

Article

Let's Code with GenAI: Exploring K-12 Teachers' Self-Efficacy, Value Beliefs, and Coding Performance

Seoljoo Kang *  and Wanju Huang 

Department of Curriculum and Instruction, Learning Design and Technology, Purdue University,
West Lafayette, IN 47907, USA; wanjuhuang@purdue.edu

* Correspondence: kang567@purdue.edu

Abstract

While computational thinking (CT) is increasingly vital in K-12 education, teaching it through text-based coding remains challenging for teachers. To address this gap, this study presents and evaluates a self-paced professional development (PD) module, “Let’s Code with GenAI,” created for K-12 teachers to enhance text-based coding and CT. Using a one-group pretest-posttest design, 34 pre-/in-service teachers completed the module in 2025, engaging with instructional videos and hands-on coding activities using Micro:bit and MakeCode (v11.3.22), with a GenAI assistant providing explanations and debugging support. Pre- and post-intervention data were collected using the Teacher Beliefs about Coding and Computational Thinking (TBaCCT) scale and a coding/CT assessment. Posttest scores were higher than pretest scores on teaching efficacy and value beliefs ($p < 0.001$ for both), as well as in coding self-efficacy ($p < 0.001$) and CT self-efficacy ($p = 0.015$). Coding/CT assessment scores were also higher at posttest ($p = 0.040$). No statistically significant correlations were found between self-efficacy and performance measures. Overall, the findings offer preliminary insights into participants’ post-intervention outcomes in a GenAI-supported, self-paced PD module, including self-efficacy, value beliefs, and coding/CT performance, while underscoring the need for future controlled studies.

Keywords: coding; computational thinking; self-efficacy; GenAI; K-12 teacher

1. Introduction

Computational thinking (CT) has become a core component of K-12 education, merging technical skills with broader computational literacy [1]. As ref. [2] notes, CT encompasses key skills such as abstraction, problem-solving, pattern recognition, and logical reasoning that are essential for addressing complex interdisciplinary problems. To foster this 21st-century competency, K-12 schools have incorporated both unplugged and plugged activities, including block-based and text-based coding [3]. While block-based programming provides an accessible entry point for younger learners, text-based coding supports deeper engagement with syntax and practical problem-solving [4]. However, the complexity of text-based coding often creates significant barriers to implementation, leading to teachers’ low confidence, technical anxiety, and diminished self-efficacy in K-12 settings [5]. In addition, insufficient teacher preparation, particularly in self-efficacy and pedagogical knowledge, remains a major obstacle to integrating coding and CT into K-12 education [6].

In this context, emerging research suggests that Generative Artificial Intelligence (GenAI) tools, such as ChatGPT (GPT-4, OpenAI) and Copilot (GPT-3, GitHub), can scaffold



Academic Editors:

José Fernández-Cerero and
Marta Montenegro-Rueda

Received: 29 June 2026

Revised: 14 July 2026

Accepted: 19 July 2026

Published: 23 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article
distributed under the terms and

conditions of the [Creative Commons
Attribution \(CC BY\) license](https://creativecommons.org/licenses/by/4.0/).

fold programming education by providing immediate code explanations, debugging assistance, and tailored feedback [7,8]. These GenAI-supported tools may help alleviate the pedagogical challenges of teaching text-based coding by offering highly personalized learning opportunities [9]. Yet, despite growing interest, the precise role of GenAI as an interactive scaffold in formal teacher professional development (PD) remains heavily underexplored [10]. This gap highlights the need to examine how GenAI-supported PD may relate to K-12 teachers' confidence, value beliefs, and readiness to teach coding and CT.

To address this issue, this study presents an exploratory pilot evaluation of "Let's Code with GenAI," a novel, self-paced online module for K-12 teachers. The distinct novelty of this module lies in its programmatic use of GenAI as an on-demand explanatory and debugging scaffold, designed to eliminate technical bottlenecks without requiring continuous human instructor resources. Methodologically, this exploratory evaluation followed three major sequential phases: (1) the systematic instructional design and development of the GenAI-integrated module, (2) the targeted recruitment of 34 K-12 in-service and pre-service teachers, and (3) a rigorous pretest-posttest assessment tracking changes in participants' CT beliefs and skills. Employing a one-group pretest-posttest design, it explores initial changes in teachers' self-efficacy, value beliefs, and coding/CT skills to guide future controlled studies. Importantly, the quantitative analysis revealed significant posttest improvements across key affective subscales, with statistical significance ranging from $p < 0.001$ to $p = 0.040$, thereby highlighting the potential efficacy of automated scaffolding and strengthening the study's empirical contribution to teacher training initiatives.

2. Literature Review

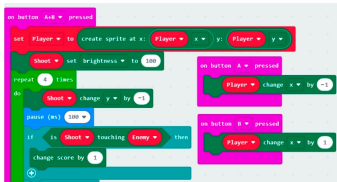
2.1. Evolution of Computational Thinking in K-12 Education

The foundations of CT were initially established by [11]. Subsequently, key CT skills—such as abstraction, problem-solving, pattern recognition, and logical reasoning—were further conceptualized by [2,12]. This perspective of CT emphasized the need for a computational approach to solve cognitive problems [13,14]. Building on the earlier ideas, ref. [15] introduced CT as a universally applicable skill set, arguing that it represents a fundamental skill for everyone—not just computer scientists—and is essential for problem-solving in daily life. Later, ref. [16] clarified the concept as "the mental activity in formulating a problem to admit a computational solution" (p. 2) and highlighted its broader relevance beyond computer science. Leveraging these foundational perspectives, CT has become integral to K-12 education, significantly enhancing learners' development. Specifically, ref. [17] framed these computational competencies as vital cognitive tools for school-level problem solving. Furthermore, ref. [18] articulated how integrated CT practices reinforce abstract concepts through interactive modeling, moving learners from passive technology consumption toward creative engagement. Consistent with this broader educational value, ref. [19] conducted a systematic review indicating that interdisciplinary CT integration effectively helps K-12 students navigate domain-specific challenges through robust structural alignment across disciplines. Recent studies extend CT further to incorporate creativity, algorithmic thinking, critical thinking, and collaboration [20,21]. Moreover, CT supports the holistic development of both students and teachers, enabling their personal and professional growth in democratic societies [22,23].

In K-12 settings, CT has been incorporated through unplugged activities as well as two forms of plugged activities: block-based coding and text-based coding [3,24]. All three activities have been utilized in various forms (See Table 1), such as animation, games, and other educational media, to support CT education [5,25]. However, many believe that text-based coding is not ideal for K-12 education due to its complex syntax and reliance

on special characters [5,26]. This perspective has hindered students from gaining the advantages of text-based coding, including a stronger conceptual understanding of CT and more authentic programming experiences [4,5].

Table 1. Comparison of unplugged, block-based, and text-based coding activities.

Type	Unplugged Activities	Block-Based Coding	Text-Based Coding
Definition	Computational activities without using computers	Visual programming utilizing code blocks that can be assembled	Traditional programming using text-based code in a specific language (e.g., Python v3.12, Java 8)
Tools	Physical objects like puzzles and toys	Software platforms like Scratch and Blockly	Text editors, Integrated development environments
Examples			<pre>43 # ----- 3. Automatically select device (prefer GPU) ----- 44 device = "cuda" if torch.cuda.is_available() else "cpu" 45 print(f"[INFO] Using device: {device}") 46 # ----- 4. Patch torch.load for GPU/CPU compatibility ----- 47 original_torch_load = torch.load 48 def patched_torch_load(f, map_location=None, **kwargs): 49 if map_location is None: 50 map_location = device 51 return original_torch_load(f, map_location=map_location, **kwargs) 52 torch.load = patched_torch_load 53 # ----- 5. Load model ----- 54 t0 = time.perf_counter() 55 model = CharacterBoxRTTS.from_pretrained(device=device) 56 torch.load = original_torch_load # Restore original torch.load 57 t1 = time.perf_counter() 58 print(f"[TIME] Model loaded in {(t1 - t0):.2f} seconds.")</pre>

2.2. Professional Development for Computational Thinking

CT education requires that teachers use pedagogical strategies such as contextualization, collaborative learning, and guided programming to support meaningful learning experiences [27,28]. However, many teachers have limited knowledge of CT and coding, which hinders their ability to incorporate these topics into classroom practice [29,30]. These challenges highlight the importance of PD in helping teachers learn to teach CT, particularly when it involves text-based coding and other unfamiliar computational methods.

Prior research indicates that CT-focused PD can support teachers' knowledge, pedagogical capability, and confidence in teaching CT [28,31]. At the same time, studies also show a persistent gap between what teachers learn in training and how they implement it in classrooms [32,33]. This gap suggests that PD must be supported by structures that enable ongoing, practice-embedded learning to facilitate meaningful classroom transfer.

Regarding these support mechanisms, ref. [31] suggested through empirical classroom observations that context-specific expert guidance and tailored peer exchanges significantly facilitate how teachers navigate distinct integration challenges. Further expanding this focus, ref. [29] demonstrated that PD incorporating active, collaborative practice effectively addresses teachers' initial knowledge gaps and technical anxieties by enhancing their computing self-efficacy. Similarly, ref. [34] utilized a design-based research framework to confirm that robust CT integration requires a community of practice in which pre- and in-service teachers co-design context-specific lessons aligned with school resources.

To clearly delineate the operational and pedagogical landscapes of these contemporary practices, Table 2 contrasts existing teacher PD models across key instructional dimensions.

As illustrated in Table 2, a critical structural limitation persists across contemporary teacher PD models. While intensive synchronous frameworks [31] and communities of practice [34] suffer from severe human-resource bottlenecks that limit scalability, standard independent asynchronous options [29] isolate learners without any interactive feedback loops. Consequently, the existing literature highlights a persistent operational gap between achieving institutional reach and providing immediate, personalized scaffolding during text-based programming tasks.

Table 2. Comparative matrix of the proposed module versus state-of-the-art teacher PD models.

PD Approach/Model	Primary Scaffolding Mechanism	Resource/ Human Cost	Scalability	Feedback Delivery
Traditional synchronous workshops [28,31]	Live instructor tutoring & real-time peer exchanges	High (requires dedicated human experts & scheduled time)	Low (constrained by cohort size and geographic limits)	Immediate but divided (instructor time is split among many)
Long-term communities of practice [34]	Collaborative co-design & collaborative lesson seed iteration	Very High (requires sustained institutional commitment)	Very Low (intended for deep localized networks)	Delayed (relies on scheduled peer reflection cycles)
Standard asynchronous e-learning [6]	Static video lectures & non-interactive code repositories	Low (pre-recorded content scales easily)	High (accessible anytime globally)	None to highly delayed (Lacks real-time syntax troubleshooting)

2.3. Self-Efficacy and Value Beliefs of Teachers

Another significant barrier to incorporating text-based coding and CT into K-12 education is teachers' low self-efficacy [6]. Self-efficacy, as defined by [35], refers to "people's beliefs about their capabilities to produce designated levels of performance that exercise influence over events that affect their lives" (p. 71). This concept underscores the profound influence of individuals' perceptions of their capabilities on decision-making processes and actions. Teachers exhibiting high self-efficacy demonstrate better classroom management, employ more effective instructional strategies, and provide more personalized learning assistance [36,37]. In addition to self-efficacy, teachers' value beliefs are crucial for effective instruction [38,39], particularly in curriculum development and content selection [40]. Moreover, alignment between teachers' personal values and organizational goals fosters greater motivation and instructional quality [41].

In essence, self-efficacy and value beliefs are foundational to teachers' instructional capabilities [42]. When teachers develop these mindsets, it not only improves instructional quality but also fosters student engagement and achievement, ultimately creating a reciprocal cycle of teacher–student growth [43,44]. Yet, building such robust confidence requires mastery experiences, targeted feedback, and reduced anxiety during task performance [35].

2.4. Personalized Learning with Generative Artificial Intelligence

Given the requirements for building self-efficacy, the conceptual difficulty of CT, and the challenges of text-based coding, GenAI has gained attention as a promising support for teacher PD [9,45]. Specifically, ref. [7] demonstrated that modifying AI-generated code suggestions can save up to 71% of programming time compared to writing from scratch. Furthermore, ref. [8] established that integrating customizable AI content and automated quiz feedback promotes more sustained study engagement, particularly benefiting underperforming learners. Similarly, ref. [46] showed that effective scaffolding depends on conversational agents providing stepwise guidance and tailored visualizations aligned with students' educational backgrounds. These capabilities are particularly relevant in programming contexts, in which support with syntax, troubleshooting, and step-by-step reasoning can lower barriers to participation.

From a theoretical standpoint, these affordances align with key mechanisms that support effective teacher PD, including guided practice, timely feedback, and opportunities for iterative learning [47,48]. This alignment is consistent with Papert's constructionism, which posits that learning occurs most deeply when learners actively construct knowledge

through making and working with tools such as programming languages [11]. In this way, GenAI can serve as a scaffold, enabling teachers to engage more actively with coding and CT concepts [49]. However, the educational value of GenAI depends on how it is integrated into instructional design [50]. Prior research highlights specific risks; ref. [51] noted that defensive institutional bans driven by integrity concerns are often counterproductive, especially given the high false-positive rates of current AI detection software. Furthermore, ref. [52] emphasized that GenAI's black-box nature and factual hallucinations necessitate critical data literacy rather than an overreliance on it as an objective reference tool. Similarly, ref. [53] identified that unregulated commercial deployment without structured policy guidelines threatens digital equity and risks widening existing student achievement gaps. These concerns underscore the importance of examining GenAI not as a standalone tool, but as part of a carefully designed instructional approach.

2.5. Synthesis and Purpose

In summary, the reviewed literature highlights a critical gap: despite the potential of GenAI tools to personalize learning and support programming education, their use in K-12 settings remains limited, particularly for text-based coding [9,54]. These pedagogical and environmental limits are deeply tied to existing instructional barriers. Specifically, ref. [6] found that a lack of initial teacher education in computer science content and pedagogy breeds low self-efficacy among elementary generalists. In addition, ref. [27] reported that inadequate pedagogical preparation leaves educators struggling to manage the wide ability gaps and differentiation needs that emerge during lessons. Likewise, ref. [31] confirmed that unsupportive school infrastructure and severe time constraints result in limited technical resources for practitioners when integrating coding and CT.

In response to these gaps, this study presents an exploratory evaluation focused on the feasibility of a self-paced GenAI-supported PD module, Let's Code with GenAI, for K-12 teachers learning text-based coding and CT. Rather than seeking definitive causal conclusions, this study is positioned as an initial step to examine how participation in this PD experience relates to changes in teachers' self-efficacy, value beliefs, and coding/CT performance. Ultimately, this inquiry seeks to provide foundational insights to inform the design of more rigorous future experimental studies.

3. Conceptual Framework

To guide this exploratory evaluation, this study adopts a conceptual framework synthesized from existing literature (see Figure 1).

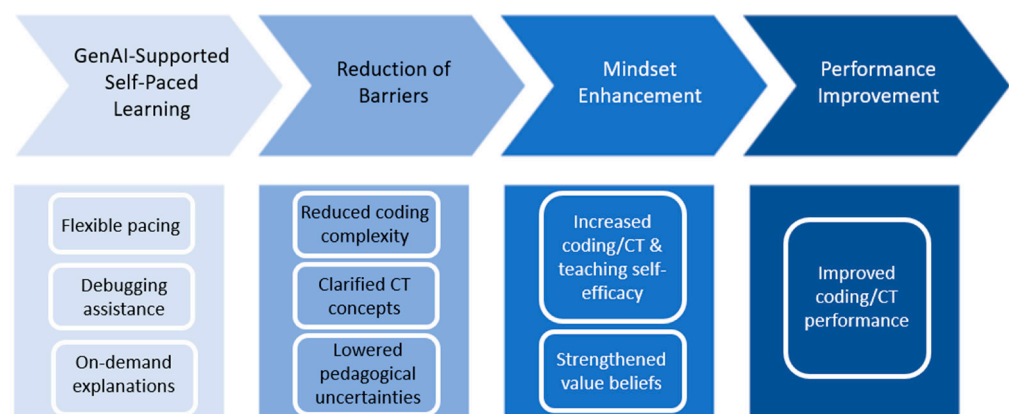


Figure 1. Conceptual framework for text-based coding and CT education in K-12.

The framework posits that GenAI-supported self-paced learning can provide flexible pacing, on-demand explanations, and debugging assistance, enabling teachers to revisit challenging coding and CT concepts as needed [5,55]. This design is consistent with the constructionist view of learning, in which understanding develops through active engagement with content and iterative problem solving [11]. By reducing coding complexity, clarifying CT concepts, and reducing pedagogical uncertainty, these supports may help mitigate common barriers to teachers' engagement with text-based coding [6,28]. As these barriers are reduced, teachers are expected to experience gains in coding/CT self-efficacy, teaching self-efficacy, and value beliefs [35,38], which may in turn be associated with improved coding and CT performance and greater readiness for K-12 instruction [4,31].

This framework provides the conceptual foundation for the present study and informs the design of the "Let's Code with GenAI" module (see Appendix A), the research questions, the selection of outcome measures, the interpretation of findings, and the study's implications for teacher PD.

4. Research Questions

This study examines the outcomes of Let's Code with GenAI, a self-paced, non-credit online module designed to support K-12 teachers' learning of coding and CT. Specifically, it addresses four research questions:

RQ1. What pre- and post-changes were observed in participants' teaching efficacy and value beliefs following the intervention?

RQ2. What pre- and post-changes were observed in participants' coding self-efficacy and CT self-efficacy following the intervention?

RQ3. What pre- and post-changes were observed in participants' coding skills and CT performance following the intervention?

RQ4. To what extent were coding self-efficacy, CT self-efficacy, and coding/CT assessment scores interrelated following the intervention?

By examining these interconnected facets, the research seeks to offer preliminary insights into how AI-supported learning tools may relate to teacher preparedness in K-12 coding and CT education and may inform future pedagogical design.

5. Methods

This study adopted an exploratory, practice-oriented design with a simple pretest–posttest structure. Its purpose was to examine initial patterns of change rather than to establish causal effects. Its primary aim was to assess whether the intervention could be implemented as intended and to identify preliminary patterns of change prior to undertaking a more rigorous comparative study. As ref. [56] notes, a feasibility study examines whether a proposed project can be conducted, whether it is appropriate to proceed, and how it should be implemented, whereas a pilot study involves carrying out the intended research, or part of it, on a smaller scale.

5.1. Participants

This study was approved by an Institutional Review Board, and all participants provided informed consent. A convenience sample of 34 pre- and in-service teachers was recruited through professional networks, including former students and colleagues in educational fields, as well as invitations distributed via LinkedIn and email. Recruitment targeted teachers interested in K-12 education. All participants were required to have sufficient English proficiency to understand and respond to the English-language study materials. The participants were recruited between mid-January and mid-April 2025. Table 3 presents demographic information. Participants represented a range of disciplines,

including social studies, English, mathematics, science, computer science, information technology, ethics, AI education, and elementary education.

Table 3. Participant demographics.

Category	Subcategory	Frequency (<i>n</i>)	Percentage (%)
Age (years)	20–29	4	11.8
	30–39	17	50.0
	40–49	9	26.5
	50–59	2	5.9
	60+	2	5.9
Degree	Bachelor	9	26.5
	Master	20	58.8
	Professional	2	5.9
	Doctorate	3	8.8
Gender	Female	16	47.1
	Male	17	50
	Non-binary	1	2.9
Teaching experiences (years)	1–5	8	23.5
	6–10	15	44.1
	11–15	5	14.7
	16–20	5	14.7
	21+	1	2.9

5.2. Instruments

5.2.1. The Teacher Beliefs About Coding and Computational Thinking Scale Test

The Teacher Beliefs about Coding and Computational Thinking (TBaCCT) scale [57] was used to measure participants' beliefs related to coding and CT. The scale was selected because it aligns with the study's focus on teachers' self-efficacy and value beliefs in response to a GenAI-supported text-based coding module. The validity of the scale was established using confirmatory factor analysis and structural equation modeling [57]. In this study, the TBaCCT was administered before and after the learning module to examine pre–post changes in participants' self-efficacy and value beliefs.

5.2.2. Coding/Computational Thinking Assessment

To strengthen the validity of the coding/CT assessment, the items were developed through an explicit alignment process linking each item to the module's learning objectives, instructional activities, and target CT concepts (see Appendix B). The assessment sampled foundational concepts such as variables, sequential reasoning, conditional logic, and loops, with item coverage designed to reflect the progression of content introduced in the module rather than general programming expertise [4]. Two computing-education experts reviewed the item set for content relevance, conceptual alignment, clarity, and appropriateness for the target participants, and their feedback was used to revise the wording and coverage of selected items. In addition, the assessment was informally reviewed by other computer education professionals to determine whether the items adequately represented the intended coding/CT domain. Together, these steps provide preliminary evidence of content validity and alignment for an exploratory measure of module-related learning gains.

5.3. Procedures

The study was delivered via Canvas (Instructure, Inc., Salt Lake City, UT, USA), a web-based learning management system, to provide flexible and accessible learning

materials [58]. Participants who agreed to take part received an email containing a Canvas link to enroll in the course. Participants without existing Canvas accounts were assisted in creating one before beginning the module. The asynchronous format allowed participants to progress at their own pace and revisit the materials as needed, which may support repeated practice and self-regulated learning [55]. Research comparing synchronous and asynchronous online learning also suggests that both formats can support similar outcomes in self-efficacy and academic achievement [59,60].

The learning module, Let's Code with GenAI, began with a pretest that included the TBaCCT scale and the coding/CT assessment. Following the pretest, participants engaged in approximately one hour of self-paced learning focused on text-based coding and CT concepts, including variables, functions, debugging, and algorithmic thinking. The module addressed four CT-related concepts drawn from established CT education standards and organized the content into three levels: basic, intermediate, and advanced [18,19]. The module was built in MakeCode (Microsoft Corporation, Redmond, WA, USA), an online platform used to teach Python with the Micro:bit (Micro:bit Educational Foundation, London, UK) device, which is a pocket-sized programmable hardware platform designed for educational use [61].

Each section of the module included short instructional videos of approximately five minutes each. In addition, a Gemini 1.5 model from Google AI Studio (Google LLC, Mountain View, CA, USA) was integrated into the learning experience. The model was configured to support K-12 Python learning by refining Micro:bit-specific APIs and using prompt engineering techniques to assist with coding tasks [62]. The full GenAI prompt template and operational rules are provided in Appendix C. During the module, the AI model provided explanations and debugging support when participants encountered challenges or unfamiliar CT concepts. This design allowed participants to consult the model while working through the activities, offering on-demand support within the learning process.

This integration of GenAI may have enabled participants to interact with the AI model during their learning, providing real-time support and fostering engagement. By supporting participants as they addressed coding errors and clarified CT concepts through consultation with the AI model, learners were able to overcome obstacles more effectively and deepen their understanding of coding and CT (see Figure 2).

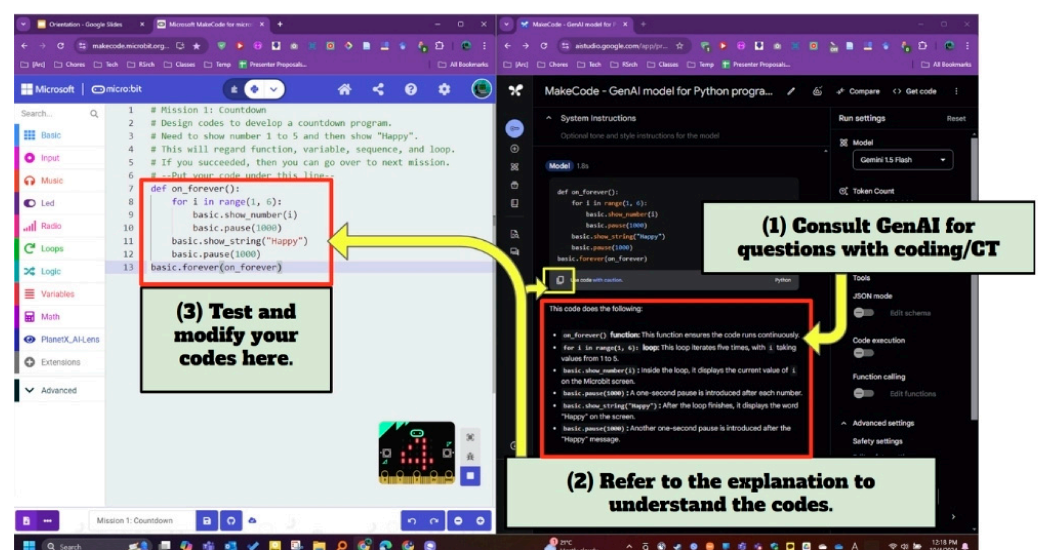


Figure 2. A scene of consulting a GenAI model to solve problems.

From a methodological standpoint, this design offers distinct upsides over alternative teacher PD structures (see Table 2 for a comparison with other approaches). While traditional synchronous or mentorship-based CT training provides human guidance, it faces severe resource constraints and instructional bottlenecks during text-based coding [29,31]. Standard asynchronous environments likewise lack interactive feedback, exacerbating learner frustration and low self-efficacy [6]. In contrast, the proposed framework bridges this gap by embedding an on-demand generative scaffold within a flexible, asynchronous framework. As noted in emerging literature, this integration leverages GenAI's capacity for immediate, contextualized debugging and conceptual support [7,8]. Consequently, this approach may support self-regulated, practice-embedded learning experiences without requiring resource-intensive, real-time human intervention.

After completing the learning module, participants completed a posttest to measure changes in their self-efficacy and beliefs regarding coding and CT concepts. This posttest included the TBaCCT scale and the coding/CT assessment, enabling researchers to analyze any differences between pre- and posttest results.

5.4. Data Analysis

Quantitative analyses were conducted to examine pre-post differences across the four research questions (see Table 4). A paired-samples *t*-test was used for variables that met the assumption of normality, as this test is appropriate for comparing mean differences within the same group across two time points. For variables that did not meet the assumption of normality, the Wilcoxon signed-rank test was used as a nonparametric alternative. In addition, correlation analyses were conducted to examine the relationships among selected variables after the intervention.

Table 4. Research questions, data sources, and data analysis methods.

Research Questions	Data Sources	Data Analysis Methods
RQ1. What pre- and post-changes were observed in participants' teaching efficacy and value beliefs following the intervention?	TBaCCT scale test (self-report survey)	Paired-samples <i>t</i> -test
RQ2. What pre- and post-changes were observed in participants' coding self-efficacy and CT self-efficacy following the intervention?	TBaCCT scale test (self-report survey)	Wilcoxon signed-rank test
RQ3. What pre- and post-changes were observed in participants' coding skills and CT performance following the intervention?	Coding/CT assessment	Wilcoxon signed-rank test
RQ4. To what extent were coding self-efficacy, CT self-efficacy, and coding/CT assessment scores interrelated following the intervention?	(1) TBaCCT scale test (self-report survey) (2) Coding/CT assessment	Correlation analysis

5.5. Descriptive Statistics

Descriptive statistics were calculated for all primary variables at the pre- and post-intervention time points (see Table 5). For the 34 participants, each variable was summarized using the minimum, maximum, mean, and standard deviation. Overall, all survey-based and test-based measures demonstrated directional increases in their mean scores from pretest to posttest, with coding self-efficacy showing the most substantial descriptive gain.

Table 5. Descriptive statistics for measures.

Type	Variables (Measures) *	Time Point	N	Min	Max	Mean	SD
Survey-based	Coding self-efficacy (8 items)	Pre	34	1.00	6.00	3.5515	1.39572
		Post	34	2.00	6.00	4.7794	0.83941
	CT self-efficacy (4 items)	Pre	34	2.25	6.00	4.2721	1.01199
		Post	34	2.25	6.00	4.5809	1.11076
	Teaching efficacy (11 items)	Pre	34	1.00	6.00	3.5802	1.27901
		Post	34	2.36	6.00	4.6016	0.96262
	Value beliefs (10 items)	Pre	34	3.30	6.00	4.5618	0.68844
		Post	34	3.10	6.00	4.8912	0.80429
Test-based	Coding/CT assessment scores (5 items)	Pre	34	1.00	5.00	3.8824	1.03762
		Post	34	2.00	5.00	4.2059	0.94643

* Each variable represents the average score of the number of items indicated in parentheses.

5.6. Reliability Check

Internal consistency reliability for each subscale of the TBaCCT instrument was assessed using Cronbach's alpha, calculated separately for pre- and posttest responses to reflect the multidimensional structure of the scale. The results indicated excellent reliability for coding self-efficacy (pre $\alpha = 0.966$, post $\alpha = 0.940$, 8 items), CT self-efficacy (pre $\alpha = 0.784$, post $\alpha = 0.803$, 4 items), and teaching efficacy (pre $\alpha = 0.961$, post $\alpha = 0.954$, 11 items). Value beliefs also showed acceptable to good reliability (pre $\alpha = 0.792$, post $\alpha = 0.872$, 10 items), indicating that all subscales demonstrated adequate internal consistency for further analysis.

5.7. Assessment of Normality Assumption

The Shapiro–Wilk test was used to examine whether the difference scores met the assumption of normality (see Table 6). Teaching efficacy and value beliefs met the normality assumption and were analyzed using the paired-samples *t*-test. In contrast, coding self-efficacy, CT self-efficacy, and coding/CT assessment scores did not meet the assumption of normality and were therefore analyzed using the Wilcoxon signed-rank test.

Table 6. Normality tests for the difference in pre- and post-intervention scores.

Variable	Shapiro–Wilk		
	Statistic	df	Sig.
Coding self-efficacy	0.885	34	0.002
CT self-efficacy	0.889	34	0.002
Teaching efficacy	0.946	34	0.092
Value beliefs	0.971	34	0.476
Coding/CT assessment score	0.828	34	<0.001

Outliers were retained in the dataset (see Figure 3) because there was no evidence that they resulted from measurement errors or data-entry mistakes. Their inclusion allowed the

analysis to preserve the full variability of the data. In addition, the use of non-parametric tests for variables that violated the normality assumption reduced the influence of outliers on the analysis.

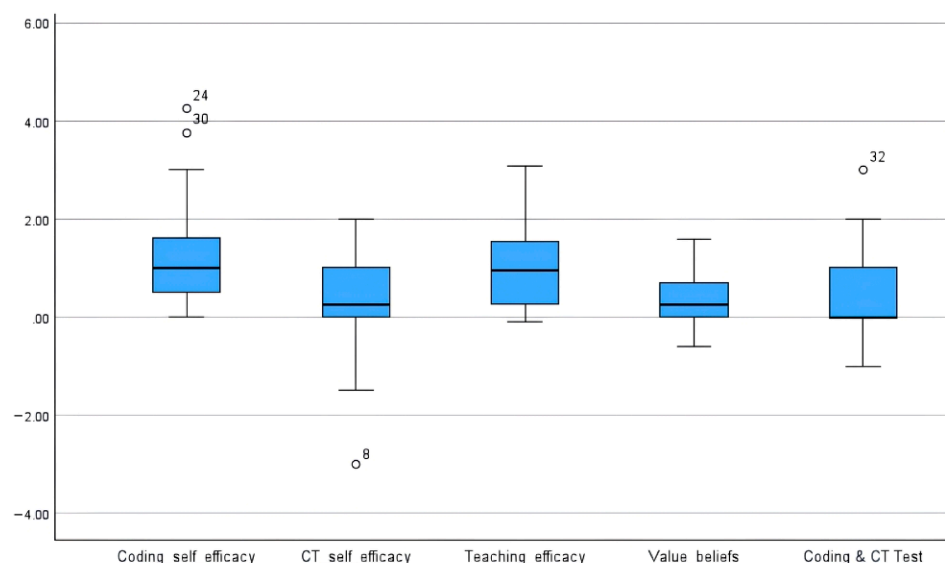


Figure 3. Box plot showing retained outliers in difference scores.

6. Results

The results present pre–post differences in participants’ teaching efficacy, value beliefs, coding self-efficacy, CT self-efficacy, and coding/CT assessment scores, as well as relationships among these measures. Statistical analyses included the paired-samples *t*-test, Wilcoxon signed-rank tests, and correlation analyses.

6.1. Observed Changes in Teacher Efficacy and Value Beliefs

A paired-samples *t*-test was conducted using participants’ pre- and posttest scores on the teaching efficacy and value beliefs subscales of the TBaCCT scale. The results showed statistically significant increases in teaching efficacy ($t = 7.653, p < 0.001$) and value beliefs ($t = 3.730, p < 0.001$) following participation in the module (See Table 7).

Table 7. Paired-samples *t*-tests for teaching efficacy and value beliefs.

Variable	Paired Differences					Significance			
	Mean	SD	Std Error Mean	95% Confidence Interval Difference		<i>t</i>	df	One-Sided <i>p</i>	Two-Sided <i>p</i>
				Lower	Upper				
Teaching efficacy	1.02138	0.77820	0.13346	0.74985	1.29290	7.653	33	<0.001	<0.001
Value beliefs	0.32941	0.51495	0.08831	0.14974	0.50908	3.730	33	<0.001	<0.001

The effect sizes, as measured by Cohen’s *d*, were 1.312 for teaching efficacy and 0.640 for value beliefs, indicating large and medium effects, respectively (See Table 8). Notably, the lower bound of the 95% confidence interval for teaching efficacy ($d = 0.846$) remained well within the large effect range, whereas the interval for value beliefs indicated a more moderate shift. Hedges’ correction estimates further confirmed these robust magnitude patterns across both measures.

Table 8. Paired-samples *t*-test effect sizes for teaching efficacy and value beliefs.

Variable	Effect Size	Standardizer *	Point Estimate	95% Confidence Interval	
				Lower	Upper
Teaching efficacy	Cohen's <i>d</i>	0.77820	1.312	0.846	1.768
	Hedges' correction	0.79646	1.282	0.827	1.727
Value beliefs	Cohen's <i>d</i>	0.51495	0.640	0.266	1.005
	Hedges' correction	0.52703	0.625	0.260	0.982

* The denominator used in estimating the effect sizes. Cohen's *d* uses the sample standard deviation of the mean difference. Hedges' correction uses the sample standard deviation of the mean difference, plus a correction factor.

6.2. Observed Changes in Coding and CT Self-Efficacy

Wilcoxon signed-rank tests were conducted using participants' pre- and posttest scores on the coding self-efficacy and CT self-efficacy subscales of the TBaCCT scale. The results indicated statistically significant pre–post increases in both coding self-efficacy and CT self-efficacy (see Tables 9 and 10).

Table 9. Wilcoxon signed-rank test result of coding self-efficacy.

Category	N	Mean Rank	Sum of Ranks	Z ⁴	<i>p</i> (2-Tailed)	<i>r</i> (Effect Size) *
Negative ranks ¹	0	0.00	0.00	-	-	-
Positive ranks ²	32	16.50	528.00	-	-	-
Ties ³	2	-	-	-	-	-
Total	34	-	-	−4.939	<0.001	−0.847

¹ Post_coding self-efficacy < Pre_coding self-efficacy; ² Post_coding self-efficacy > Pre_coding self-efficacy; ³ Post_coding self-efficacy = Pre_coding self-efficacy; ⁴ Based on negative ranks; * The negative sign of Cohen's *r* indicates the direction of ranking, signifying an increase in coding self-efficacy scores.

Table 10. Wilcoxon signed-rank test result of CT self-efficacy.

Category	N	Mean Rank	Sum of Ranks	Z ⁴	<i>p</i> (2-Tailed)	<i>r</i> (Effect Size) *
Negative ranks ¹	6	12.00	72.00	-	-	-
Positive ranks ²	19	13.32	253.00	-	-	-
Ties ³	9	-	-	-	-	-
Total	34	-	-	−2.443	0.015	−0.419

¹ Post_CT self-efficacy < Pre_CT self-efficacy; ² Post_CT self-efficacy > Pre_CT self-efficacy; ³ Post_CT self-efficacy = Pre_CT self-efficacy; ⁴ Based on negative ranks; * The negative sign of Cohen's *r* indicates the direction of ranking, signifying an increase in CT Self-efficacy scores.

Coding self-efficacy increased from pretest to posttest ($Z = -4.939$, $p < 0.001$), with 32 of 34 participants showing gains. The effect size was large ($r = -0.847$). Crucially, the rank distribution in Table 9 highlights a uniform upward shift, with a vast majority of the sample exhibiting positive ranks (32 participants, 94.1%), while the remaining showed ties (2 participants, 5.9%) and zero negative ranks (0 participants, 0.0%) were recorded.

CT self-efficacy also improved significantly ($Z = -2.443$, $p = 0.015$), with 19 participants showing higher posttest scores and a moderate effect size ($r = -0.419$). In contrast to coding self-efficacy, the ranks for CT self-efficacy (Table 10) were more widely distributed across the sample; while around half achieved positive ranks (19 participants, 55.9%), a notable portion of the sample exhibited either ties (9 participants, 26.5%) or negative ranks (6 participants, 17.6%), accounting for the relatively lower magnitude of the resulting effect size.

6.3. Observed Changes in Coding/CT Assessment Scores

Wilcoxon signed-rank tests were conducted to compare participants' pre- and posttest scores on the coding/CT assessment. The results indicated a statistically significant pre-post increase in assessment scores ($Z = -2.054$, $p = 0.040$), with 11 of 34 participants showing higher posttest scores (see Table 11). The effect size was moderate ($r = -0.352$). From a structural standpoint, the rank distribution in Table 11 reveals that more than half of the sample exhibited ties (19 participants, 55.9%), indicating no change in their pre- and post-intervention test scores. Among the remaining participants, around one-third demonstrated positive ranks (11 participants, 32.4%), while a small minority showed negative ranks (4 participants, 11.8%). This unique structural distribution, particularly the high prevalence of tied scores, accounts for the more conservative moderate effect size observed in contrast to the affective self-efficacy measures.

Table 11. Wilcoxon signed-rank test result of coding/CT assessment score.

Category	N	Mean Rank	Sum of Ranks	Z ⁴	p (2-Tailed)	r (Effect Size) *
Negative ranks ¹	4	6.50	26.00	-	-	-
Positive ranks ²	11	8.55	94.00	-	-	-
Ties ³	19	-	-	-	-	-
Total	34	-	-	-2.054	0.040	-0.352

¹ Post_coding/CT assessment score < Pre_coding/CT assessment score; ² Post_coding/CT assessment score > Pre_coding/CT assessment score; ³ Post_coding/CT assessment score = Pre_coding/CT assessment score; ⁴ Based on negative ranks; * The negative sign of Cohen's r indicates the direction of ranking, signifying an increase in coding/CT assessment scores.

6.4. Post-Intervention Relationships Among Measures

Correlation analyses were conducted using participants' post-intervention coding self-efficacy scores, CT self-efficacy scores, and coding/CT assessment scores to examine the interrelationships among affective and performance measures (see Table 12). Interestingly, the resulting correlation matrix in Table 12 reveals an entirely decoupled pattern between the two subjective self-efficacy constructs and the objective, test-based performance measure, with both Pearson correlation coefficients trending near zero and failing to reach statistical significance ($r = -0.038$, $p = 0.832$ for coding self-efficacy and assessment scores; $r = -0.053$, $p = 0.767$ for CT self-efficacy and assessment scores). In contrast, the relationship between the two affective measures—coding self-efficacy and CT self-efficacy—demonstrated a moderate positive directional trend ($r = 0.333$) that closely approached the empirical threshold of significance ($p = 0.054$), although it ultimately remained statistically non-significant under the conventional criterion of $\alpha = 0.05$.

Table 12. Correlation analysis of coding self-efficacy, CT self-efficacy, and coding/CT assessment scores.

	Statistic	Coding Self-Efficacy	CT Self-Efficacy	Coding/CT Assessment Scores
Coding self-efficacy	r	1	0.333	-0.038
	p (2-tailed)	-	0.054	0.832
	N	34	34	34
CT self-efficacy	r	0.333	1	-0.053
	p (2-tailed)	0.054	-	0.767
	N	34	34	34
Coding/CT assessment scores	r	-0.038	-0.053	1
	p (2-tailed)	0.832	0.767	-
	N	34	34	34

7. Discussions

This study reports preliminary pre–post differences observed among participants who completed a GenAI-supported PD module for K-12 teachers.

7.1. *Interpreting Shifts in Teacher Self-Efficacy and Value Beliefs (RQ1, RQ2)*

Across the TBaCCT subscales, participants showed higher posttest scores in coding self-efficacy, CT self-efficacy, teaching efficacy, and value beliefs. These positive trends align with prior literature establishing that structured PD for CT is often associated with increases in educator confidence [28,31]. Specifically, these patterns corroborate prior findings that interactive modules offering timely explanations, individualized support, and self-paced tasks can foster positive psychological outcomes [45,63]. In teacher PD, fostering such self-efficacy is a critical prerequisite; an educator’s belief in their own capability plays a central role in their persistence when facing technical setbacks and their ultimate willingness to adopt unfamiliar technologies in the classroom [6,35]. Therefore, tracking these psychological dimensions provides vital insight into whether a module can lower the initial dispositional barriers that historically hindered curriculum integration [5]. Crucially, while traditional frameworks suggest that such affective shifts are primarily observed within intensive, human-led coaching or long-term design communities [29,34], our exploratory pilot highlights a preliminary connection between an automated, GenAI-supported learning environment and similar positive shifts in teacher beliefs.

This on-demand scaffolding—focusing primarily on real-time debugging and troubleshooting assistance—may have played a supportive role in minimizing the technical uncertainties that typically challenge teachers in conventional asynchronous formats [6]. Furthermore, the observed increase in value beliefs mirrors broader educational frameworks within which early mastery-like experiences in training often relate to a teacher’s openness to future classroom implementation [31,38]. While the present single-group design does not allow causal attribution to the GenAI component alone, these exploratory data provide empirical baseline evidence that supports emerging literature on the role of interactive AI scaffolds in lowering early dispositional barriers to text-based coding [7,8].

7.2. *Exploring the Association Between Self-Efficacy and Performance Gains (RQ3, RQ4)*

Although both self-efficacy and performance scores increased, posttest correlation analyses showed no statistically significant correlations among these measures, indicating that affective confidence and conceptual skill acquisition did not necessarily develop in tandem. This divergence directly corroborates the foundational perspective that self-efficacy reflects perceived capability under specific conditions rather than objective performance outcomes [35]. In the context of technology-mediated PD, this pattern highlights a critical phenomenon: immediate automated feedback can relate to initial increases in learners’ comfort levels, yet actual text-based coding competence typically requires longitudinal, repeated application to fully solidify [21].

This descriptive mismatch presents an interesting contrast to some traditional, instructor-led CT interventions in which skill gains and confidence often exhibit parallel trajectories due to real-time human monitoring and iterative correction [28,31]. In our self-paced module, the high number of tied performance scores suggests that, although the GenAI scaffold may have reduced initial anxiety and increased motivation, the short duration of the pilot was not sufficient for participants to internalize complex syntax and computational thinking logic. Consequently, these findings suggest that GenAI may support prompt confidence-building and reduce early bottlenecks, but it should be evaluated within a broader, iterative learning sequence that includes extended practice, reflective code-writing, and delayed post-testing to capture genuine competence transfer [21,35].

7.3. Theoretical and Practical Contributions

To clarify the broader impact of this exploratory pilot, these baseline trends offer distinct contributions that extend existing computing education and PD frameworks. First, this study contributes a novel theoretical perspective on scaffolding by positioning an automated, GenAI-supported environment as an on-demand alternative for real-time human expert mediation [7,8]. While prior literature frames text-based computing PD as an inherently resource-intensive endeavor [29,34], our findings suggest the viability of low-resource, asynchronous frameworks to support early affective readiness. Second, by empirically capturing the early decoupling between initial confidence building and delayed conceptual mastery [21,35], this study provides a vital architectural caveat to the AIED field. It theoretically underscores that reducing psychological bottlenecks does not automatically parallel immediate cognitive internalization. Ultimately, these insights provide a sustainable empirical and practical foundation for researchers and district leaders seeking to design multi-layered, scalable learning sequences that balance affective support with rigorous skill transfer.

8. Limitations and Future Research

This study has several limitations that should be considered when interpreting the findings. First, reliance on convenience sampling, the English-language proficiency requirement, and the absence of a control group limit the generalizability of the results beyond the current participant pool. Due to this single-group structure, observed changes cannot be definitively isolated from confounding factors such as the self-paced format of the module, which may have influenced participants' pacing and engagement, as well as the short duration of the training, participants' initial motivation levels, prior knowledge, maturation, or testing effects. Second, the relatively small sample size may have reduced the statistical power needed to detect meaningful correlations between self-efficacy and performance gains. Third, participants' interactions with the GenAI tools could not be fully standardized; the frequency, type, and depth of prompt usage likely varied across individuals. These variations may have shaped how participants experienced the GenAI-supported module and, in turn, influenced their learning processes and outcomes. Fourth, the study examined only immediate pre- and post-intervention changes, providing no evidence regarding the long-term retention of coding and CT skills. Finally, although the five-item assessment aligned with the module's objectives, it may not have been sensitive enough to capture nuanced learning gains, which could explain the high number of tied scores.

To address these limitations, future research should employ larger, more diverse samples and incorporate control groups to better isolate the unique contributions of GenAI-supported learning relative to other instructional approaches. Longitudinal follow-up studies are also necessary to determine whether improvements in self-efficacy, value beliefs, and coding/CT performance are sustained over time. Crucially, future evaluations should assess whether teachers can successfully transfer these skills into authentic classroom practices and independently design coding activities for their students. In addition, follow-up studies should incorporate usage tracking and participant surveys to investigate how different GenAI models, prompt-design strategies, and levels of scaffolded support influence teachers' learning experiences and outcomes in text-based coding and CT [62].

9. Conclusions

This exploratory study found that participation in the GenAI-supported, self-paced online module, Let's Code with GenAI, was associated with higher posttest scores in coding self-efficacy, CT self-efficacy, teaching efficacy, value beliefs, and text-based coding/CT performance among K-12 teachers. These findings point to the potential value of AI-supported

learning experiences for teacher learning, although the study design does not allow the effects of GenAI to be separated from other module features, such as its self-paced and asynchronous format. The lack of correlations among self-efficacy measures and performance scores suggests that confidence and skill development may not always progress in tandem. Given the small sample size, the limited sensitivity of the coding/CT assessment, and the large number of unchanged scores, the observed performance gains should be viewed as preliminary rather than as evidence of substantial skill development. Moreover, future PD programs may benefit from combining confidence-building activities, such as on-demand GenAI scaffolding, with authentic coding tasks, iterative feedback, and learning pathways responsive to teachers' prior knowledge. Embedding these supports within sustained professional learning contexts may help strengthen both teacher confidence and coding/CT competence over time.

Author Contributions: Study conception and design, S.K. and W.H.; material preparation, data collection and analysis, S.K. and W.H.; writing—original draft preparation, S.K.; writing—review and editing, all authors; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: The study was approved by the Institutional Review Board of Purdue University (IRB-2024-1733 dated 31 October 2024).

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

Data Availability Statement: The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request.

Acknowledgments: We would like to thank the participants for contributing to this study.

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

Abbreviations

The following abbreviations are used in this manuscript:

CT	Computational thinking
PD	Professional development
TBaCCT	Teacher Beliefs about Coding and Computational Thinking
GenAI	Generative Artificial Intelligence

Appendix A

Let's Code with GenAI (Online Text-Based Coding Learning Module with GenAI Facilitation)

1. Lesson objectives: Participants will be able to write and run simple codes with the assistance of GenAI by using six fundamental coding and computational thinking (CT) concepts, including (1) functions, (2) debugging, (3) variables, and three algorithmic statements: (4) sequence, (5) loop, and (6) condition.
2. Target audience: K-12 teachers with basic digital literacy skills, including the ability to access the required LMS system. They are not expected to have any programming skills.
3. Materials and resources: Laptop (with Chrome web browser), GenAI prompt (with Micro:bit API for MakeCode Platform), Google Account
4. Session flow (50–60 min in total)
 - a. Pretest (10 min):
 - i. Participants take part in the pretest for (10 min for 38 items in total)

1. Coding/CT assessment (5 items)
 - a. Variable
 - b. Sequential statement
 - c. Conditional statement
 - d. Loop statement
 - e. Combination of variable, conditional statement, and loop statement
 2. Teacher Beliefs about Coding and Computational Thinking (TBaCCT) (33 items)
 - a. Section 1: Coding self-efficacy (8 items)
 - b. Section 2: CT self-efficacy (4 items)
 - c. Section 3: Teaching efficacy (11 items)
 - d. Section 4: Value beliefs (10 items)
- b. Main activity (30–40 min):
- i. Computational concept exploration (10 min): Use Canvas (LMS)/MakeCode to introduce and demonstrate basic coding concepts with corresponding scenarios according to the degree of difficulty:
 1. Function: A named block of code that performs a specific task (e.g., show me number 1). MakeCode's Micro:bit requires basic.forever (_function_slot_) to continuously execute inserted functions, which must be understood before proceeding.
 2. Debugging: The process of identifying and removing errors, or bugs, from a computer program. Left alignment and typos are the primary issues that GenAI can help address.
 3. Variables: Storage for data (e.g., x = "Earth", change x to "Moon.")
 4. Sequential statements: Letting instructions happen one after the other, in order (e.g., say "hi"—wait 1 s and then say "bye.")
 5. Loop statements: Repeat a set of instructions (e.g., play a short sound effect, 3 times).
 6. Conditional statements: Making choices (e.g., If variable x reaches 5, say "hello!")
 - ii. Problem-solving with Code Creation activity (10 min): Participants are asked to solve Missions by consulting GenAI without seeking help from human instructors.
 1. Mission 1: Design codes to develop a countdown program
 - a. Objective: Participants design a fully functional program that incorporates concepts related to function, variable, sequential statement, and loop statement.
 - b. Description: Participants need to design codes that show numbers 1 to 5 and then show the word "Happy" on the screen. This activity is designed to help learners understand how functions, variables, sequences, and loop statements work. If they succeed, they can proceed to the next mission.
 2. Mission 2: Create a program that responds to tilting action
 - a. Objective: Participants design a fully functional program that incorporates functions, variables, sequential statements, and conditional statements.

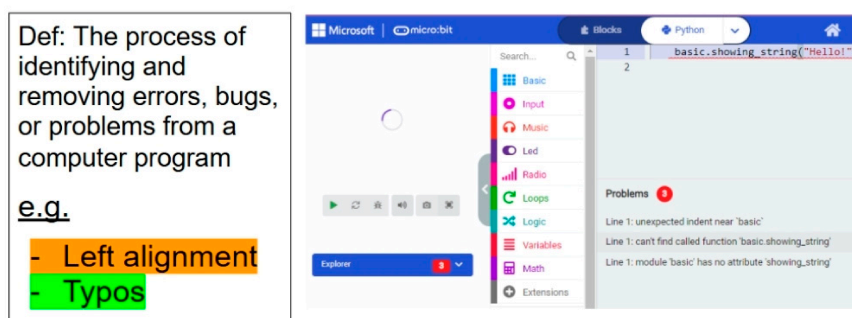


Figure A2. Example of debugging code with GenAI support.

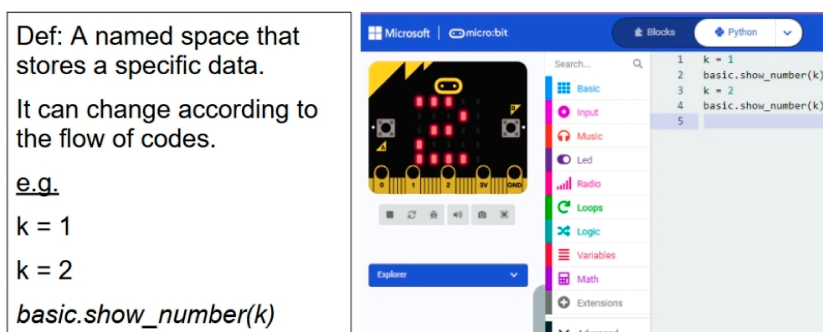


Figure A3. Example of variable initialization and modification.

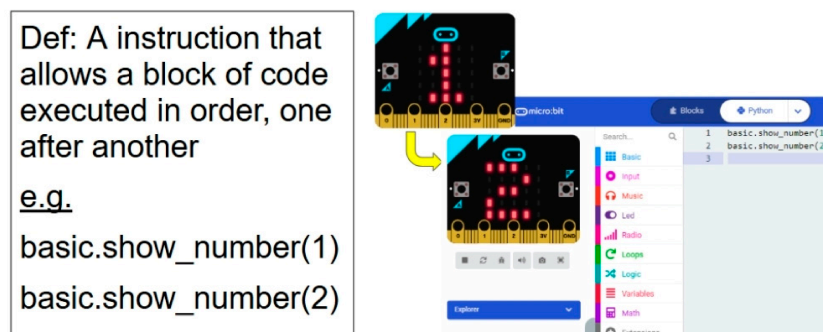


Figure A4. Example of sequential code execution flow.

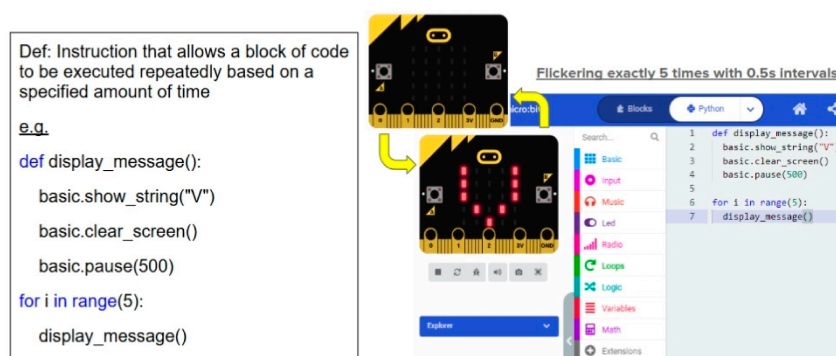


Figure A5. Example of loop statements repeating instruction blocks.

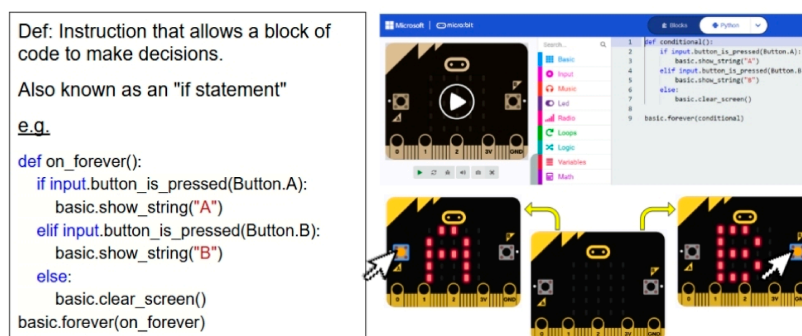


Figure A6. Example of conditional statements for decision-making.

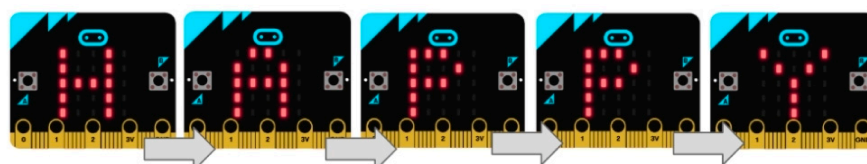


Figure A7. Example of Micro:bit hardware display for Mission 1.

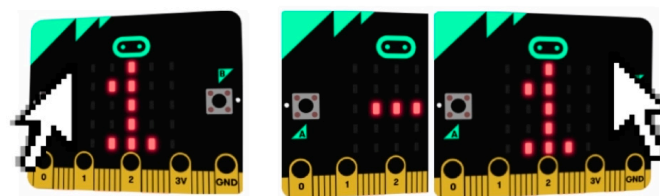


Figure A8. Example of visual response for tilting action in Mission 2.

Appendix B

Coding and Computational Thinking Assessment

Q1. Variable

- focus: understanding of variables as named storage locations in a program

Box A1. Assessment item on variable concept.

A variable is a named location in a program that stores a value. Once a variable is defined, its value remains fixed throughout the program's execution.

[] True [] False

Q2. Sequential statement

- focus: understanding of the execution order of sequential statements

Box A2. Assessment item on sequential statement concept.

In the following code, print("First line") executes before print("Second line").

```
print("Second line")
print("First line")
```

[] True [] False

Q3. Conditional statement

- focus: understanding of conditional logic and program output

Box A3. Assessment item on conditional statement concept.

What message will the following code print as output?

```
if 10 < 5:
    print("It is cold")
else:
    print("It is hot")
```

☐ (No output)	☐ It is cold
☐ It is hot	☐ It is cold It is hot

Q4. Loop statement

- focus: understanding of repeated execution in a loop structure

Box A4. Assessment item on loop statement concept.

```
for i in range(3):
    print("Hello")
```

☐ 1 ☐ 2 ☐ 3 ☐ 4

Q5. Integrated Concepts

- focus: understanding of the integrated use of variables, conditional logic, and loops

Box A5. Assessment item on integrated CT concepts.

What will be printed as output by the following code?

```
total = 0
for number in range(5):
    if number > 2:
        total += number
print(total)
```

☐ 5 ☐ 7 ☐ 9 ☐ 11

Appendix C

GenAI Prompt Template and Operational Rules

1. **Scope of Use:** This scope was included to keep the model focused on the exact learning domain and to prevent irrelevant or off-topic responses.
 - Respond only to questions related to MakeCode Python programming for Micro:bit.
 - Stay within the scope of the provided coding materials and learning tasks.
 - If the requested content is not included in the provided materials, state that no relevant information is available.
2. **API-Based Context:** This API-based constraint was included to ensure that the AI responses matched the actual programming environment and remained technically reliable.
 - Use only the supplied MakeCode API context, including selected commands from basic, input, music, led, and radio.
 - Do not introduce unsupported parameters, undocumented syntax, or external Python functionality.
 - Keep responses aligned with the programming environment used in the module.
3. **Response Constraints:** These response constraints were included to reduce unsupported or speculative outputs and to keep the AI support instructionally appropriate.
 - Do not import additional modules.

- Do not introduce general Python features not supported in the provided MakeCode context.
 - Do not mention functions, parameters, or programming features not included in the supplied API material.
 - Keep example code simple unless more complexity is explicitly requested.
 - If the topic is outside the available API scope, do not speculate.
4. **Code-Generation Rules:** These code-generation rules were included to promote code validity, avoid unnecessary complexity, and improve consistency across examples.
 - Generate only MakeCode-compatible Python code for Micro:bit.
 - Avoid invalid enum operations.
 - Avoid overly complex structures unless explicitly requested.
 - Provide no more than two examples unless more are specifically requested.
 - Prefer examples that work in the MakeCode emulator whenever possible.
 5. **Explanatory Rules:** These explanatory rules were included to make the generated examples easier to understand and to preserve the documented behavior of specific MakeCode events.
 - Place comments above any example code.
 - Explain the logic of each example concisely after the code.
 - For logo pressed or long-pressed events, note that the effect occurs when the logo is released.
 6. **Instructional Purpose:** This overarching purpose was included to frame the AI as a bounded instructional scaffold rather than a free-form code generator.
 - These rules were intended to constrain the model's behavior, ensure consistency with the target programming environment, and improve reproducibility of the GenAI-supported learning experience.
 - The GenAI tool functioned as a structured instructional aid for explanation and debugging within a bounded API context rather than as an unrestricted code-generation system.

References

1. Kafai, Y.B.; Proctor, C. A Revaluation of Computational Thinking in K-12 Education: Moving toward Computational Literacies. *Educ. Res.* **2022**, *51*, 146–151. [\[CrossRef\]](#)
2. Angeli, C.; Giannakos, M. Computational Thinking Education: Issues and Challenges. *Comput. Hum. Behav.* **2020**, *105*, 106185. [\[CrossRef\]](#)
3. Caeli, E.N.; Yadav, A. Unplugged Approaches to Computational Thinking: A Historical Perspective. *TechTrends* **2020**, *64*, 29–36. [\[CrossRef\]](#)
4. Weintrop, D.; Wilensky, U. Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Trans. Comput. Educ.* **2017**, *18*, 1–25. [\[CrossRef\]](#)
5. Sun, L.; Zhou, L. Does Text-Based Programming Improve K-12 Students' CT Skills? Evidence from a Meta-Analysis and Synthesis of Qualitative Data in Educational Contexts. *Think. Ski. Creat.* **2023**, *49*, 101340. [\[CrossRef\]](#)
6. Mason, S.L.; Rich, P.J. Preparing Elementary School Teachers to Teach Computing, Coding, and Computational Thinking. *Contemp. Issues Technol. Teach. Educ.* **2019**, *19*, 790–824.
7. Idrisov, B.; Schlippe, T. Program Code Generation with Generative AIs. *Algorithms* **2024**, *17*, 62. [\[CrossRef\]](#)
8. Pesovski, I.; Santos, R.; Henriques, R.; Trajkovik, V. Generative AI for Customizable Learning Experiences. *Sustainability* **2024**, *16*, 3034. [\[CrossRef\]](#)
9. Zafari, M.; Bazargani, J.S.; Sadeghi-Niaraki, A.; Choi, S.-M. Artificial Intelligence Applications in K-12 Education: A Systematic Literature Review. *IEEE Access* **2022**, *10*, 61905–61921. [\[CrossRef\]](#)
10. Moore, R.L.; Jiang, S.; Abramowitz, B. What Would the Matrix Do? A Systematic Review of K-12 AI Learning Contexts and Learner-Interface Interactions. *J. Res. Technol. Educ.* **2023**, *55*, 7–20. [\[CrossRef\]](#)
11. Papert, S. *Mindstorms: Children, Computers, and Powerful Ideas*; Basic Books: New York, NY, USA, 1980; ISBN 978-0-465-04627-0.

12. Lockwood, J.; Mooney, A. Computational Thinking in Education: Where Does It Fit? *Int. J. Comput. Sci. Educ. Sch.* **2017**, *2*, 1–17. [\[CrossRef\]](#)
13. Hsiao, J.H. Understanding Human Cognition through Computational Modeling. *Top. Cogn. Sci.* **2024**, *16*, 349–376. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Aho, A.V. Computation and Computational Thinking. *Comput. J.* **2012**, *55*, 832–835. [\[CrossRef\]](#)
15. Wing, J.M. Computational Thinking. *Commun. ACM* **2006**, *49*, 33–35. [\[CrossRef\]](#)
16. Wing, J. Computational Thinking's Influence on Research and Education for All. *Ital. J. Educ. Technol.* **2017**, *25*, 7–14. [\[CrossRef\]](#)
17. Grover, S.; Pea, R. Computational Thinking in k-12: A Review of the State of the Field. *Educ. Res.* **2013**, *42*, 38–43. [\[CrossRef\]](#)
18. Voogt, J.; Fisser, P.; Good, J.; Mishra, P.; Yadav, A. Computational Thinking in Compulsory Education: Towards an Agenda for Research and Practice. *Educ. Inf. Technol.* **2015**, *20*, 715–728. [\[CrossRef\]](#)
19. Yeni, S.; Grgurina, N.; Saeli, M.; Hermans, F.; Tolboom, J.; Barendsen, E. Interdisciplinary Integration of Computational Thinking in K-12 Education: A Systematic Review. *Inform. Educ.* **2024**, *23*, 223–278. [\[CrossRef\]](#)
20. Payne, L.; Tawfik, A.; Olney, A.M. Computational Thinking in Education: Past and Present. *TechTrends* **2022**, *66*, 745–747. [\[CrossRef\]](#)
21. Yilmaz, R.; Karaoglan Yilmaz, F.G. The Effect of Generative Artificial Intelligence (AI)-Based Tool Use on Students' Computational Thinking Skills, Programming Self-Efficacy and Motivation. *Comput. Educ. Artif. Intell.* **2023**, *4*, 100147. [\[CrossRef\]](#)
22. Nouri, J.; Zhang, L.; Mannila, L.; Norén, E. Development of Computational Thinking, Digital Competence and 21 Century Skills When Learning Programming in K-9. *Educ. Inq.* **2020**, *11*, 1–17. [\[CrossRef\]](#)
23. Caeli, E.N.; Bundsgaard, J. Computational Thinking in Compulsory Education: A Survey Study on Initiatives and Conceptions. *Educ. Technol. Res. Dev.* **2020**, *68*, 551–573. [\[CrossRef\]](#)
24. Erümit, A.K.; Sahin, G. Plugged or Unplugged Teaching: A Case Study of Students' Preferences for the Teaching Programming. *Int. J. Comput. Sci. Educ. Sch.* **2020**, *4*, 3–32. [\[CrossRef\]](#)
25. Curzon, P.; McOwan, P.W.; Plant, N.; Meagher, L.R. Introducing Teachers to Computational Thinking Using Unplugged Storytelling. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*; ACM: Berlin, Germany, 2014; pp. 89–92.
26. Kirçali, A.Ç.; Özden, N. A Comparison of Plugged and Unplugged Tools in Teaching Algorithms at the K-12 Level for Computational Thinking Skills. *Technol. Knowl. Learn.* **2023**, *28*, 1485–1513. [\[CrossRef\]](#)
27. Sentance, S.; Csizmadia, A. Computing in the Curriculum: Challenges and Strategies from a Teacher's Perspective. *Educ. Inf. Technol.* **2017**, *22*, 469–495. [\[CrossRef\]](#)
28. Pears, A.; Barendsen, E.; Dagienė, V.; Dolgopolas, V.; Jasutė, E. Holistic STEAM Education through Computational Thinking: A Perspective on Training Future Teachers. In *Informatics in Schools. New Ideas in School Informatics*; Pozdniakov, S.N., Dagienė, V., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2019; Volume 11913, pp. 41–52. ISBN 978-3-030-33758-2.
29. Chookhampaeng, C.; Kamha, C.; Chookhampaeng, S. Problems and Needs Assessment to Learning Management of Computational Thinking of Teachers at the Lower Secondary Level. *J. Curric. Teach.* **2023**, *12*, 172. [\[CrossRef\]](#)
30. Saidin, N.D.; Khalid, F.; Martin, R.; Kuppasamy, Y.; Munusamy, N.A. Benefits and Challenges of Applying Computational Thinking in Education. *Int. J. Inf. Educ. Technol.* **2021**, *11*, 248–254. [\[CrossRef\]](#)
31. Ghani, A.; Griffiths, D.; Salha, S.; Affouneh, S.; Khalili, F.; Khlaif, Z.N.; Burgos, D. Developing Teaching Practice in Computational Thinking in Palestine. *Front. Psychol.* **2022**, *13*, 870090. [\[CrossRef\]](#) [\[PubMed\]](#)
32. Kale, U.; Akcaoglu, M.; Cullen, T.; Goh, D.; Devine, L.; Calvert, N.; Grise, K. Computational What? Relating Computational Thinking to Teaching. *TechTrends* **2018**, *62*, 574–584. [\[CrossRef\]](#)
33. Sabo, H.C.; Odden, T.O.B.; Gregers, T.F. Challenges of Preparing Secondary STEM Pre-Service Teachers in Computational Thinking. In *Proceedings of the 2022 Physics Education Research Conference, Grand Rapids, MI, USA, 13–14 July 2022*; pp. 395–400. [\[CrossRef\]](#)
34. Killen, H.; Coenraad, M.; Byrne, V.; Cabrera, L.; Mills, K.; Ketelhut, D.J.; Plane, J.D. Teacher Education to Integrate Computational Thinking into Elementary Science: A Design-Based Research Study. *ACM Trans. Comput. Educ.* **2023**, *23*, 1–36. [\[CrossRef\]](#)
35. Bandura, A. Self-Efficacy. In *Encyclopedia of Human Behavior*; Ramachandran, V.S., Ed.; Academic Press: New York, NY, USA, 1994; Volume 4, pp. 71–81.
36. Holzberger, D.; Philipp, A.; Kunter, M. How Teachers' Self-Efficacy Is Related to Instructional Quality: A Longitudinal Analysis. *J. Educ. Psychol.* **2013**, *105*, 774–786. [\[CrossRef\]](#)
37. Zee, M.; Koomen, H.M.Y. Teacher Self-Efficacy and Its Effects on Classroom Processes, Student Academic Adjustment, and Teacher Well-Being: A Synthesis of 40 Years of Research. *Rev. Educ. Res.* **2016**, *86*, 981–1015. [\[CrossRef\]](#)
38. Vongkulluksn, V.W.; Xie, K.; Hawk, N.A. Investing Time in Technology: Teachers' Value Beliefs and Time Cost Profiles for Classroom Technology Integration. *Teach. Coll. Rec.* **2020**, *122*, 1–38. [\[CrossRef\]](#)

39. Cheng, S.-L.; Lu, L.; Xie, K.; Vongkulluksn, V.W. Understanding Teacher Technology Integration from Expectancy-Value Perspectives. *Teach. Teach. Educ.* **2020**, *91*, 103062. [\[CrossRef\]](#)
40. Bas, G.; Senturk, C. Teachers' Educational Beliefs and Curriculum Orientations: A Relational Research. *Teach. Curric.* **2019**, *19*, 45–53. [\[CrossRef\]](#)
41. Cao, J.; Zhang, W. Investigating the Impact of Value Congruence on Work Engagement in EFL Teachers: The Role of Teacher Enthusiasm. *Front. Psychol.* **2023**, *14*, 1264126. [\[CrossRef\]](#) [\[PubMed\]](#)
42. OECD. *Pedagogical Knowledge and the Changing Nature of the Teaching Profession*; Guerriero, S., Ed.; Educational Research and Innovation; OECD: Paris, France, 2017; ISBN 978-92-64-27068-8.
43. Shahzad, K.; Naureen, S. Impact of Teacher Self-Efficacy on Secondary School Students' Academic Achievement. *J. Educ. Educ. Dev.* **2017**, *4*, 48. [\[CrossRef\]](#)
44. Tran, V.D. Impact of Teachers' Self-Efficacy on Their Work Engagement: A Case of Vietnam. *J. Educ. E-Learn. Res.* **2023**, *10*, 768–777. [\[CrossRef\]](#)
45. Zawacki-Richter, O.; Marín, V.I.; Bond, M.; Gouverneur, F. Systematic Review of Research on Artificial Intelligence Applications in Higher Education—Where Are the Educators? *Int. J. Educ. Technol. High. Educ.* **2019**, *16*, 39. [\[CrossRef\]](#)
46. Penney, J.; Pimentel, J.F.; Steinmacher, I.; Gerosa, M.A. Anticipating User Needs: Insights from Design Fiction on Conversational Agents for Computational Thinking. In *Proceedings of the Chatbot Research and Design*; Springer Nature Switzerland: Cham, Switzerland, 2024; pp. 204–219.
47. Qiao, Y.; Shihab, M.I.H.; Hundhausen, C. A Systematic Literature Review of the Use of GenAI Assistants for Code Comprehension: Implications for Computing Education Research and Practice. *ACM Trans. Comput. Educ.* **2026**, *26*, 1–33. [\[CrossRef\]](#)
48. Creely, E. Generative AI to Foster Computational Thinking in Initial Teacher Education: A Thematic Literature Review and Model. *Behav. Sci.* **2026**, *16*, 575. [\[CrossRef\]](#) [\[PubMed\]](#)
49. Hsu, H.-P. From Programming to Prompting: Developing Computational Thinking through Large Language Model-Based Generative Artificial Intelligence. *TechTrends* **2025**, *69*, 485–506. [\[CrossRef\]](#)
50. Cordero, J.; Torres-Zambrano, J.; Cordero-Castillo, A. Integration of Generative Artificial Intelligence in Higher Education: Best Practices. *Educ. Sci.* **2024**, *15*, 32. [\[CrossRef\]](#)
51. Knight, S.; Dickson-Deane, C.; Heggart, K.; Kitto, K.; Kozanoğlu, D.Ç.; Maher, D.; Narayan, B.; Zarrabi, F. Generative AI in the Australian Education System: An Open Data Set of Stakeholder Recommendations and Emerging Analysis from a Public Inquiry. *Australas. J. Educ. Technol.* **2023**, *39*, 101–124. [\[CrossRef\]](#)
52. Mishra, P.; Warr, M.; Islam, R. TPACK in the Age of ChatGPT and Generative AI. *J. Digit. Learn. Teach. Educ.* **2023**, *39*, 235–251. [\[CrossRef\]](#)
53. Perera, P.; Lankathilake, M. Preparing to Revolutionize Education with the Multi-Model GenAI Tool Google Gemini? A Journey towards Effective Policy Making. *J. Adv. Educ. Philos.* **2023**, *7*, 246–253. [\[CrossRef\]](#)
54. Lee, S.J.; Kwon, K. A Systematic Review of AI Education in K-12 Classrooms from 2018 to 2023: Topics, Strategies, and Learning Outcomes. *Comput. Educ. Artif. Intell.* **2024**, *6*, 100211. [\[CrossRef\]](#)
55. Alfares, N. Is Synchronous Online Learning More Beneficial than Asynchronous Online Learning in a Saudi EFL Setting: Teachers' Perspectives. *Front. Educ.* **2024**, *9*, 1454892. [\[CrossRef\]](#)
56. Eldridge, S.M.; Lancaster, G.A.; Campbell, M.J.; Thabane, L.; Hopewell, S.; Coleman, C.L.; Bond, C.M. Defining Feasibility and Pilot Studies in Preparation for Randomised Controlled Trials: Development of a Conceptual Framework. *PLoS ONE* **2016**, *11*, e0150205. [\[CrossRef\]](#) [\[PubMed\]](#)
57. Rich, P.J.; Larsen, R.A.; Mason, S.L. Measuring Teacher Beliefs about Coding and Computational Thinking. *J. Res. Technol. Educ.* **2021**, *53*, 296–316. [\[CrossRef\]](#)
58. Santiana, S.; Silvani, D.; Ruslan, R. Optimizing LMS Canvas for Interactive Online Learning Perceived by the Students. *J. Engl. Educ. Teach.* **2021**, *5*, 529–543. [\[CrossRef\]](#)
59. Fabriz, S.; Mendzheritskaya, J.; Stehle, S. Impact of Synchronous and Asynchronous Settings of Online Teaching and Learning in Higher Education on Students' Learning Experience during COVID-19. *Front. Psychol.* **2021**, *12*, 733554. [\[CrossRef\]](#) [\[PubMed\]](#)
60. Hung, C.-T.; Wu, S.-E.; Chen, Y.-H.; Soong, C.-Y.; Chiang, C.; Wang, W. The Evaluation of Synchronous and Asynchronous Online Learning: Student Experience, Learning Outcomes, and Cognitive Load. *BMC Med. Educ.* **2024**, *24*, 326. [\[CrossRef\]](#) [\[PubMed\]](#)
61. Kastner-Hauler, O.; Tengler, K.; Sabitzer, B.; Lavicza, Z. Combined Effects of Block-Based Programming and Physical Computing on Primary Students' Computational Thinking Skills. *Front. Psychol.* **2022**, *13*, 875382. [\[CrossRef\]](#) [\[PubMed\]](#)

62. Qian, Y. Prompt Engineering in Education: A Systematic Review of Approaches and Educational Applications. *J. Educ. Comput. Res.* **2025**, *63*, 1782–1818. [[CrossRef](#)]
63. Al Nabhani, F.; Hamzah, M.B.; Abuhassna, H. The Role of Artificial Intelligence in Personalizing Educational Content: Enhancing the Learning Experience and Developing the Teacher's Role in an Integrated Educational Environment. *Contemp. Educ. Technol.* **2025**, *17*, ep573. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.