickinson Gibbons. 1990. *Rank Correlation Methods* (5th ed.). Edward Arnold, London. This book covers Kendall's W and its relation to Friedman test.

[20] David S. Kerby. 2014. The simple difference formula: An approach to teaching nonparametric correlation. *Teaching of Psychology* 41, 1 (2014), 18–22. https://journals.sagepub.com/doi/full/10.2466/11.it.3.1

[21] Gideon Keren. 1993. Between or within subjects design: A methodological dilemma. *Quality & Quantity* 27, 3 (1993), 235–249.

[22] William H Kruskal and W Allen Wallis. 1952. Use of ranks in one-criterion variance analysis. *J. Amer. Statist. Assoc.* 47, 260 (1952), 583–621.

[23] Henry B. Mann and Donald R. Whitney. 1947. On a Test of Whether One of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (March 1947), 50–60. doi:10.1214/aoms/1177730491

[24] M.L. McHugh. 2013. The Chi-square test of independence. *Biochemia Medica* 23, 2 (2013), 143–149. doi:10.11613/BM.2013.018

[25] R. P. Medeiros, G. L. Ramalho, and T. P. Falcao. 2018. A Systematic Literature Review on Teaching and Learning Introductory Programming in Higher Education. *IEEE Transactions on Education* 62, 2 (2018), 77–90. doi:10.1109/te.2018.2864133

[26] Uday Mittal, Siva Sai, Vinay Chamola, and Devika Sangwan. 2024. A Comprehensive Review on Generative AI for Education. *IEEE Access* 12 (2024), 142733–142759. doi:10.1109/ACCESS.2024.3468368

[27] K. Pearson. 1900. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50, 302 (1900), 157–175.

[28] Hung Pham, Huyen Ha, van Tong, Dung Hoang, Duc Tran, and Tuyen Ngoc Le. 2024. MAGECODE: Machine-Generated Code Detection Method Using Large Language Models. *IEEE Access* 12 (2024), 190186–190202. doi:10.1109/ACCESS.2024.3509987

[29] JW Pratt. 1959. Remarks on zeros and ties in the Wilcoxon signed rank procedures. *J. Amer. Statist. Assoc.* 54, 287 (1959), 655–667. doi:10.1080/01621459.1959.10501526

[30] Lutz Prechelt, Guido Malpohl, and Michael Philippsen. 2002. Finding Plagiarisms among a Set of Programs with JPlag. *Zenodo (CERN European Organization for Nuclear Research)* (11 2002). doi:10.3217/jucs-008-11-1016

[31] Junaid Qadir. 2023. Engineering Education in the Era of ChatGPT: Promise and Pitfalls of Generative AI for Education. In *2023 IEEE Global Engineering Education Conference (EDUCON)*. 1–9. doi:10.1109/EDUCON54358.2023.10125121

[32] Yizhou Qian and James Lehman. 2017. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ.* 18, 1, Article 1 (Oct. 2017), 24 pages. doi:10.1145/3077618

[33] Frank W. Scholz and Michael A. Stephens. 1987. K-Sample Anderson–Darling Tests. *J. Amer. Statist. Assoc.* 82, 399 (Sept. 1987), 918–924. doi:10.2307/2288805

[34] M. E. S. Simaremare, C. Pardede, I. N. I. Tampubolon, D. A. Simangunsong, and P. E. Manurung. 2024. The penetration of Generative AI in Higher Education: a survey. In *2022 IEEE Integrated STEM Education Conference (ISEC)*. 1–5. doi:10.1109/isec61299.2024.10664825

[35] Adele Smolansky, Andrew Cram, Corina Raduescu, Sandris Zeivots, Elaine Huber, and Rene F. Kizilcec. 2023. Educator and Student Perspectives on the Impact of Generative AI on Assessments in Higher Education. In *Proceedings of the Tenth ACM Conference on Learning @ Scale* (Copenhagen, Denmark) *(L@S '23)*. Association for Computing Machinery, New York, NY, USA, 378–382. doi:10.1145/3573051.3596191

[36] K. Sol, S. Sok, and K. Heng. 2025. Rethinking Assessment in Higher Education in the Age of Generative AI. In *Encyclopedia of Educational Innovation*, M. A. Peters and R. Heraud (Eds.). Springer, Singapore. doi:10.1007/978-981-13-2262-4_327-2

[37] Laerd Statistics. 2023. Friedman Test in SPSS Statistics - How to Run the Procedure and Interpret the Output. https://statistics.laerd.com/spss-tutorials/friedman-test-using-spss-statistics.php. Accessed 7 September 2025.

[38] Tobias Steffening, Philipp Groß, et al. 2024. *JPlag: Source Code Plagiarism Detection Tool, version 5.1.0*. JPlag Team. https://github.com/jplag/JPlag/releases/tag/v5.1.0 Accessed: 2024-12-15.

[39] Maciej Tomczak and Ewa Tomczak. 2014. The need to report effect size estimates revisited. An overview of some recommended measures of effect size. *Trends in Sport Sciences* 21, 1 (2014), 19–25. http://www.wbc.poznan.pl/Content/325867/5_Trends_Vol21_2014_%20no1_20.pdf

[40] UNESCO. 2023. UNESCO Survey: Less than 10% of Schools and Universities have Formal Guidance on AI. https://www.unesco.org/en/articles/unesco-survey-less-10-schools-and-universities-have-formal-guidance-ai. Accessed: 2025-09-01.

[41] UNESCO. 2025. UNESCO Survey: Two-Thirds of Higher Education Institutions Have or Are Developing Guidance on AI Use. https://www.unesco.org/en/articles/unesco-survey-two-thirds-higher-education-institutions-have-or-are-developing-guidance-ai-use. Accessed: 2025-09-01.

[42] Ben Ward, Deepshikha Bhati, Fnu Neha, and Angela Guercio. 2025. Analyzing the Impact of AI Tools on Student Study Habits and Academic Performance. In *2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC)*. 00434–00440. doi:10.1109/CCWC62904.2025.10903692

[43] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (1945), 80–83. doi:10.2307/3001968

[44] Ying Xie, Shaoen Wu, and Sumit Chakravarty. 2023. AI meets AI: Artificial Intelligence and Academic Integrity - A Survey on Mitigating AI-Assisted Cheating in Computing Education. In *Proceedings of the 24th Annual Conference on Information Technology Education* (Marietta, GA, USA) *(SIGITE '23)*. Association for Computing Machinery, New York, NY, USA, 79–83. doi:10.1145/3585059.3611449

[45] Abdullahi Yusuf, Nasrin Pervin, and Marcos Román-González. 2024. Generative AI and the future of higher education: a threat to academic integrity or reformation? Evidence from multicultural perspectives. *International Journal of Educational Technology in Higher Education* 21, 1 (3 2024). doi:10.1186/s41239-024-00453-6

A Topics Per Assignment

Assignment	Topics Covered
Assignment 1 (A1)	Fundamental mathematical operations, including addition, subtraction, multiplication, and division.
Assignment 2 (A2)	Use of global variables within functions; defining and calling functions; use of nested lists and dictionaries for variable and data organization; reading and analyzing file content; handling exceptions to prevent runtime errors; and use of built-in Python methods such as `replace()`, `strip()`, and `isdigit()`.
Assignment 3 (A3)	Object-oriented programming concepts including class creation and inheritance; implementation of functions and methods; use of nested lists and dictionaries; use of built-in functions such as `map()`, `join()`, and `max()`; reading and analyzing file content; and importing external libraries such as `statistics` for numerical computations.

Table 9: Topics covered by each assignment.