Generative artificial intelligence-supported programming education: Effects on learning performance, self-efficacy and processes

Siran Li
School of Education, Soochow University, Suzhou, Jiangsu, China

Jiangyue Liu
Center for Chinese Urbanization Studies, Soochow University, Jiangsu Province, China
School of Education, Soochow University, Suzhou, Jiangsu, China
Department of Computer Science, Humboldt-Universität zu Berlin, Berlin, Germany

Qianyan Dong
School of Education, Soochow University, Suzhou, Jiangsu, China

Recent advancements in generative artificial intelligence (GenAI) have drawn significant attention from educators and researchers. However, its effects on learners' programming performance, self-efficacy and learning processes remain inconclusive, while the mechanisms underlying its efficiency-enhancing potential are underexplored. This study addresses these gaps through a quasi-experiment comparing an experimental group using GenAI for self-directed programming learning with a control group relying on alternative tools. Additionally, the experimental group was divided into high- and low-performance subgroups to examine the relationship between learning behaviour patterns and academic outcomes using process mining techniques. The findings reveal that (a) GenAI demonstrates strong potential to enhance learning outcomes and self-efficacy but negatively affects long-term knowledge transfer; (b) excessive reliance on GenAI and cognitive outsourcing impede effective knowledge acquisition; (c) high-performing learners exhibit greater epistemic agency, actively critiquing and engaging with AI-generated content to construct knowledge proactively. This study underscores the risks of over-reliance on GenAI and the detrimental effects of cognitive offloading, highlighting the critical role of cognitive engagement and epistemic agency in fostering hybrid intelligence. It provides empirical and theoretical insights to inform the design of GenAI-supported programming education strategies and interventions.

Implications for practice or policy:

- Instructors can enhance programming self-efficacy by integrating GenAI tools like ChatGPT into self-learning activities, particularly for reinforcing academic performance.
- Course leaders should emphasise GenAI use in programming courses to support student engagement, though they should also prepare students for problem-solving without external resources.
- Educational institutions may consider developing guidelines for balanced GenAI usage to maximise learning benefits while addressing potential limitations in problem-solving skills.

Keywords: generative artificial iIntelligence (GenAI), programming, learning performance, learning behaviour, learning process mining

Introduction

In November 2022, OpenAI's release of ChatGPT 3.5 brought significant attention to generative artificial intelligence (GenAI). These applications utilise pre-trained large language models, which are further refined through extensive training on vast data sets, enabling them to generate coherent and human-like responses tailored to user queries (Sabzalieva & Valentini, 2023).

GenAI's vast information repository, coupled with its advanced natural language understanding and generation capabilities, holds transformative potential for education. It offers a unique opportunity to integrate machine intelligence and enhance educational quality. The concept of hybrid intelligence suggests that humans and AI complement each other, forming collective intelligence that enables achievements beyond either alone (Dellermann et al., 2019; Weber et al., 2024). Researchers are increasingly applying GenAI across various teaching stages to optimise instructional processes and effectiveness through hybrid intelligence. In instructional preparation, GenAI helps teachers design materials more effectively (Mhlanga, 2023). During instruction, it serves as a teaching assistant, supporting personalised learning (Eysenbach, 2023; Wang & Chen, 2023). In assessments, GenAI aids personalised evaluations and provides timely feedback (Bernius et al., 2022; Chen et al., 2025; Moore et al., 2022). GenAI also finds applications in fields such as foreign language learning (Fan et al., 2024; Tai & Chen, 2024), law (Weber et al., 2024), physics (Ding et al., 2023), engineering (Qadir, 2023), media (Pavlik, 2023) and business (Alshater, 2022).

Learning abstract computer languages is challenging due to the complexity of programming and the difficulty of conceptualising, designing, implementing and evaluating solutions (Medeiros et al., 2019). This is especially true in introductory university programming courses, where around one third of students globally fail, leading to high dropout rates (Bennedsen & Caspersen, 2007; Watson & Li, 2014). Various teaching approaches, such as block-based programming, virtual reality environments, project-based learning, pair programming and game-based learning, have been developed to address these challenges. However, block-based programming is rarely used in higher education due to its limited relevance to real-world job markets. Virtual reality-based learning faces technical and scalability issues, while other methods often struggle with interpersonal challenges such as differences in prior programming experience (Satratzemi et al., 2018), skill levels (Dou & He, 2010), gender dynamics (Kuttal et al., 2019) and personality traits (Tsompanoudi et al., 2019).

GenAI offers strong potential for human-machine collaboration in programming, overcoming interpersonal challenges and providing personalised support to novice learners through hybrid intelligence. As this technology rapidly evolves, understanding its educational impact in programming becomes crucial. Although studies have explored GenAI's applications in education, research on its effects on learning outcomes, self-efficacy and processes in programming education remains limited in depth and breadth. By investigating these aspects, this study not only advances the understanding of GenAI's role in programming education but also provides actionable insights for designing evidence-based strategies to harness hybrid intelligence in fostering effective and equitable learning environments.

Literature review

The impact of GenAI on programming learning performance

Academic performance quantifies students' cognitive gains and serves as an objective measure of intellectual achievement. Recent studies have increasingly explored GenAI's impact on academic performance, particularly in programming education. Research has shown GenAI's potential to enhance learning outcomes. For instance, Sun et al. (2024) conducted a quasi-experiment demonstrating that GenAI-supported programming learning improves task quality. Similarly, Gong et al. (2024) validated GenAI's impact through post-study programming assessments, showing that students in the GenAI-assisted group outperformed those in the control group. These studies highlight both the immediate and long-term benefits of GenAI in programming education, indicating its potential to enhance learning outcomes and facilitate knowledge transfer.

However, not all findings are consistent. Jayagopal et al. (2022) applied GenAI tools in undergraduate programming courses, enabling students to pair with GenAI for task completion. Although their quasi-experiment revealed faster task completion, it found no significant differences in learning performance between GenAI-supported and control groups, suggesting that speed enhancements alone do not equate to deeper learning gains or justify broad adoption.

These mixed results underscore a critical gap in understanding GenAI's true contribution to learning performance, particularly its dual impact on immediate task success and long-term knowledge retention. This study, therefore, seeks to address the following research question:

- RQ1: What are the effects of GenAI on the quality of programming outputs and knowledge transfer?

The role of GenAI in enhancing programming self-efficacy

Self-efficacy refers to an individual's belief in their ability to successfully complete a task or achieve a goal (Bandura, 1982). Research has shown that self-efficacy plays a crucial role in programming education, directly influencing academic performance and learning processes. For example, Kanaparan et al. (2017) surveyed 433 introductory programming students and found that programming self-efficacy positively impacts effort and persistence, which in turn enhances academic achievement. Similarly, Lin (2016) identified a significant positive correlation between self-efficacy and understanding of fundamental programming concepts, indicating that learners with higher self-efficacy develop deeper conceptual understanding, while those with lower self-efficacy face greater difficulties. Therefore, fostering programming self-efficacy is key to improving learning outcomes and enhancing programming education quality.

Despite its significance, few studies have explored the impact of GenAI on self-efficacy in programming. Yilmaz and Yilmaz (2023) found that GenAI tools significantly boosted learners' programming self-efficacy. Additionally, studies by Liang et al. (2023) and Zhang and. Xu (2025) confirmed that GenAI usage enhances self-efficacy across various subject areas.

Given the importance of self-efficacy and the limited research on GenAI's effects, this study seeks to address the following question:

- RQ2: How does GenAI support influence learners' programming self-efficacy?

GenAI-supported programming learning processes

Recent research is increasingly combining qualitative and quantitative methods to explore the "black box" of human-machine collaboration, offering micro-level insights into GenAI's impact on learning and informing the optimisation of GenAI-supported educational processes. For instance, Barke et al. (2023) employed grounded theory to examine programming tasks performed with GenAI tools, identifying two interaction modes: acceleration mode, where GenAI is used to expedite goal-oriented tasks, and exploration mode, where GenAI aids in exploring uncertain solutions. Sun et al. (2024) utilised lag sequence and cognitive network analyses to compare the learning behaviours of GenAI-supported and control groups, revealing that GenAI users exhibited more frequent debugging and code optimisation, demonstrating a greater ability to leverage GenAI for programming improvements.

However, critical gaps persist. Studies have not established a clear link between learning behaviours and academic performance. Additionally, although basic interaction patterns have been identified, strategies for optimising human-machine collaboration to enhance learning outcomes remain underexplored. This study, therefore, poses the following research questions:

- RQ3: How does GenAI influence learners' programming behaviours?
- RQ4: How do different behaviour patterns in GenAI-supported learning affect students' academic performance?

By addressing these questions, this study aims to uncover the mechanisms of human-machine collaboration, offering insights for adapting programming education in the age of widespread GenAI adoption.

Methodology

We conducted a quasi-experiment by initially creating a self-learning environment infused with GenAI in university classrooms. Subsequently, we collected data on students' learning processes, including learning behaviour and learning products. Before the experiment began, the participants were fully informed about the purpose of the study and were explicitly made aware that they could withdraw from the experiment at any time without any conditions. The research project was also approved by our institution.

Participants

The participants were 55 undergraduate Physics Education students (26 males, 29 females) from a public university in China, with an average age of 21. All had completed a foundational Python programming course but had no prior experience with the Python Turtle library.

Research procedure

As shown in Figure 1, prior to the experiment, all students completed a pretest on programming self-efficacy and a comprehensive test on fundamental Python knowledge. Based on the results of the Python knowledge test, students were divided into two groups with similar proficiency levels. The experimental group had 27 participants, and the control group had 28, with no significant differences in age, gender or programming self-efficacy between the groups.

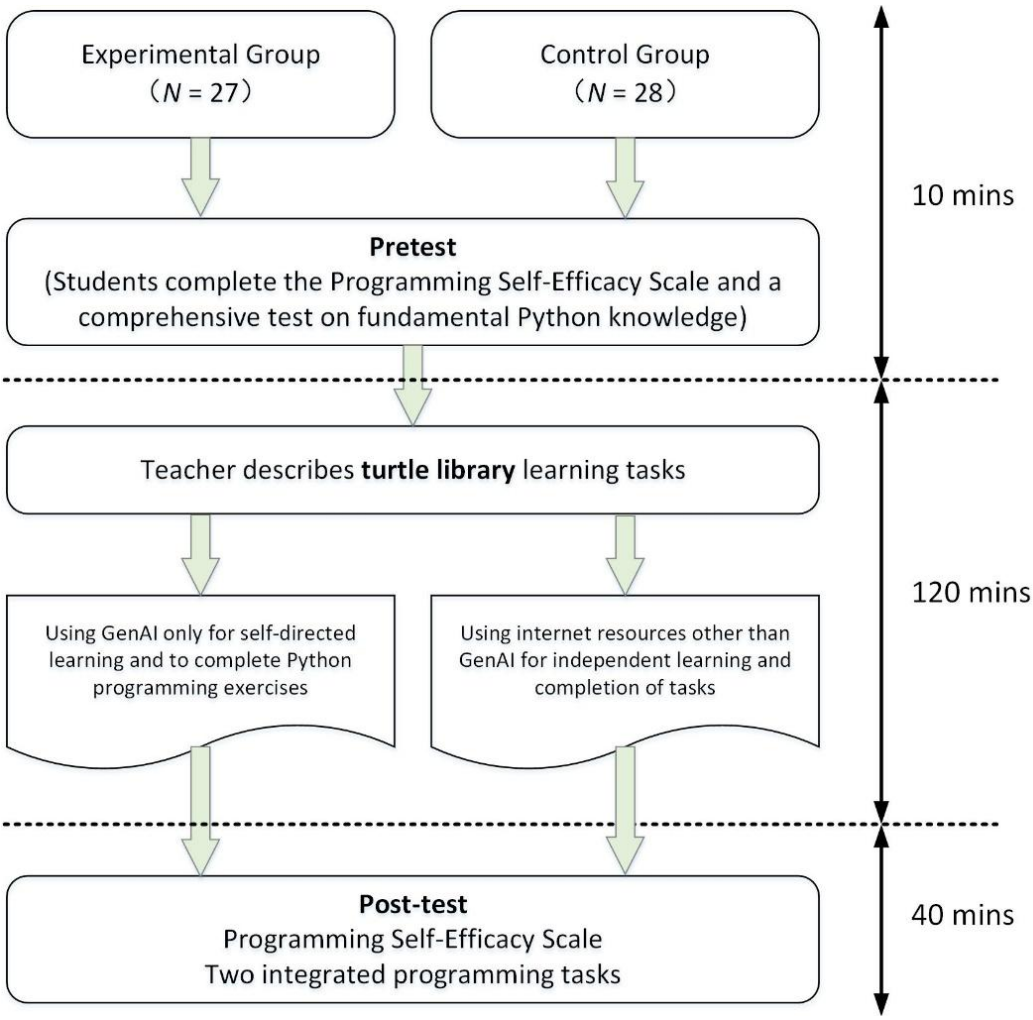

Figure 1. Research procedure

During the experiment, both groups received the same task sheet with self-learning tasks: (a) understanding the Turtle library, (b) creating a mind map for the Turtle library and (c) completing five programming exercises. The tools allowed for self-learning differed between groups: the experimental group used GenAI exclusively, while the control group had access to other Internet resources.

The GenAI tool, based on Sun et al. (2024), was the ChatGPT-Next-Web platform (Figure 2), utilising the GPT-3.5-turbo model. Students interacted with ChatGPT for feedback by submitting questions. For the experiment, we added a user login page, retaining only basic conversation functions, ensuring a clear, simple and secure environment.

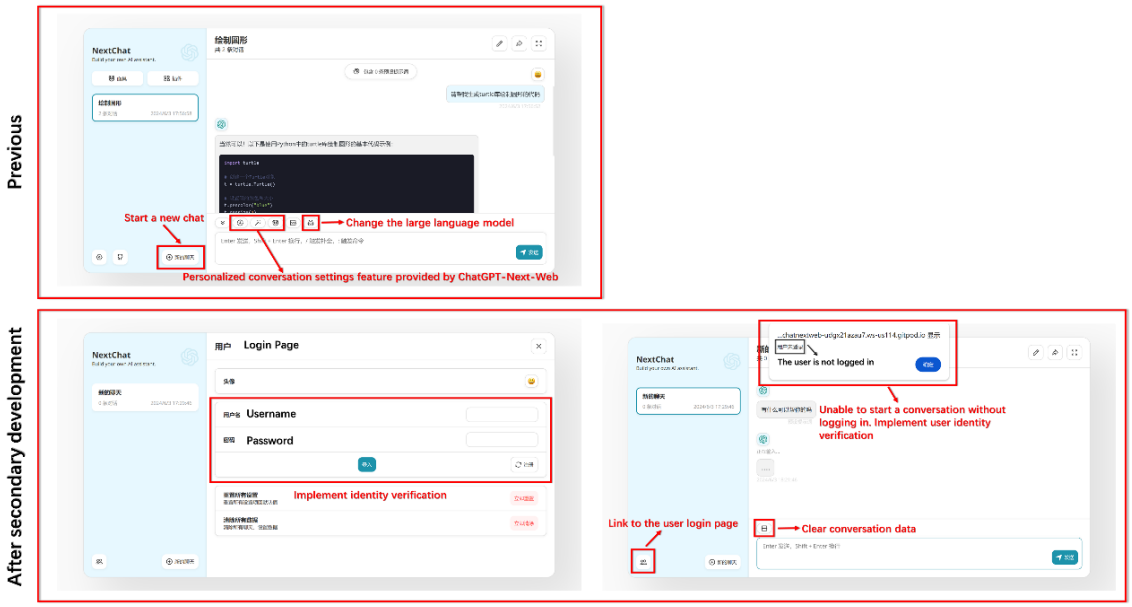

Figure 1. The GenAI platform used for the experiment

To facilitate smooth interaction with GenAI, we provided a "GenAI User Guide" (Liu et al., 2024) to the experimental group. Then both groups simultaneously conducted their studies in computer labs and were tasked with completing self-learning activities within 120 minutes under the guide of the task sheet. By the end of the second phase, students were required to submit five practice programmes.

During the post-test phase, we distributed programming self-efficacy post-tests to all students. After completing these assessments, both groups were assigned two more complex programming tasks based on the Turtle library, to be completed in 40 minutes without external resources.

The entire experimental process took place in standardised computer labs, where all computers had the same hardware configuration, and the learning process data of the learners were thoroughly recorded.

Data collection and data analysis

Learning product evaluation

Jones et al. (2013) argued that computer science involves both foundational principles and widely applicable concepts, integrating problem-solving and cognitive skills like abstraction and logical reasoning. Therefore, programming education should focus not only on coding but also on developing higher-order thinking skills (Buitrago Flórez et al., 2017). Academic assessment in programming should move beyond binary evaluations of correctness, instead deconstructing tasks into smaller components for a more systematic, objective and comprehensive framework.

Therefore, this study developed a two-dimensional programme evaluation framework to address RQ1, focusing on programming concepts and practices (Table 1), with six evaluation criteria. Programming

concepts assess programme functionality, while programming practices focus on code quality. After reaching consensus on the framework, we independently rated learners' Python programmes on a 5-level scale (0–4). A Spearman's rho correlation test showed significant consistency between raters' scores across all dimensions (coefficients > 0.7). The final evaluation for each programme was the average of both raters' scores. The final scores for the exercise and test phases were calculated by averaging the ratings for the exercise programme and post-test programme, respectively.

Table 1
Programme scoring framework

Dimension	Scoring criteria	Description
Programming concepts	Correct syntax	The syntax is correct.
	Logical correctness	The basic logic for problem-solving is correct.
	Functionality Completeness	All required programme functionalities are implemented.
Programming practices	Code structure	The code structure is clear, with modular and functional design.
	Code style	The code style adheres to Python coding standards, including consistent indentation, reasonable variable naming, clear and concise comments and code readability.
	Algorithm design	The problem-solving algorithm is characterised by novelty, simplicity and innovation, including the consideration of exception handling in the programme.

Programming self-efficacy measurement

To address RQ2, the study adapted the Computer Programming Self-Efficacy Scale designed by Ramalingam and Wiedenbeck (1998) to collect data on students' programming self-efficacy. The intensity of self-efficacy is measured using a 5-point Likert scale, ranging from 1 (*completely disagree*) to 5 (*completely agree*). The scale is divided into three dimensions: independence and persistence, complex programming tasks and self-regulation. The study used the scale for two tests, conducted before and after the experiment, and the scale demonstrated high internal consistency (Cronbach's $\alpha = 0.93$).

Learning behaviour data collection and analysis

To address RQ3, we collected learners' behavioural trajectory data during the programming process, using the learning behaviour recording script (Liu et al., 2024) from our prior research. The script recorded screenshots and actions from keyboard and mouse inputs in real time. The raw data were processed with Python and exported to a .csv file, including operation start time, location, behaviour window, specific events and keyboard input.

We then developed a programming behaviour coding framework to interpret the behavioural trajectory data. Building on Blikstein et al.'s (2014) classification of programming activities into coding, debugging and copying/adapting code, we refined the framework to account for learners' use of various Internet resources. The detailed coding scheme is presented in Table 2.

Table 2
Programming behaviour coding framework

Primary category	Secondary category	Code	Explanation
Code editing behaviour	Independent Coding	IC	Independently writing code
	Deleting	DC	Deleting input errors in code or text
	Copying	PC	Copying and pasting code from the Internet resources
	Applying	AC	Completing programming tasks with the help of Internet resources
	Debugging and Running	MC	Running, debugging code and checking results

	Content Revising	RC	Modifying code content
	Formatting Revising	RF	Modifying code formatting
Non-code editing behaviour	Creating or Opening File	OF	Initiating the writing of a programme
	Closing File	CF	Concluding the writing of a programme
	Saving File	SF	Saving the file
	Editing Comments	IE	Editing annotations
	Observing or reflecting	OC	Mouse operation events are present in the programming interface, but there are no keyboard input events.
Resource interaction behaviour	Information Retrieval	IR	Retrieving task-related information through GenAI or search engines
	Information Viewing	IL	Viewing Internet resources or GenAI information related to programming tasks
Unrelated behaviour		UB	Behaviours not related to programming (e.g., browsing other pages)

Guided by the coding framework, two of us (SL & QD) independently coded 15% of the data, using screenshots to assign codes at each behavioural shift. The initial Kappa coefficient of 0.765 indicated strong inter-coder reliability, with discrepancies resolved through discussion. The remaining data was coded independently, and disagreements were settled by consensus. Quantitative analysis and process mining were then used to assess GenAI's impact on students' learning behaviours.

Data collection and analysis of interactive behaviours of experimental group Learners

To address RQ4, the study explored behavioural differences among experimental group learners during collaborative programming tasks with GenAI and compared these patterns' impact on academic performance. The resource interaction behaviour section of the programming behaviour coding framework (Table 2) was refined to enable a more detailed analysis of the learners' interactions with GenAI, providing deeper insights into the learning process. The updated framework for resource interaction behaviours is shown in Table 3, while the code editing behaviour, non-code editing behaviour and irrelevant behaviour sections remained unchanged.

Table 3
Resource interaction behaviours coding framework of experimental group

Primary category	Secondary category	Code	Explanation
Resource interaction behaviour	Information Retrieval	IR	Retrieving task-related information through GenAI or search engines
	Information Viewing	IL	Viewing Internet resources or GenAI information related to programming tasks
	Direct Command	DC	Initiating the first inquiry to GenAI about a problem, obtaining information related to task completion
	Direct Question	DQ	Commanding GenAI to generate content in response to a new request
	Expansion Command or Question	EA	Adding additional conditions or probing for more in-depth content
	Paraphrased Command or Question	PA	Re-interacting with GenAI after modifying the expression
	Repeated Command or Question	RA	Repeating the inquiry
	Feedback	FB	Providing feedback on the content generated by GenAI

The two of us (SL & QD) then re-coded the experimental group's learning behaviours and clustered learners into high- and low-performance groups. This allowed for a comparison of academic performance differences linked to distinct learning behaviour patterns under GenAI support.

Results

The impact of GenAI support on students' programming academic performance

Analysis of overall scores

The Shapiro-Wilk normality test (Table 4) showed that the final scores of both the practice and post-test programmes did not follow a normal distribution. Therefore, all subsequent difference tests in the statistical analysis were conducted using SPSS non-parametric methods.

Table 4
Shapiro-Wilk normality test for programme scores

Programme type	Group	Shapiro-Wilk		
		Statistic	df	Sig.
Exercise programme	Experimental	.476	27	.000***
	Control	.793	28	.000***
Test programme	Experimental	.825	27	.000***
	Control	.588	28	.000***

*$p < 0.05$. **$p < 0.01$. ***$p < 0.001$.

Descriptive statistics for the overall scores of the two groups' practice programmes indicate that the experimental group, which used GenAI for autonomous learning, outperformed the control group (see Table 5). The Mann-Whitney test (Table 6) shows that the median score for the experimental group was 22.30, compared to 18.75 for the control group, with a significant difference in practice programme scores ($Z = -5.129$; $p = 0.000 < 0.05$). This suggests a significant positive impact of GenAI on phase-based learning outcomes. However, no significant difference was found in test programme performance, despite the experimental group's higher average score ($Z = -6.15$; $p = 0.538 > 0.05$), indicating that the positive impact of GenAI on phase-based outcomes did not persist in the subsequent test, which excluded Internet resources.

Table 5
Descriptive statistics of programme scores

Programme type	Group	Mean	Variance	SD	Minimum	Maximum
Exercise programme	Experimental	21.074	13.316	3.649	5.80	23.20
	Control	17.800	14.823	3.850	3.90	21.30
Test programme	Experimental	17.796	19.755	4.445	2.50	22.75
	Control	17.580	23.824	4.881	.00	21.25

Table 6
Mann-Whitney test results

Programme type	Group	M (P25, P75)	Wilcoxon rank-sum test	
			Z value	*Z* value
Exercise programme	Experimental	22.3 (21.3, 22.5)	-5.129	0.000***
	Control	18.75 (16.875, 20.5)		
Test programme	Experimental	19.25 (15.5, 21.0)	-0.615	0.538
	Control	18.875 (17.875, 19.75)		

*$p < 0.05$. **$p < 0.01$. ***$p < 0.001$.

Analysis of scores across different dimensions

The study compared the experimental and control groups' learning products across various dimensions using non-parametric tests due to non-normal data distribution. As shown in Table 7, the Mann-Whitney

test revealed significant differences in algorithm design, code structure and code style. The GenAI-supported experimental group exhibited better adherence to Python naming conventions, more appropriate comments, improved readability, and more efficient algorithms. However, in the test programmes, the advantage was limited to code style, with the experimental group showing significant shortcomings in logical correctness and functionality completeness compared to the control group.

Table 7
Mann-Whitney test results

Programme type	Scoring dimension	Group	M (P25, P75)	Wilcoxon rank-sum test	
				Z value	*p* value
Exercise programme	Correct Syntax	Experimental	4.0 (3.9, 4.0)	-1.771	.077
		Control	4.0 (3.625,4.0)		
	Logical Correctness	Experimental	3.9 (3.7, 4.0)	-1.195	.232
		Control	3.75 (3.225, 4.0)		
	Functionality Completeness	Experimental	4.0 (3.8, 4.0)	-1.903	.057
		Control	3.8 (3.0, 4.0)		
	Algorithm Design	Experimental	3.6 (3.3, 3.8)	-3.757	.000***
		Control	2.65 (2.2, 3.4)		
	Code Structure	Experimental	3.1 (3.0, 3.5)	-3.975	.000***
		Control	2.65 (2.35, 3.0)		
	Code Style	Experimental	3.6 (3.4, 3.8)	-5.439	.000***
		Control	2.05 (1.8, 2.3)		
Test programme	Correct Syntax	Experimental	4.0 (3.5, 4.0)	-1.096	.273
		Control	4.0 (3.75,4.0)		
	Logical Correctness	Experimental	3.5 (2.5, 3.75)	-2.245	.025*
		Control	4.0 (3.3125, 4.0)		
	Functionality Completeness	Experimental	3.25 (2.25, 3.75)	-2.319	.020*
		Control	4.0 (3.25, 4.0)		
	Algorithm Design	Experimental	3.0 (2.25, 3.75)	-.314	.754
		Control	2.125 (2.5, 3.5)		
	Code Structure	Experimental	2.75 (2.5, 3.0)	-.372	.710
		Control	2.75 (2.5, 2.75)		
	Code Style	Experimental	2.75 (2.0, 3.25)	-3.899	.000***
		Control	1.75 (1.5, 2.1875)		

*$p < 0.05$. **$p < 0.01$. ***$p < 0.001$.

The impact of GenAI support on students' programming self-efficacy

Since the mean scores on the three dimensions of the questionnaire met the criteria for normal distribution, independent samples *t* tests were used to test for significant differences in pretest and posttest self-efficacy between the experimental and control groups. The results, as shown in the Table 8, indicate that there were no significant differences in pretest self-efficacy across the three dimensions between the two groups.

Table 8
Independent samples t test results for pre-test self-efficacy

Dimensions	Statistics				Independent samples *t* test		
	Group	N	Mean	SD	*t*	df	Sig.
Independent and Persistent	Experimental	27	3.7963	.72107	.369	52	.713
	Control	27	3.7222	.75249			
Self-Regulation	Experimental	27	3.5648	.48334	1.637	52	.108
	Control	27	3.3056	.66627			
Complex Programming Tasks	Experimental	27	2.7259	.77885	.728	52	.470
	Control	27	2.5556	.93370			

As for the post-test, independent samples t tests revealed that the changes in self-efficacy across various dimensions differed between groups (as shown in Table 9). Specifically, students in the experimental group, who collaborated with GenAI to complete programming tasks, showed significantly higher mean scores in the dimensions of self-regulation and complex programming tasks compared to the control group. Overall, the application of GenAI has a positive impact on enhancing students' programming self-efficacy.

Table 9
Paired samples t test results for self-efficacy before and after the experiment by group

Dimensions	Statistics				Independent samples *t* test		
	Group	N	Mean	SD	*t*	df	Sig.
Independent and Persistent	Experimental	27	3.99	.57	1.125	52	.266
	Control	27	3.80	.67			
Self-Regulation	Experimental	27	3.61	.35	2.778	52	.008**
	Control	27	3.34	.36			
Complex Programming Tasks	Experimental	27	3.39	.75	2.550	52	.014*
	Control	27	2.81	.89			

*$p < 0.05$. **$p < 0.01$. ***$p < 0.001$.

The impact of GenAI support on students' programming learning behaviour

Comparison of behaviour frequencies

The study analysed the frequency distribution of students' programming behaviours during self-directed learning (see Table 10 and Figure 3). Both groups primarily engaged in coding activities, with debugging and running (MC) being the most frequent tasks. The control group exhibited higher frequencies in independent coding (IC), deleting (DL), debugging and running (MC) and saving files (SF), indicating more frequent engagement in core programming tasks. Conversely, the experimental group had higher frequencies in editing comments (IE), information retrieval (IR) and copying (PC), suggesting greater reliance on external resources and more frequent editing of comments during programming.

Table 10
Frequency of behaviour occurrence per individual

Code	Experimental group			Control group		
	Frequency	SD	Weight (%)	Frequency	SD	Weight (%)
Applying (AC)	7.11	5.048	4.51%	8.39	5.138	4.71%
Closing File (CF)	6.22	2.225	3.95%	5.75	1.531	3.21%
Deleting (DL)	4.85	3.780	3.08%	6.86	5.714	4.13%
Independent Coding (IC)	4.44	4.173	2.82%	11.00	8.944	6.44%
Editing Comments (IE)	1.41	1.966	0.89%	.46	.999	0.26%
Information Viewing (IL)	18.85	8.529	11.95%	27.11	12.491	15.06%
Information Retrieval (IR)	16.44	6.624	10.43%	5.68	3.632	3.23%
Debugging and Running (MC)	26.15	10.369	16.58%	32.36	15.205	17.99%

Observing or Reflecting (OC)	11.81	7.142	7.49%	11.86	10.742	6.58%
Creating or Opening File (OF)	6.22	2.190	3.95%	5.68	1.467	3.17%
Copying (PC)	12.63	6.452	8.01%	6.29	5.346	3.23%
Content Revising (RC)	10.67	7.616	6.76%	19.54	10.101	11.01%
Formatting Revising (RF)	4.22	3.320	2.68%	4.64	5.057	2.63%
Saving File (SF)	24.37	9.931	15.45%	30.07	13.383	16.76%
Unrelated Behaviour (UB)	2.30	2.145	1.46%	2.86	2.475	1.60%

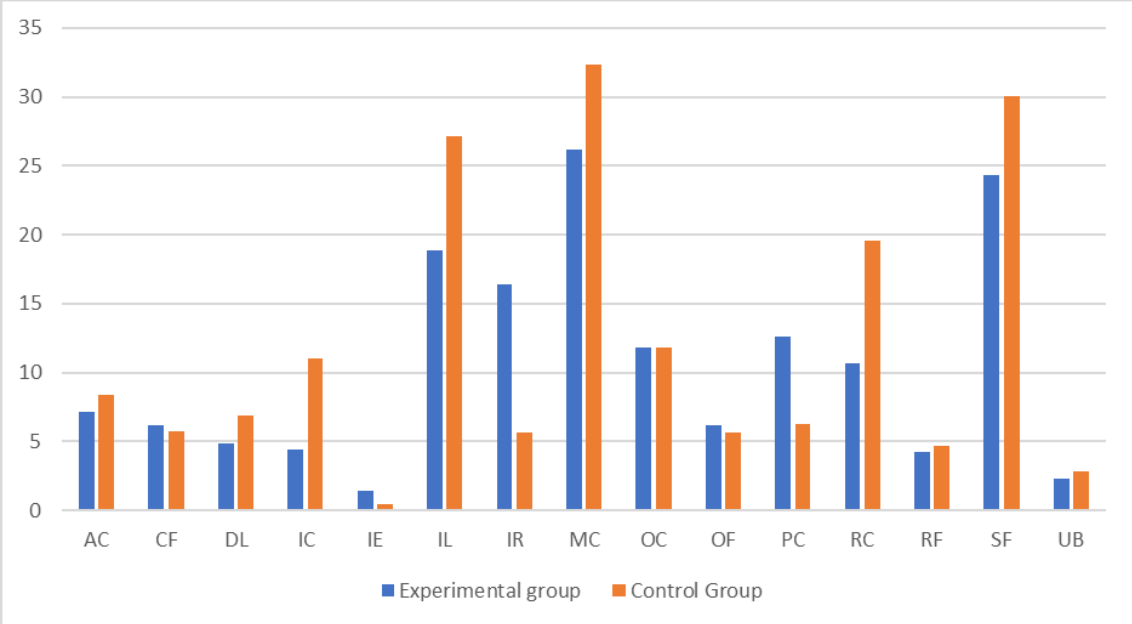

Figure 2. Frequency distribution of programming behaviours for both groups

The study conducted a differential analysis of behaviour frequencies between the two groups using independent samples *t* tests in SPSS. Normality tests indicated that only the data for applying (AC) and information viewing (IL) followed a normal distribution. Therefore, a logarithmic transformation was applied to the behaviour frequency data, and non-parametric tests were used for non-normally distributed data (see Table 11 and Table 12).

Significance tests revealed significant differences between the two groups in behaviours such as independent coding (IC), information viewing (IL), information retrieval (IR), content revising (RC), editing comments (IE) and copying (PC) ($p < 0.05$). Specifically, the experimental group relied more on GenAI-generated code, while the control group engaged more in independent coding and debugging. In terms of resource interaction, the experimental group focused more on information retrieval (IR), while the control group engaged more in information viewing (IL). This suggests that the experimental group accessed relevant information more efficiently via GenAI, compared to the control group, which had to browse external sources.

Table 11
Results of the independent samples t test of behaviour frequencies

Code	t	df	Sig.
Applying (AC)	-1.411	51	.164
Closing File (CF)	.896	53	.375
Editing Comments (IE)	1.281	18	.217
Debugging and Running (MC)	-1.559	53	.125
Observing or reflecting (OC)	.336	52	.738
Creating or Opening File (OF)	1.089	53	.281
Formatting Revising (RF)	-1.440	43	.157
Saving File (SF)	-1.794	53	.079
Unrelated Behaviour (UB)	-.147	42	.884

*$p < 0.05$. **$p < 0.01$. ***$p < 0.001$.

Table 12
Results of the non-parametric tests

Code	Mann-Whitney U	Wilcoxon W	Z	Sig.
Deleting (DL)	298	676	-1.353	0.176
Independent Coding (IC)	189.5	567.5	-3.192	.001**
Information Viewing (IL)	202	580	-2.966	.003**
Information Retrieval (IR)	51.5	457.5	-5.506	.000***
Copying (PC)	149.5	555.5	-3.854	.000***
Content Revising (RC)	184.5	562.5	-3.263	.001**

*$p < 0.05$. **$p < 0.01$. ***$p < 0.001$.

Learning process mining

The study used ProM Lite 1.4 with the Heuristic Miner algorithm to model learners' programming behaviours. Two thresholds were set for visualisation: a frequency threshold (0.1) to filter behaviours occurring at least three times, and a dependency threshold (0.9) to include only transitions with a dependency measurement of 0.9 or higher. The control group exhibited 13 types of behaviours, while the experimental group showed 11. Notably, behaviours like format revising (RF) and independent coding (IC) were absent in the experimental group's learning process model.

As shown in Figure 4, the control group' s learning process starts with OF (Creating or Opening File) and ends with CF (Closing File). Learners follow three common paths before debugging (MC): "OF → OC → IC → AC → MC", "OF → OC → IC → RC → (PC) → MC" and "OF → IL → AC → (IC) → MC". In the first path, learners open the file, reflect (OC), code independently (IC) and use Internet resources (AC) if needed. The second path involves coding independently (IC), correcting errors (RC) and seeking help (PC) if necessary. The third path starts with information retrieval (IR) or viewing (IL), followed by applying Internet resources (AC) and independent coding (IC).

Six bidirectional transitions focus on independent coding (IC) and code revising (RC): "IC ↔ OC", "IC ↔ AC," , "IC ↔ RC", "PC ↔ RC", "RF ↔ RC", and "MC ↔ RC". This shows that learners initially attempt tasks independently, use Internet resources when needed and then revise through debugging.

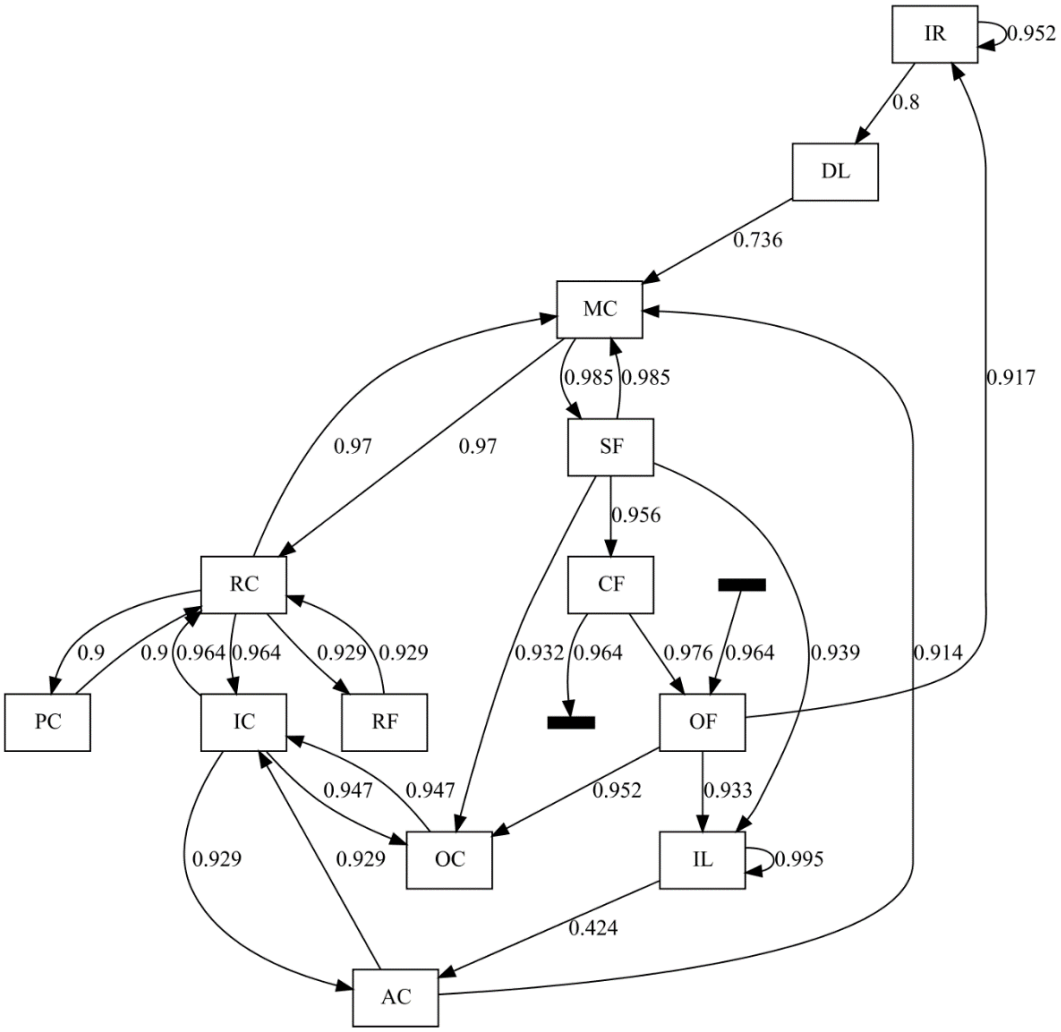

Figure 3. Control group learning process model

As shown in Figure 5, the experimental group's learning process starts with OF (Creating or Opening File) and ends with CF (Closing File). Before debugging, learners typically follow the path "OF → OC → IL → AC/PC → MC". After creating or opening the file (OF), they reflect on the task (OC), review GenAI content (IL) and either copy and paste the GenAI-generated code (PC) or apply it to the task (AC) before debugging (MC).

Learners experience bidirectional transitions between behaviors like "PC ↔ IL" and "AC ↔ IL", repeatedly searching for useful information. They cycle through observing the code (OC) and browsing information (IL), gradually building their understanding. They then debug (MC), seek solutions from GenAI (IR) and copy GenAI-generated content (PC), indicating strong reliance on GenAI outputs.

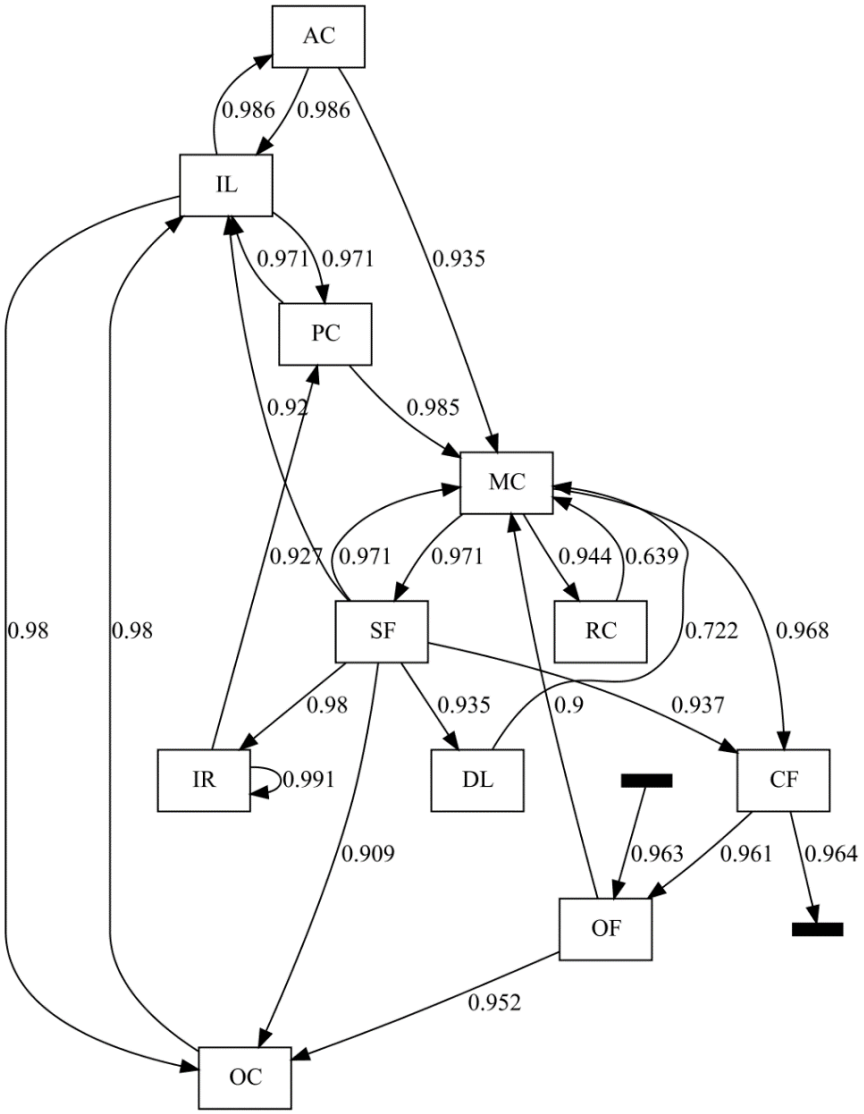

Figure 4. Experimental group learning process model

The relationship between different learning behaviour patterns under GenAI Support and academic performance

Learner clustering based on behaviour patterns

Ward's clustering (WC) tends to merge smaller groups, determining similarity between data points by calculating the sum of squared deviations between sequences (Murtagh & Legendre, 2014). Ouyang et al. (2023) noted that compared to other clustering methods, WC generates effective clusters of relatively uniform size. Therefore, this study used WC to categorise learners' behaviour sequences into distinct patterns. To address length discrepancies in the sequences, which could affect clustering, a hidden Markov model (HMM) was employed to standardise the sequences by padding them to equal lengths. Learners were then clustered using WC based on these standardised sequences.

HMM captures hidden states, revealing dynamic characteristics and transition patterns in data (Eddy, 1996). In this study, hidden states represented distinct learning strategies. The optimal HMM was fitted to learners' behaviour sequences, with the number of hidden states (n_states) varied between 2 and 10. The best model was selected from 100 runs using hmmlearn, with model fit assessed via the Bayesian information criterion. HMM predicted and completed behaviour sequences, and clustering was performed using WC via Python's SciPy package, as shown in Figure 6.

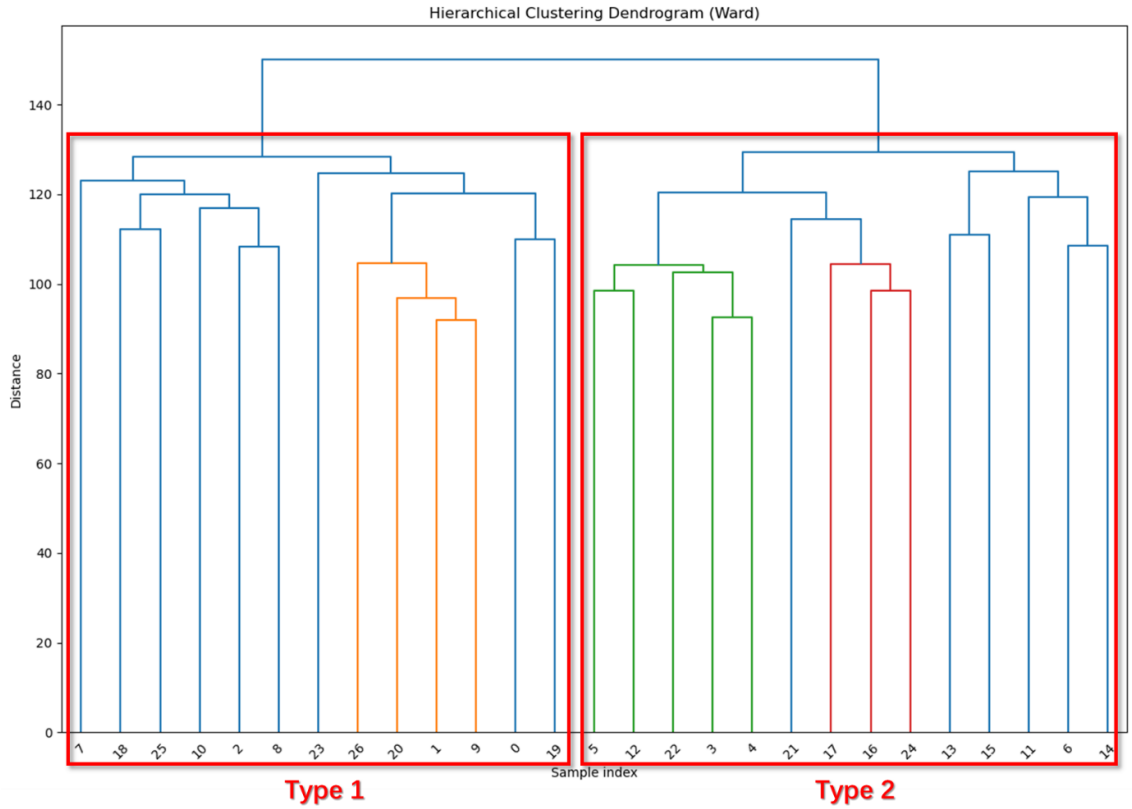

Figure 5. Clustering results of learners based on learning behaviours

The clustering results revealed two distinct learning behaviour patterns among the experimental group learners, consisting of 13 and 14 students, respectively corresponding to Type 1 and Type 2 (see Figure 6). Type 2 learners had a higher average test score (Mean = 19.50) than Type 1 learners (Mean = 15.96), with a significant difference (p = 0.039 < 0.05). SPSS was used to standardise pretest and average test scores using Z scores, and the difference between post-test and pretest scores measured learning improvement. Type 2 learners showed greater gains (Mean = 0.13), while Type 1 learners showed minimal improvement (Mean = -0.14). Thus, Type 1 correlates with lower performance, and Type 2 with higher performance (see Table 13 and Figure 7), with Type 1 as the low-performance group and Type 2 as the high-performance group.

Table 13
Descriptive statistics and significance test of learners' post-test scores and learning gain

	Descriptive statistics					Significance of difference testing			
Statistic	Group	N	Mean	SD	SE	Mann-Whitney U	Wilcoxon W	Z	Sig.
Post-test score	Type 1	13	16.13	5.48	1.52	48.50	139.50	-2.065	0.039*
	Type 2	14	19.34	2.55	0.68				
Learning gain	Type 1	13	-0.14	1.44	0.40	No significant difference			
	Type 2	14	0.13	0.86	0.23				

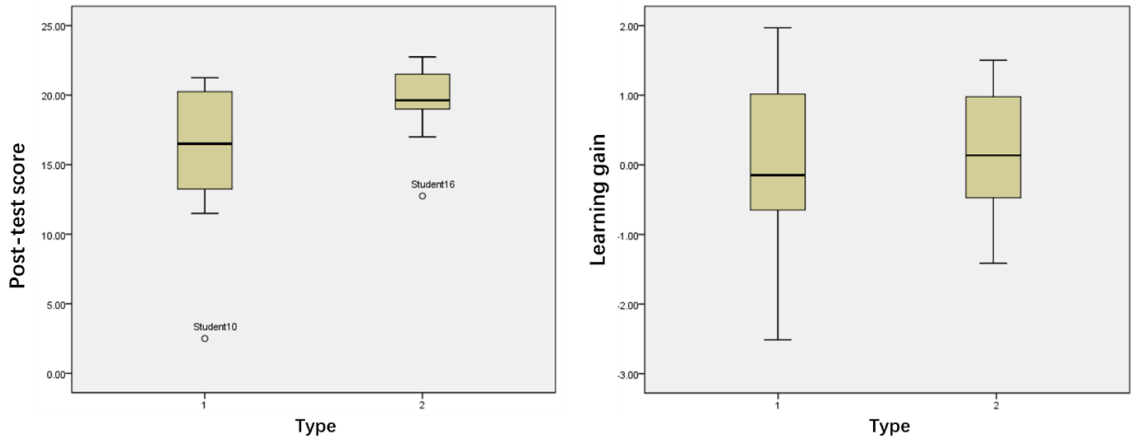

Figure 7. Post-test scores and learning gains in different clustering groups

Analysis of differences in learning process models

The R package pMineR (Gatta et al., 2017) generates overlay graphs to visually compare differences between two learning process models. The study used pMineR to compare the learning process models of high and low-performance groups (see Figure 8). In the graph, the elliptical nodes represent different learning behaviours, while the connecting lines indicate possible transition paths between behaviours. Arrows show the direction of these transitions. Edges in green represent transitions where the low-performance group has a probability at least 0.3 lower than the high-performance group, while edges in red indicate transitions where the low-performance group has a probability at least 0.3 higher. Black edges signify no significant difference in transition probabilities between the two groups.

Figure 8 shows that low-performance learners are more likely to transition to code copying (PC), while high-performance learners engage more in code application (AC) and code observation (OC), which is linked to independent thinking. Low-performance learners tend to copy GenAI-generated code entirely, whereas high-performance learners apply it selectively. In direct interactions with GenAI, high-performance learners build their understanding through continuous questioning (DQ) and are more likely to engage in independent thinking afterwards (EA→OC, RA→OC, DC→OC, DQ→OC). In contrast, low-performance learners tend to copy the generated content directly (EA→PC, PA→PC, RA→PC, DC→PC). High-performance learners also engage more in evaluating GenAI content (FB). After providing feedback, they are more likely to expand the interaction (FB→EA) or selectively apply the content (FB→AC), reflecting critical application and deeper reflection.

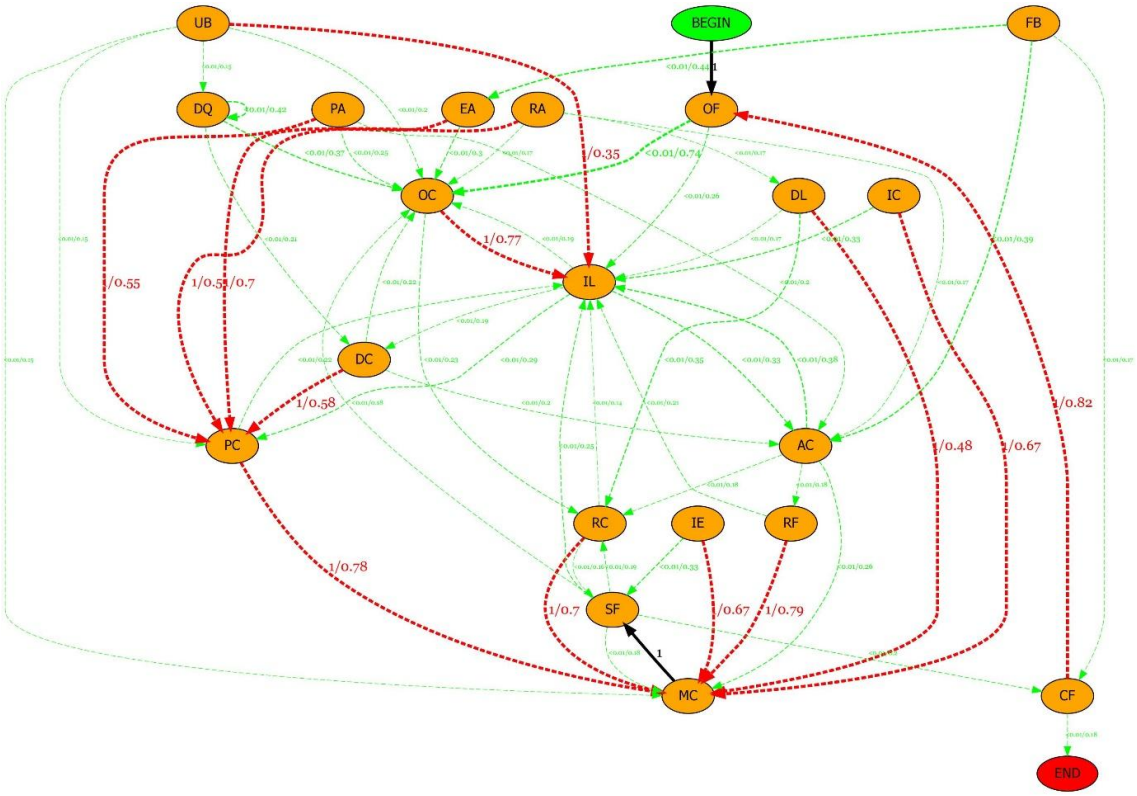

Figure 8. Comparison of programming process models between two groups

Discussion

The impact of GenAI on programming academic performance

For RQ1, we compared the academic performance of students in the experimental group (using GenAI for programming tasks) with the control group (independent problem-solving) across two phases (practice and testing) and six dimensions (syntax accuracy, logical correctness, functional completeness, algorithm design, code structure and code style). The experimental group outperformed the control group in the practice phase, particularly in algorithm design, code structure and code style, supporting the findings of Sun et al. (2024) and Qureshi (2023) that GenAI improves programming performance. However, our study found that these improvements did not persist into the testing phase, unlike Gong et al. (2024), suggesting that GenAI aids task completion but does not necessarily promote knowledge transfer. These results highlight the risk of over-reliance on GenAI, which may hinder independent critical thinking and limit long-term cognitive gains.

These findings highlight the importance of designing educational interventions that focus not only on immediate performance gains but also on the promotion of knowledge internalisation and transfer. To mitigate the risks of over-dependence on GenAI, educational institutions must address it by incorporating structured instructional designs that foster deeper cognitive engagement. Educators should guide students in balancing GenAI support with the development of sustainable problem-solving skills.

The impact of GenAI on programming self-efficacy

For RQ2, we utilised Ramalingam and Wiedenbeck's (1998) programming self-efficacy scale to compare the subjective emotional experiences of learners in the experimental and control groups during the programming learning process. Although no significant differences were observed in the pretest, post-test results revealed significant differences in the dimensions of self-regulation and complex programming tasks, with the experimental group demonstrating higher overall levels in these areas. This

indicates that GenAI-supported self-directed learning enhances learners' confidence in solving programming problems, thereby influencing their learning behaviours.

Studies have shown that learners with higher self-efficacy tend to achieve better learning outcomes compared to those with lower self-efficacy (Rohatgi et al., 2016; Wilson & Narayan, 2016). Similarly, tools like ChatGPT positively impact learners' subjective experiences, such as programming self-efficacy, contributing to improved learning outcomes and enhanced learning experiences. These findings provide theoretical support for integrating GenAI technologies into programming learning and broader educational contexts.

The impact of GenAI on programming learning behaviours

In addressing RQ1 and RQ2, a paradox arises regarding GenAI's impact on learning performance and self-efficacy. Although GenAI enhances learners' emotional experiences, it does not translate into improved cognitive outcomes. RQ3 explores this discrepancy by analysing learning behaviours at a micro-level.

We collected behavioural data through activity tracking scripts to examine students' approaches to programming tasks during self-directed learning. Using frequency analysis and process mining, we identified significant differences in behaviour patterns between the experimental group (using GenAI) and the control group (no GenAI support). The experimental group frequently sought external resources (IR) and copied content (PC), with a recurring sequence — copying code (PC) → debugging and running (MC) — indicating a heavy reliance on GenAI for task completion. This aligns with research showing that tools like ChatGPT can increase plagiarism (Bašić et al., 2023). In contrast, the control group exhibited more independent programming (IC), code revision (RC), deletion (DL) and transitions involving external resource application (AC), indicating a greater reliance on their own efforts and more critical engagement with resources. Unlike Sun et al. (2024), we found no significant difference in debugging and running (MC) frequencies between groups, with the experimental group even showing a lower MC frequency. This suggests that GenAI support does not necessarily increase student engagement.

The behavioural patterns of the experimental group, distinct from the control group, highlight their over-reliance on GenAI, which helps explain their lower post-test performance. This finding aligns with Fan et al. (2024), who identified metacognitive laziness in GenAI-supported environments. Thus, the use of technology in education triggers cognitive offloading , reducing cognitive demands for problem-solving by shifting resources to other tasks or facilitating complex cognition (Risko & Gilbert, 2016). Although cognitive offloading can enhance learning by reducing cognitive load (Sweller, 2011), over-reliance on technology may impair learning abilities (Sparrow et al., 2011). In unregulated learning environments, GenAI's powerful computational capacity fosters excessive cognitive outsourcing (Yu & Wang, 2023), leading students to develop a dependency on GenAI, further amplifying their laziness. This over-dependence leads to negative cognitive offloading, where GenAI dominates problem-solving, undermining students' knowledge mastery and hampering the development of critical thinking and problem-solving skills. Therefore, balancing cognitive outsourcing and preventing negative offloading is essential for maximising GenAI's benefits, fostering hybrid intelligence and addressing key challenges in future education.

The relationship between different learning behaviour patterns and academic performance under GenAI support

RQ4 explores the underlying mechanisms through which GenAI enhances the programming learning performance by comparing the learning behaviour patterns of high- and low-performance groups within the experimental group. The analysis reveals that high-performance learners actively construct their understanding of programming problems through questioning (DQ) and engage in independent thinking after interacting with GenAI (e.g., EA→OC, DQ→OC), often refining their understanding through feedback or follow-up questions. In contrast, low-performance learners tend to frequently copy GenAI-generated content (e.g., PA→PC, RA→PC).

This contrast highlights the importance of epistemic agency (Rudolph et al., 2023), with high-performing students actively questioning, analysing and critiquing AI-generated content to construct knowledge and solve problems. They engage in iterative questioning and deep interactions with GenAI, developing their own understanding. In contrast, low-performing students exhibit over-reliance and passive cognitive offloading. To maximise GenAI's benefits, fostering epistemic agency and cognitive engagement is crucial. This can be achieved through structured guidance, scaffolding and well-designed instructional activities that optimise GenAI's positive impact. Standardised learning support systems can help balance technological assistance with students' cognitive agency, encouraging ethical, in-depth collaboration with GenAI focused on problem-solving rather than superficial reliance.

Conclusions, limitations and future directions

This study empirically highlights the unique impact of GenAI-supported programming education, deepening our understanding of technology-enabled learning. It analyses GenAI's effects on academic performance, self-efficacy and learning behaviours, explaining the contradictions between academic performance and self-efficacy through behavioural analysis. Additionally, the study compares the learning behaviour patterns of high- and low-performance groups, offering insights into the mechanisms by which GenAI enhances learning.

However, the study has limitations. Its sample size and experimental duration restrict its coverage of different educational backgrounds and long-term effects. Additionally, it focused mainly on programming education; future research could extend to other academic fields to explore GenAI's broader impact on learning.

Although the study is limited to programming, its findings have broader implications for teaching across disciplines. On one hand, it highlights that the benefits of cognitive offloading in GenAI-supported learning environments depend on learners' cognitive engagement and reliance on technology. This underscores the importance of active questioning and knowledge construction, enriching our understanding of human-AI collaboration and complementing existing theories on technology-enhanced education. These insights also provide a theoretical foundation for developing future AI-assisted learning tools and are relevant to other learning activities involving ill-structured problem-solving. Achieving human intelligence augmentation through technology remains a core goal of educational technology. On the other hand, this study offers practical recommendations for educators and instructional designers to optimise technology-supported learning. By integrating GenAI into instructional design, its potential to support learning can be maximized while mitigating excessive technological dependence. For example, GenAI can provide reflective text analysis resources in writing, create contextualized dialogue exercises in language learning and offer initial stylistic suggestions in artistic creation. Across different learning scenarios, leveraging GenAI's rich cognitive resources can facilitate meaning-making, critical thinking and creative expression through human-AI interaction, ultimately harnessing human intelligence to achieve optimal learning outcomes in a hybrid intelligence framework.

Although this study provides initial insights into GenAI's impact on student learning, further research is needed. First, as the findings are based on a learning context with Chinese students, similar studies in

diverse cultural and national settings could help validate and expand these insights. Second, with GenAI's inevitable integration into higher education, future research should explore effective ways to incorporate GenAI into teaching practices, investigating its potential roles (e.g., learning companion, teaching assistant) and developing educational strategies and tools. This would enhance learners' cognitive agency, mitigate over-reliance on technology and achieve human-AI collaborative learning goals, ultimately empowering education through technology.

Author contributions

Author 1: Conceptualisation, Investigation, Data curation, Formal analysis, Behavioural coding, Writing – original draft; Author 2: Conceptualisation, Supervision, Investigation, Writing – review and editing; Author 3: writing – review and editing, Behavioural coding.

Acknowledgements

This work was supported by the Educational Science Planning Project Fund of Jiangsu Province, China, for the year 2023 (B/2023/01/193).

References

Alshater, M. (2022). *Exploring the role of artificial intelligence in enhancing academic performance: A case study of ChatGPT*. SSRN. https://doi.org/10.2139/ssrn.4312358

Bandura, A. (1982). Self-efficacy mechanism in human agency. *American Psychologist, 37*(2), 122–147. https://doi.org/10.1037//0003-066x.37.2.122

Barke, S., James, M. B., & Polikarpova, N. (2023). Grounded Copilot: How programmers interact with code-generating models. *Proceedings of the ACM on Programming Languages*, *7*(OOPSLA1), 85–111. https://doi.org/10.1145/3586030

Bašić, Ž., Banovac, A., Kružić, I., & Jerković, I. (2023). *Better by you, better than me? ChatGPT-3 as writing assistance in students' essays*. EdArXiv. https://doi.org/10.35542/osf.io/n5m7s

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin, 39*(2), 32–36. https://doi.org/10.1145/1272848.1272879

Bernius, J. P., Krusche, S., & Bruegge, B. (2022). Machine learning based feedback on textual student answers in large courses. *Computers and Education: Artificial Intelligence, 3*, Article 100081. https://doi.org/10.1016/j.caeai.2022.100081

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences, 23*(4), 561–599. https://doi.org/10.1080/10508406.2014.954750

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

Chen, A., Xiang, M., Zhou, J., Jia, J., Shang, J., Li, X., Gašević, D., & Fan, Y. (2025). Unpacking help-seeking process through multimodal learning analytics: A comparative study of ChatGPT vs Human expert. *Computers & Education, 226*, Article 105198. https://doi.org/10.1016/j.compedu.2024.105198

Dellermann, D., Ebel, P., Söllner, M., & Leimeister, J. M. (2019). Hybrid intelligence. *Business & Information Systems Engineering, 61*(5), 637–643. https://doi.org/10.1007/s12599-019-00595-2

Ding, L., Li, T., Jiang, S., & Gapud, A. (2023). Students' perceptions of using ChatGPT in a physics class as a virtual tutor. *International Journal of Educational Technology in Higher Education, 20*, Article 63. https://doi.org/10.1186/s41239-023-00434-1

Dou, W., & He, W. (2010). Compatibility and requirements analysis of distributed pair programming. In Z. Hu & Z. Ye (Eds.), *Proceedings of the Second International Workshop on Education Technology and Computer Science* (Vol. 3, pp. 467–470). IEEE. https://doi.org/10.1109/etcs.2010.367

Eddy, S. R. (1996). Hidden markov models. *Current Opinion in Structural Biology, 6*(3), 361–365. https://doi.org/10.1016/S0959-440X(96)80056-X

Eysenbach, G. (2023). The role of ChatGPT, generative language models, and artificial intelligence in medical education: A conversation with ChatGPT and a call for papers. *JMIR Medical Education, 9*, Article e46885. https://doi.org/10.2196/46885

Fan, Y., Tang, L., Le, H., Shen, K., Tan, S., Zhao, Y., Shen, Y., Li, X., & Gašević, D. (2024). Beware of metacognitive laziness: Effects of generative artificial intelligence on learning motivation, processes, and performance. *British Journal of Educational Technology*, *56*(2), 489–530. https://doi.org/10.1111/bjet.13544

Gatta, R., Lenkowicz, J., Vallati, M., Rojas, E., Damiani, A., Sacchi, L., De Bari, B., Dagliati, A., Fernandez-Llatas, C., Montesi, M., Marchetti, A., Castellano, M., & Valentini, V. (2017). pMineR: An innovative R library for performing process mining in medicine. In A. Teije, C. Popow, J. Holmes, & L. Sacchi (Eds.), *Lecture notes in computer science: Vol. 10259. Artificial intelligence in medicine* (pp. 351–355). Springer. https://doi.org/10.1007/978-3-319-59758-4_42

Gong, X., Li, Z., & Qiao, A. (2024). Impact of generative AI dialogic feedback on different stages of programming problem solving. *Education and Information Technologies*. https://doi.org/10.1007/s10639-024-13173-1

Jayagopal, D., Lubin, J., & Chasins, S. E. (2022). Exploring the learnability of program synthesizers by novice programmers. In M. Agrawala, J. O. Wobbrock, E. Adar, & V. Setlur (Eds.), *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (pp. 1–15). ACM. https://doi.org/10.1145/3526113.3545659

Jones, S. P., Mitchell, B., & Humphreys, S. (2013). *Computing at school in the UK*. Microsoft. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/ComputingAtSchoolCACM.pdf

Kanaparan, G., Cullen, R., & Mason, D. D. M. (2017). Self-efficacy and behavioural engagement in introductory programming courses. *Proceedings of the 21st Pacific-Asia Conference On Information Systems*, *Malaysia*, Article 209. https://aisel.aisnet.org/pacis2017/209

Kuttal, S. K., Gerstner, K., & Bejarano, A. (2019). Remote pair programming in online CS education: Investigating through a gender Lens. In J. Smith, C. A. Bogart, J. Good, & S. D. Fleming (Eds.), *Proceedings of the 2019 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 75–85). IEEE. https://doi.org/10.1109/vlhcc.2019.8818790

Liang, J., Wang, L., Luo, J., Yan, Y., & Fan, C. (2023). The relationship between student interaction with generative artificial intelligence and learning achievement: Serial mediating roles of self-efficacy and cognitive engagement. *Frontiers in Psychology*, *14*, Article 1285392. https://doi.org/10.3389/fpsyg.2023.1285392

Lin, G. Y. (2016). Self-efficacy beliefs and their sources in undergraduate computing disciplines: An examination of gender and persistence. *Journal of Educational Computing Research*, *53*(4), 540–561. https://doi.org/10.1177/0735633115608440

Liu, J., Li, S., & Dong, Q. (2024). Collaboration with generative artificial intelligence: An exploratory study based on learning analytics. *Journal of Educational Computing Research, 62*(5), 1014–1046. https://doi.org/10.1177/07356331241242441

Medeiros, R. P., Ramalho, G. L., & Falcao, T. P. (2019). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, *62*(2), 77–90. https://doi.org/10.1109/te.2018.2864133

Mhlanga, D. (2023). *FinTech and artificial intelligence for sustainable development*. Springer. https://doi.org/10.1007/978-3-031-37776-1_17

Moore, S., Nguyen, H. A., Bier, N., Domadia, T., & Stamper, J. (2022). Assessing the quality of student-generated short answer questions using GPT-3. In In I. Hilliger, P. J. Muñoz-Merino, T. De Laet, A. Ortega-Arranz, & T. Farrell (Eds.), *Lecture notes in computer science: Vol. 13450. Educating for a new future: Making sense of technology-enhanced learning adoption* (pp. 243–257). Springer. https://doi.org/10.1007/978-3-031-16290-9_18

Murtagh, F., & Legendre, P. (2014). Ward's hierarchical agglomerative clustering method: Which algorithms implement Ward's criterion? *Journal of Classification*, *31*(3), 274–295. https://doi.org/10.1007/s00357-014-9161-z

Ouyang, F., Xu, W., & Cukurova, M. (2023). An artificial intelligence-driven learning analytics method to examine the collaborative problem-solving process from the complex adaptive systems perspective. *International Journal of Computer-Supported Collaborative Learning*, *18*, 39–66. https://doi.org/10.1007/s11412-023-09387-z

Pavlik, J. V. (2023). Collaborating with ChatGPT: Considering the implications of generative artificial intelligence for journalism and media education. *Journalism & Mass Communication Educator*, *78*(1), 84–93. https://doi.org/10.1177/10776958221149577

Qadir, J. (2023). Engineering education in the era of ChatGPT: Promise and pitfalls of generative AI for education. In M. El-Abd & A. Zeid (Eds.), *Proceedings of the 2023 IEEE Global Engineering Education Conference* (pp. 1–9). IEEE. https://doi.org/10.1109/EDUCON54358.2023.10125121

Qureshi, B. (2023). *Exploring the use of chatgpt as a tool for learning and assessment in undergraduate computer science curriculum: Opportunities and challenges*. EdArXiv. https://doi.org/10.48550/arXiv.2304.11214

Ramalingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research*, *19*(4), 367–381. https://doi.org/10.2190/C670-Y3C8-LTJ1-CT3P

Risko, E. F., & Gilbert, S. J. (2016). Cognitive offloading. *Trends in Cognitive Sciences, 20*(9), 676–688. https://10.1016/j.tics.2016.07.002

Rohatgi, A., Scherer, R., & Hatlevik, O. E. (2016). The role of ICT self-efficacy for students' ICT use and their achievement in a computer and information literacy test. *Computers & Education*, *102*, 103–116. https://doi.org/10.1016/j.compedu.2016.08.001

Rudolph, J., Tan, S., & Tan, S. (2023). ChatGPT: Bullshit spewer or the end of traditional assessments in higher education? *Journal of applied learning and teaching*, *6*(1), 342–363. https://doi.org/10.37074/jalt.2023.6.1.9

Sabzalieva, E., & Valentini, A. (2023). *ChatGPT and artificial intelligence in higher education: Quick start guide*. UNESCO. https://etico.iiep.unesco.org/en/chatgpt-and-artificial-intelligence-higher-education-quick-start-guide

Satratzemi, M., Xinogalos, S., Tsompanoudi, D., & Karamitopoulos, L. (2018). Examining student performance and attitudes on distributed pair programming. *Scientific Programming*, Article 523538, 1–8. https://doi.org/10.1155/2018/6523538

Sparrow, B., Liu, J., & Wegner, D. M. (2011). Google effects on memory: Cognitive consequences of having information at our fingertips. *Science*, *333*(6043), 776–778. https:/doi.org/10.1126/science.1207745

Sun, D., Boudouaia, A., Zhu, C., & Li, Y. (2024). Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. *International Journal of Educational Technology in Higher Education*, *21*, Article 14. https://doi.org/10.1186/s41239-024-00446-5

Sweller, J. (2011). Cognitive load theory. In J. P. Mestre & B. H. Ross (Eds.), *The psychology of learning and motivation: Cognition in education* (Vol. 55, pp. 37–76). Elsevier Academic Press. https://doi.org/10.1016/B978-0-12-387691-1.00002-8

Tai, T.-Y., & Chen, H. H.-J. (2024). Improving elementary EFL speaking skills with generative AI chatbots: Exploring individual and paired interactions. *Computers & Education*, *220*, Article 105112. https://doi.org/10.1016/j.compedu.2024.105112

Tsompanoudi, D., Satratzemi, M., Xinogalos, S., & Karamitopoulos, L. (2019). An empirical study on pair performance and perception in distributed pair programming. In M. Auer & T. Tsiatsos (Eds.), *Advances in intelligent systems and computing: Vol. 917. The challenges of the digital transformation in education* (pp. 762–771). Springer. https://doi.org/10.1007/978-3-030-11935-5_72

Wang, M., & Chen, Y. (2023). Research on the future vision of the intelligent integration of ChatGPT and online education. *Advances in Educational Technology and Psychology*, *7*(2), 128–134. https://clausiuspress.com/article/6675.html

Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. I In Å. Cajander & M. Daniels (Eds.), *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (pp. 39–44). ACM. https://doi.org/10.1145/2591708.2591749

Weber, F., Wambsganss, T., & Söllner, M. (2024). Enhancing legal writing skills: The impact of formative feedback in a hybrid intelligence learning environment. *British Journal of Educational Technology, 56*(2), 650–677. https://doi.org/10.1111/bjet.13529

Wilson, K., & Narayan, A. (2014). Relationships among individual task self-efficacy, self-regulated learning strategy use and academic performance in a computer-supported collaborative learning environment. *Educational Psychology, 36*(2), 236–253. https://doi.org/10.1080/01443410.2014.926312

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

Yu, S., & Wang, F. (2023). 人工智能教育应用的认知外包陷阱及其跨越 [The cognitive offloading trap in AI educational applications and its overcoming]. *e-Education Research*, *12*, 5–13. https://link.oversea.cnki.net/doi/10.13811/j.cnki.eer.2023.12.001

Zhang, L., & Xu, J. (2025). The paradox of self-efficacy and technological dependence: Unraveling generative AI's impact on university students' task completion. *The Internet and Higher Education, 65*, Article 100978. https://doi.org/10.1016/j.iheduc.2024.100978

Corresponding author: Jiangyue Liu, LJY@suda.edu.cn

Please cite as: Li, S., Liu, J., & Dong, Q. (2025). Generative artificial intelligence-supported programming education: Effects on learning performance, self-efficacy and processes. *Australasian Journal of Educational Technology*, *41*(3), 1–25. https://doi.org/10.14742/ajet.9932

Appendices

Appendix A: Programming self-efficacy scale

Independence and Persistence

(1) If I have enough time, I can complete programming tasks.
(2) With the help of technical tools, I can complete programming tasks.
(3) When I encounter difficulties while completing programming tasks, I promptly seek resources for help.
(4) If someone first demonstrates how to solve the problem, I can solve the programming tasks.
(5) If I encounter difficulties in programming tasks, I can find ways to overcome them.
(6) I can debug the programs I write and correct any errors to ensure they run properly.

Self-Regulation

(1) Even with many distractions, I find ways to focus on completing my programming tasks.
(2) Even if I am not interested in the task domain, I can find ways to motivate myself to complete programming tasks.
(3) If there is a time limit to complete programming tasks, I can effectively manage my time.
(4) I cannot come up with appropriate strategies for specific programming projects in a short time.

Complex Programming Tasks

(1) I can write logically correct code blocks using Python.
(2) I can write functional functions in Python and call pre-written functions.
(3) I can write a programme that others can understand and add functionalities to later.
(4) I can write lengthy and complex Python programmes to solve any given problem.
(5) I can rewrite lengthy and confusing parts of the code to make them more readable and clear.

Appendix B: GenAI usage experience questionnaire

Perceived Ease of Use

(1) I find the GenAI usage process clear and easy to understand.
(2) It is easy to obtain useful information from GenAI.
(3) Overall, it is easy to use generative AI tools to help me complete programming tasks.

Perceived Usefulness

(1) GenAI has improved the quality of my programming products.
(2) GenAI has increased the speed at which I complete programming tasks.
(3) With GenAI's help, I have proficiently mastered knowledge related to the Turtle library.
(4) GenAI has made learning programming easier for me.
(5) Overall, GenAI has an advantage for my programming learning.

Intention to Use

(1) I am willing to use GenAI in future programming learning.
(2) I am willing to try GenAI in various fields in the future, such as content creation, multimedia, entertainment and other work and life scenarios.
(3) GenAI is an attractive learning method for me.

Overall Attitude

(1) Overall, I enjoy using GenAI.
(2) To what extent do I trust the content generated by GenAI during its use?
3. To what extent do I believe that using generative AI tools in programming is ethical?
(3) What difficulties or thoughts do I have when using generative AI tools to complete programming tasks?

Appendix C: The results of the inter-rater reliability test for scoring data

Scoring dimensions	Correct syntax	Logical correctness	Functionality completeness	Code structure	Code style	Algorithm design
Spearman correlation coefficient	.901	.711	.883	.714	.886	.750
Sig.	.000***	.000***	.000***	.000***	.000***	.000***

*$p < 0.05$. **$p < 0.01$. ***$p < 0.001$.