information

MDPI

Article

A Novel Approach to Boosting Programming Self-Efficacy: Issue-Based Teaching for Non-CS Undergraduates in Interdisciplinary Education

Chih-Yi Tseng [1,*], Tsang-Hsiang Cheng [2] and Chih-Hung Chang [3]

[1] College of Humanities and Social Sciences, Feng Chia University, Taichung 40724, Taiwan
[2] Department of Business Administration, Southern Taiwan University of Science and Technology, Tainan 71005, Taiwan; cts@stust.edu.tw
[3] Department of Applied Mathematics, National University of Kaohsiung, Kaohsiung 81148, Taiwan; chchang@nuk.edu.tw
* Correspondence: chihytseng@o365.fcu.edu.tw

Abstract: This study examines the impact of issue-based teaching (IBT) on programming self-efficacy among non-Computer Science students. Grounded in social cognitive theory, the research investigates how IBT influences learning satisfaction and project success compared to traditional metrics. This study employed a mixed-methods approach, combining the quantitative analysis of student performance and self-efficacy measures with qualitative feedback from learning portfolios and project reports. The findings indicate that programming self-efficacy is a stronger predictor of learning satisfaction and project success than traditional performance metrics like grades. For novice programmers, IBT effectively enhances self-efficacy, positively influencing goal identification and performance. This cascade effect highlights the importance of fostering self-efficacy in programming education for non-technical students. Qualitative analysis reveals that IBT contributes to students' sense of achievement, motivation, and learning satisfaction, encouraging them to view programming as a practical problem-solving tool. This study concludes that IBT offers an effective approach to enhancing interdisciplinary and STEAM education, recommending that educators focus on building self-efficacy through issue-based, learner-centered approaches.

Keywords: interdisciplinary education; issue-based teaching; programming; self-efficacy

Citation: Tseng, C.-Y.; Cheng, T.-H.; Chang, C.-H. A Novel Approach to Boosting Programming Self-Efficacy: Issue-Based Teaching for Non-CS Undergraduates in Interdisciplinary Education. *Information* **2024**, *15*, 820. https://doi.org/10.3390/info15120820

Academic Editors: Petros Lameras, Sylvester Arnab and Panagiotis Petridis

Received: 18 November 2024
Revised: 12 December 2024
Accepted: 17 December 2024
Published: 20 December 2024

1. Introduction

The rapid advancement of Artificial Intelligence (AI) and the increasing demand for interdisciplinary competencies have significantly transformed the landscape of modern education and professional development. This paradigm shift necessitates a reevaluation of traditional educational approaches, particularly in the realm of programming education for non-Computer Science (non-CS) students. As AI technologies become integral to application software design and development, there is a pressing need for educational approaches that transcend traditional single-discipline training. Recognizing programming as a fundamental skill for navigating the digital economy, many countries have begun integrating programming education into interdisciplinary curricula.

In Taiwan, however, programming courses are predominantly confined to Computer Science (CS) departments, often neglecting the needs of non-CS students who may struggle to see the relevance of programming to their fields. This gap highlights the urgent need for innovative teaching methods that engage non-CS students and integrate programming with diverse subjects to enhance interdisciplinary thinking. This study addresses this critical issue by exploring whether issue-based teaching (IBT), an instructional approach that engages students by connecting learning to real-world issues relevant to their experiences and interests, can effectively enhance programming self-efficacy among non-CS students.

Programming self-efficacy, defined as an individual's belief in their ability to successfully perform programming tasks and solve coding problems, is a crucial factor in this context, given its significant impact on learning outcomes, particularly for non-CS students who may lack confidence in their programming abilities. This study specifically focuses on programming self-efficacy as a crucial factor, given its significant impact on learning outcomes, particularly for non-CS students who may lack confidence in their programming abilities. By examining how IBT influences programming self-efficacy, goal identification, and learning satisfaction among non-CS students, this research aims to provide valuable insights for interdisciplinary programming education and offer a foundation for developing innovative teaching methodologies.

Given the challenges and opportunities presented by the current landscape of programming education for non-CS students, this study aims to address several critical questions that emerge from the aforementioned context.

1. To what extent does issue-based teaching (IBT) influence the programming self-efficacy of non-CS students?
2. To what extent does IBT enhance the connection between programming skills and real-world applications for non-CS students?
3. What are the effects of IBT on these variables, programming self-efficacy, goal alignment, and learning satisfaction within a non-CS context?

This study aims to contribute to the field of programming education in several significant ways. Firstly, it seeks to expand our understanding of how IBT impacts programming self-efficacy among non-CS students, addressing a critical gap in the literature. Secondly, it endeavors to provide empirical evidence on the effectiveness of IBT in enhancing the connection between programming skills and real-world applications for students from diverse academic backgrounds. Lastly, this research aims to elucidate the interrelationships between programming self-efficacy, goal alignment, and learning satisfaction within the context of IBT, potentially offering valuable insights for curriculum design and pedagogical approaches in interdisciplinary programming education.

2. Literature Review

2.1. Issue-Based Teaching

Issue-based teaching (IBT) is an instructional approach that engages students by connecting learning to real-world issues relevant to their experiences and interests. For instance, in a Python programming course designed for non-CS students, the instructor introduced a "COVID-19 case analysis" topic. By connecting programming with a relevant current issue, this approach effectively engaged students. This method encourages critical thinking, collaboration, and active participation, making learning more meaningful and applicable. Hahn [1] originally described IBT as utilizing themes that resonate with learners to ignite interest and facilitate discussions that link these themes to specific subject knowledge, fostering reflection and action.

Recent studies have highlighted the effectiveness of IBT in promoting interdisciplinary learning and enhancing student engagement. For instance, Corlu et al. [2] noted that integrating IBT within STEM education not only addresses complex problems but also cultivates creativity and innovative problem-solving skills across disciplines. Bybee [3] emphasized the importance of selecting topics that align with societal needs, as demonstrated in Finland, where schools have replaced traditional subject-based teaching with IBT, effectively integrating various subjects such as economics, history, and literature.

Furthermore, the recent literature suggests that IBT can significantly improve students' motivation and self-efficacy. According to Tsai et al. [4], IBT encourages students to take ownership of their learning by allowing them to choose issues they are passionate about, thereby enhancing their engagement and commitment to the learning process. This approach is particularly beneficial in programming education, where students often struggle to see the relevance of coding skills in their fields.

In the context of programming education, combining IBT with coding instruction can foster computational thinking and problem-solving abilities. As noted by Kafai and Burke [5], engaging students in meaningful projects related to real-world issues helps demystify programming concepts and enhances their confidence in applying these skills. This study posits that IBT not only improves programming self-efficacy among non-CS students but also promotes critical skills necessary for success in an increasingly digital world.

In summary, for interdisciplinary students with non-information technology backgrounds, IBT can achieve innovation in programming education in the following three aspects:

1. Dynamic flexibility to enhance learning motivation: IBT offers dynamic topic flexibility, enabling the connection of programming issues with students' experiences and interests, thereby stimulating learning motivation.
2. Integration with real-world social issues and applications: Programming topics can be integrated with diverse real-world problems, fostering interdisciplinary thinking. This is particularly crucial in the ongoing development of AI applications across multiple domains.
3. Enhancement of active learning: Under IBT, educators can encourage students to take control of their active learning process by choosing topics they are passionate about; for instance, granting students the autonomy to select project directions enhances engagement and commitment.

2.2. Computer Programming Self-Efficacy

Computer programming self-efficacy refers to an individual's belief in their ability to successfully perform programming tasks and solve coding problems. Grounded in Bandura's [6] Social Cognitive Theory, self-efficacy is a crucial determinant of motivation and performance across various domains, including education and computer skills training [7]. Bandura posited that self-efficacy influences not only the choice of activities but also the effort and persistence individuals exhibit when faced with challenges.

Recent studies have expanded on this foundational concept, illustrating its significance in programming education. For example, Tsai et al. [4] found that students with higher programming self-efficacy are more likely to engage in complex programming tasks and demonstrate improved performance outcomes. This aligns with the findings by Li and Chen [8], who reported that self-efficacy positively correlates with goal alignment and learning satisfaction among programming students.

Askar and Davenport [9] conducted a seminal study examining factors influencing Java programming self-efficacy among engineering students. Their research revealed that prior programming experience significantly predicted self-efficacy, aligning with Bandura's [6] social cognitive theory. Interestingly, gender differences had minimal impact on programming self-efficacy. For novice programmers, self-efficacy was particularly sensitive to achievement during the initial learning stages, emphasizing the importance of early success in programming courses. These findings provide valuable insights into programming self-efficacy development, offering a foundation for understanding how factors such as prior experience, gender, and early achievements influence students' beliefs in their programming abilities, with important implications for designing effective programming instruction, particularly for non-CS majors and novice programmers.

Recent studies [10,11] have highlighted the unique challenges faced by non-CS students when learning programming, which is particularly relevant to our research on issue-based teaching (IBT) for non-CS majors. These students often encounter significant barriers due to their lack of relevant background knowledge, resulting in a steeper learning curve compared to their CS counterparts. Non-CS students frequently struggle to connect programming concepts to their primary field of study, leading to a perceived lack of relevance and diminished motivation. Furthermore, the lower intrinsic motivation among non-CS students to learn programming, coupled with the technical nature of programming courses, can lead to frustration and disengagement [12,13]. These challenges underscore the importance of tailored approaches, such as the issue-based teaching method proposed in

our study, to create more effective and engaging learning experiences for non-CS students in programming courses.

The research conducted by Gao et al. [13] further emphasizes the significance of programming self-efficacy, particularly for non-CS students. Their study demonstrates that programming self-efficacy serves as a more crucial predictor of students' programming learning performance, surpassing even prior learning experiences. This finding underscores the necessity of cultivating self-efficacy in programming education, especially for students from non-CS backgrounds.

However, the current literature on programming self-efficacy and IBT reveals significant research gaps. Firstly, studies have predominantly focused on Computer Science majors, neglecting the unique challenges faced by non-CS students in programming education. Secondly, while IBT has been explored in various educational contexts, its impact on programming self-efficacy, particularly among non-CS students, remains largely unexamined. Mudambi and Zhao [14] highlight that despite IBT's potential benefits in enhancing student engagement, its application and effects in programming education are understudied.

These research gaps underscore the need for investigating the effectiveness of IBT in enhancing programming self-efficacy among non-CS students, especially given the increasing importance of programming skills across diverse disciplines. This study aims to address these gaps by exploring how IBT can effectively improve programming self-efficacy among non-CS students and provide valuable insights for interdisciplinary programming education.

2.3. Goal Alignment and Learning Satisfaction

Goal alignment refers to the process by which students set, pursue, and achieve educational objectives that resonate with their personal interests and academic aspirations. According to Locke and Latham [15], goal alignment is crucial for enhancing motivation and improving performance outcomes. Bandura [16] further asserts that self-efficacy plays a significant role in this alignment, as students who believe in their capabilities are more likely to set challenging goals and persist in achieving them.

Recent research has reinforced the connection between goal alignment, self-efficacy, and learning satisfaction. Lim and Kim [17] conducted a study involving 200 Computer Science students that demonstrated a positive relationship between self-efficacy and goal alignment in programming tasks. Their findings indicated that students with higher self-efficacy not only set more ambitious goals but also reported greater satisfaction with their learning experiences. This aligns with the work of Li and Chen [8], who found that clear goal-setting enhances learning satisfaction, leading to improved academic performance.

Learning satisfaction is defined as the degree to which students feel fulfilled and content with their educational experiences. It encompasses emotional responses to the learning process, including feelings of achievement, engagement, and relevance [18]. Research by Uçar and Sungur [19] highlights that students who experience higher levels of satisfaction are more likely to engage deeply with the material, leading to better retention of knowledge and skills.

In programming education, fostering goal alignment through effective teaching strategies can significantly impact students' learning satisfaction. For instance, when educators implement IBT, they encourage students to connect programming tasks with real-world issues that matter to them. This relevance not only motivates students but also helps them set meaningful goals aligned with their interests. As noted by Mudambi and Zhao [14], such alignment can alleviate anxiety associated with programming, enhancing overall satisfaction with the learning experience.

Moreover, research indicates that when students perceive a strong connection between their goals and the curriculum, they are more likely to invest effort into their studies. This dynamic is particularly relevant for non-CS students who may initially struggle to see the value of programming in their fields. By integrating IBT into programming instruction, educators can create an environment where students feel empowered to set personal goals that align with both their academic pursuits and real-world applications.

This study aims to elucidate the interrelationships among IBT, programming self-efficacy, goal alignment, and learning satisfaction within the context of non-CS students. Specifically, we hypothesize that IBT enhances programming self-efficacy, which in turn positively influences goal alignment and learning satisfaction. By examining these relationships, we address our research questions on how IBT impacts programming self-efficacy and its subsequent effects on learning outcomes. Understanding these interactions is crucial for developing effective pedagogical strategies in interdisciplinary programming education.

3. Research Methods

3.1. Instructional Design

This study employs an issue-based instructional design framework aimed at enhancing programming self-efficacy among non-CS students. The instructional design is structured to facilitate active learning through real-world issues that resonate with students' experiences and interests. As illustrated in Table 1, the teaching process begins with discussions centered on relevant issues, guiding students to set specific programming goals before introducing Python coding.

Table 1. Examples of frameworks for issue-based programming instruction.

Topic—Healthy Diet	Topic—Oppa Face
- Discussion: discuss the importance of healthy dining options on campus. - Goals: determine how to use appropriate data types to store calorie information; design a structure to store basic vendor information, including vendor distance and product purchase calculations. - Coding Skills: understand and use Python data types (integers, strings, Booleans); lists, dictionaries, tuples, and sets; proficiency in using variables and various operations.	- Discussion: explore face-swapping apps that can predict future appearances or recreate past looks. - Goals: develop a feature to calculate facial similarity; build an app for recognizing emotions (joy, anger, sorrow, happiness); design an app for whitening effects and selfie retouching. - Coding Skills: master the skill of installing external packages, such as the image library Pillow.
Topic—COVID-19 Analysis	**Topic—Chatbot AI Support**
- Discussion: discuss the global spread of COVID-19 and its impact on daily life. - Goals: develop a mask map app to display nearby locations where masks are available; analyze COVID-19 case numbers. - Coding Skills: use Python to handle Web APIs, web data, and open data; learn to use the "requests" library to make HTTP requests and access Web APIs.	- Discussion: create an emotional support AI app that gauges moods and tells jokes. - Goals: develop a chatbot for emotional support; design an interactive storytelling AI. - Coding Skills: use Python for semantic applications and chatbot development; design a voice memo program and a memo-taking chatbot.

Given that our target audience consists of Python beginners from various departments with non-CS backgrounds, in topic formulation for IBT, a systematic approach was employed to ensure relevance and educational value. The systematic criteria used included:

1. Relevance to current events or student experiences: topics such as COVID-19 analysis and healthy diet enhance student engagement.
2. Potential for real-world application: examples include AI chatbots and face-swapping apps, demonstrating practical utility.
3. Interdisciplinary connections: students from different fields can link programming to their studies; for instance, Western Languages students might focus on "English learning", while Environmental Science students could explore "carbon emission calculations".

This approach aims to ensure that the programming tasks are accessible, engaging, and relevant to the students' diverse academic backgrounds and interests. This aligns with recent findings by Tsai et al. [4], which suggest that contextualized learning significantly enhances student motivation and the retention of programming concepts.

The instructional activities follow a systematic progression, as outlined in Table 2, which includes the following six steps. This instructional design not only enhances pro-

gramming skills but also promotes logical thinking, problem-solving abilities, and interdisciplinary integration [14]. By employing issue-based teaching methods, this study aims to create a learner-centered environment that alleviates anxiety associated with programming tasks while fostering deeper engagement with the subject matter.

Table 2. Steps of issue-based programming instruction.

	Teaching Steps	Explanation
1.	Issues and Problem Discussion	Select topics that evoke shared experiences, such as current events (e.g., COVID-19), air quality, or popular applications (e.g., facial recognition, AI chatbots). Engage students in discussions and problem formulation to boost motivation and interest in learning.
2.	Introduction to Programming Goals, Process, and Demonstration of Examples	After defining the problem through discussion, outline the programming objectives and processes. Provide demonstrations of relevant Python code examples to illustrate key concepts.
3.	Comprehensive Application and Design Thinking	Gradually deepen and expand the topics and programming objectives each week. Encourage students to integrate their knowledge creatively to solve problems, regularly prompting them with questions to inspire innovative topic designs.
4.	Python Teaching and Coding Exercises	Sequentially teach Python fundamentals, including data types (integers, strings, Booleans), lists, dictionaries, tuples, and sets, as well as Python libraries, Web APIs, web data handling, open data access, and web scraping techniques.
5.	Topic Assignments and Project Implementation	Apply weekly Python lessons to new assignments that encourage students to extrapolate and modify their work. Students should compile completed projects into personal portfolios for peer review, integrating feedback from these reviews into their final grade assessment.
6.	Optimization Suggestions and Ratings	Provide constructive feedback during project presentations. Use the revised versions of projects as the basis for evaluation to ensure continuous improvement.

3.2. Research Model

Based on the literature review, a comprehensive research model is proposed, as illustrated in Figure 1. This model encompasses several key constructs: programming self-efficacy (PSE), programming goal identification (PGI), and self-satisfaction with learning outcomes (SLO). Additionally, previous programming performance (PPP) is evaluated through assignments and mid-term project grades, while current programming performance (CPP) is assessed via final project grades. The proposed hypotheses are as follows:

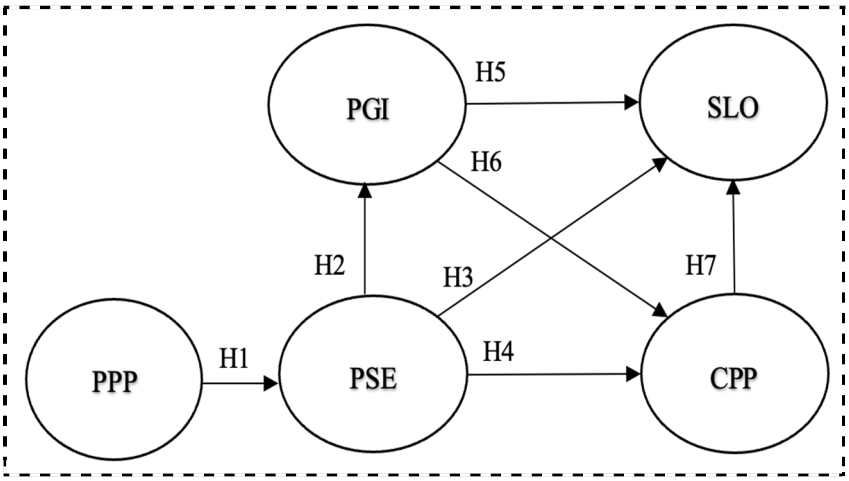

Figure 1. The research model.

H1: *Higher previous programming performance (PPP) (e.g., assignment scores, mid-term project grades) is positively associated with higher programming self-efficacy (PSE). Bandura [6] posited that successful past performance enhances self-efficacy beliefs, suggesting that students who perform well in earlier tasks will feel more confident in their programming abilities.*

H2: *Higher programming self-efficacy (PSE) is positively correlated with higher programming goal identification (PGI). Usher and Pajares [20] found that students with greater self-efficacy in science learning demonstrate stronger alignment with their learning goals, indicating that a similar relationship may exist in programming contexts.*

H3: *Higher programming self-efficacy (PSE) leads to greater self-satisfaction with learning outcomes (SLO). Doménech-Betoret et al. [21] identified self-efficacy as a critical factor influencing satisfaction, suggesting that students who believe in their capabilities are more likely to feel satisfied with their learning experiences.*

H4: *Higher programming self-efficacy (PSE) results in improved current programming performance (CPP). This aligns with findings from Compeau and Higgins [7] and Webster and Martocchio [22], who consistently demonstrated a positive relationship between self-efficacy and performance in computer skills training.*

H5: *Higher programming goal identification (PGI) is associated with increased self-satisfaction with learning outcomes (SLO). Ames [23] found a significant positive correlation between goal identification and satisfaction, suggesting that students who set clear goals are more likely to feel fulfilled by their learning process.*

H6: *Higher programming goal identification (PGI) leads to better current programming performance (CPP). Research by Kim and Kwon [24] indicates that individuals with well-defined goals achieve superior performance outcomes, supporting this hypothesis.*

H7: *Higher current programming performance (CPP) positively influences self-satisfaction with learning outcomes (SLO). Recent findings by Li and Chen [8] suggest that students who perform well in programming tasks tend to report higher levels of satisfaction regarding their educational experiences.*

This research model underscores the interconnectedness of self-efficacy, goal alignment, and performance outcomes within the context of issue-based teaching for non-CS students. By examining these relationships, this study aims to provide insights into how enhancing programming self-efficacy can lead to improved academic performance and satisfaction among learners.

3.3. Research Subjects

This study was conducted in a course titled Python Basis Programming, offered at a university in Taiwan. The course was open exclusively to undergraduate students from non-CS majors. A total of 120 students enrolled in the course (two classes). This targeted sample is essential for examining how issue-based teaching (IBT) influences programming self-efficacy, specifically within a single institution. Prior to this study, these students had limited exposure to programming concepts, making them ideal candidates for assessing the impact of IBT on their self-efficacy and overall learning experience.

Among them, 117 students completed the study questionnaire. After removing missing data and outliers using the z-score method, 98 valid samples were obtained, with approximately 40% male and 60% female. Our sample represents a substantial portion (81.67%) of the 120 students initially enrolled. The majority of students were from the College of Humanities and Social Sciences (No. 46, 47%), followed by the College of Management (No. 29, 30%), collectively accounting for 77% of the sample. Other participants were from the College of Science (No. 13, 13%), the College of Engineering

(No. 9, 9%), and the College of Law (No. 1, 1%). The subjects represented a broader non-CS student population.

3.4. Construct Measurement

To effectively evaluate the constructs within this study, a well-defined measurement framework was developed as a questionnaire in Appendix A. As seen in Table 3, each construct is measured using validated instruments, ensuring reliability and validity in the data collection process. The measurement instruments were pre-tested with a small group of students to ensure clarity and appropriateness before full implementation. Data collected through these instruments were analyzed using statistical methods to explore the relationships among the constructs, thereby providing insights into how IBT influences programming self-efficacy and related outcomes.

Table 3. Construct measurement and definition.

Construct	Measurement Definition	Measurement Instrument
Programming Self-Efficacy (PSE)	Assesses students' confidence in performing specific programming tasks using a 7-point Likert scale (1 = not confident at all, 7 = very confident).	Computer Self-Efficacy Scale (modified from Compeau and Higgins [7])
Programming Goal Identification (PGI)	Evaluates students' ability to identify and articulate their programming goals in relation to course objectives, rated on a 7-point Likert scale.	Goal-setting Questionnaire (modified from Locke and Latham [15])
Self-Satisfaction with Learning Outcomes (SLO)	Measures students' perceived satisfaction with their learning experiences in the course using a 7-point Likert scale.	Satisfaction Scale (modified from Uçar and Sungur [19])
Previous Programming Performance (PPP)	Quantified through students' grades prior to the study, serving as an objective measure of past performance.	Grades from assignments and mid-term projects
Current Programming Performance (CPP)	Assessed based on the final project grades received by participants at the conclusion of the course, reflecting overall performance after instruction.	Final project grades

4. Analysis

4.1. Quantitative Analysis

Table 4 presents the descriptive statistics and reliability test results for each construct. The internal consistency (Cronbach's α) values for all constructs exceed the threshold level of 0.7 [25], indicating a good level of reliability for the scales used in this study. Additionally, Table 4 preliminarily demonstrates the benefits of IBT methods. PSE scored the highest at 6.52, followed by SLO at 5.40 and PGI at 4.87, all of which are above moderate levels (7-point scale). The average score for regular assignments and mid-term project grades (previous programming performance) was 88.43, while the final project grade (programming performance) improved to 91.13. This indicates that issue-based teaching methods effectively enhance non-CS students' programming self-efficacy, goal achievement, and programming performance.

Table 4. Descriptive statistics and reliability test results.

Construct Variable	Mean	Standard Deviation	Cronbach's α
PPP	88.43	8.25	#
CPP	91.13	4.48	#
PSE	6.52	1.57	0.93
SLO	5.40	0.86	0.91
PGI	4.87	1.09	0.83

 # Note: PPP and CPP are formative constructs and do not have Cronbach's α values.

Subsequently, exploratory factor analysis was conducted to assess the reliability and validity of the construct items, as shown in Table 5. Using principal component analysis

and varimax rotation, all item loadings exceeded 0.5 and were lower on other constructs, indicating adequate convergent validity among the three constructs. Table 6 presents the results for the correlation coefficients, discriminant validity, and composite reliability among the three constructs. The square root of the average variance extracted (AVE) was above 0.7, surpassing the 0.5 threshold [26]; additionally, the composite reliability Cronbach's α values for all constructs were above 0.80, meeting the threshold of 0.7 [26], indicating that the assessment questionnaire possesses sufficient convergent validity and discriminant validity.

Table 5. Exploratory factor analysis results.

	PSE	SLO	PGI
PSE5	0.843	−0.073	0.173
PSE4	0.832	0.029	0.100
PSE6	0.820	0.148	0.255
PSE7	0.809	0.133	0.277
PSE9	0.788	0.037	−0.093
PSE3	0.748	0.251	0.003
PSE8	0.734	0.193	0.173
PSE10	0.718	0.080	0.074
PSE1	0.716	0.281	−0.041
PSE2	0.679	0.299	−0.018
SLO3	0.144	0.918	0.087
SLO1	0.119	0.909	0.046
SLO2	0.250	0.857	0.030
SLO4	0.138	0.750	0.276
PGI1	0.135	0.094	0.889
PGI2	0.131	0.136	0.877
PGI4	0.120	−0.004	0.736
PGI3	−0.007	0.113	0.653

Table 6. Correlation coefficients, discriminant validity, and composite reliability test results.

CONSTRUCT VARIABLE	COMPOSITE RELIABILITY	PPP	CPP	PSE	SLO	PGI
PPP	#	**#**				
CPP	#	−0.04	**#**			
PSE	0.94	−0.03	0.17	**0.77**		
SLO	0.92	−0.12	0.06	0.53 **	**0.86**	
PGI	0.87	0.17	0.18	0.22 *	0.21	**0.80**

*# Formative construct; ** $p < 0.01$, * $p < 0.05$ (two-tailed). Diagonal bold values represent the square root of the average variance extracted (AVE).*

Two important findings emerge from Table 6. First, when IBT methods are introduced to non-CS learners, there is a significant positive correlation between their programming self-efficacy and self-satisfaction with learning outcomes ($\beta = 0.53$ **, $p < 0.01$). Second, a significant positive correlation is also observed between their programming self-efficacy and programming goal identification ($\beta = 0.22$ *, $p < 0.05$).

Given the instructional nature of this study's IBT experiment within the context of a specific course, the sample size ($n = 98$) was relatively small, yet approaching the recommended minimum threshold [27]. Consequently, partial least squares structural equation modeling (PLS-SEM) was deemed appropriate for model evaluation and hypothesis testing [28–30]. The assessment of the local model fit was conducted through examination of R^2 values, path coefficients, and their respective significance levels utilizing bootstrapping procedures with 5000 resamples. Notably, the model accounted for 21.1% of the variance in self-satisfaction with learning outcomes (SLO) (Figure 2), indicating moderate explanatory power.

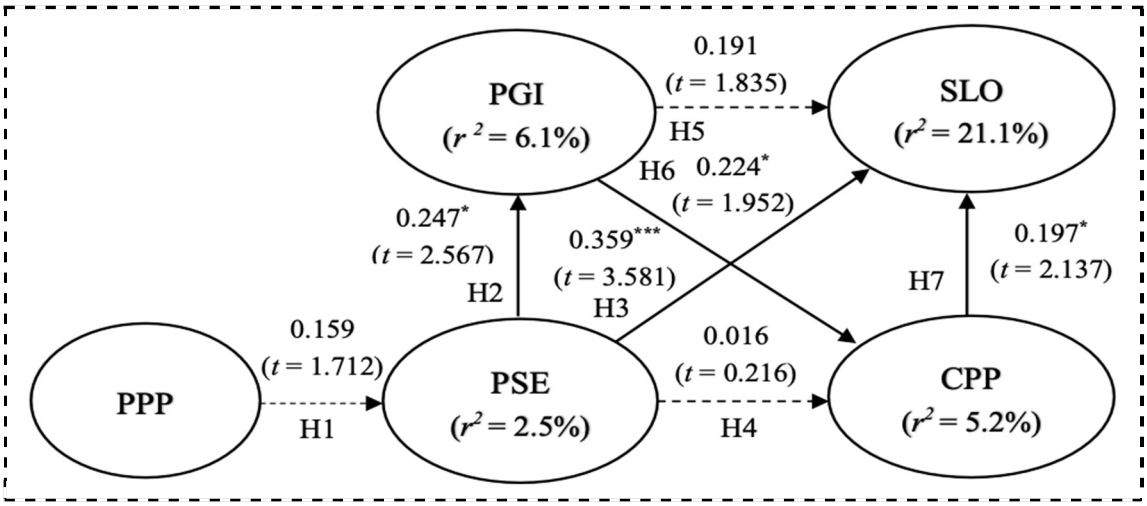

Figure 2. Model testing results. **: t-value > 1.96, p < 0.05 (two-tailed); ***: t-value > 3.291, p < 0.001 (two-tailed).*

The results of the hypothesis testing, illustrated in Figure 2 and detailed in Table 7, reveal three key findings. First, the model accounts for 21.1% of the variance in self-satisfaction with learning outcomes (SLO) among non-CS learners, demonstrating strong predictive power. Second, programming self-efficacy (PSE) had the strongest impact on self-satisfaction with learning outcomes (H3, $\beta = 0.359$ ***, $t = 3.581$, $p < 0.001$), even surpassing grades in current programming performance (CPP) on SLO (H7, $\beta = 0.197$ *, $t = 2.137$, $p < 0.05$). Third, although PSE did not directly influence CPP (H4, $\beta = 0.016$, $t = 0.216$, $p > 0.05$), it significantly impacted programming goal identification (PGI) (H2, $\beta = 0.247$ *, $t = 2.567$, $p < 0.05$), which in turn positively affected CPP (H6, $\beta = 0.224$ *, $t = 1.952$, $p < 0.05$). This indicates that for non-CS students new to Python programming, IBT effectively boosts programming self-efficacy and confidence in achieving programming goals (H2), leading to improved current programming performance (H6) and greater satisfaction with learning outcomes (H7).

Table 7. Hypothesis testing results.

	Hypothesis	Result
H1	Higher previous programming performance (PPP) is positively associated with higher programming self-efficacy (PSE).	NS.
H2	Higher programming self-efficacy (PSE) is positively correlated with higher programming goal identification (PGI).	S.
H3	Higher programming self-efficacy (PSE) leads to greater self-satisfaction with learning outcomes (SLO).	S.
H4	Higher programming self-efficacy (PSE) results in improved current programming performance (CPP).	NS.
H5	Higher programming goal identification (PGI) is associated with increased self-satisfaction with learning outcomes (SLO).	NS.
H6	Higher programming goal identification (PGI) leads to better current programming performance (CPP).	S.
H7	Higher current programming performance (CPP) positively influences self-satisfaction with learning outcomes (SLO).	S.

Note: NS.—Not supported. S.—Supported.

4.2. Students' Feedback Analysis

This study additionally collected students' feedback from their learning portfolios and project reports. The qualitative analyses were designed to explore several key areas: perceptions of issue-based teaching, impact on programming self-efficacy, and learning satisfaction. These themes were selected as they directly align with the study's core objectives

of examining the effects of IBT on non-CS students' programming self-efficacy and its impact on learning satisfaction. This approach allows for a comprehensive exploration of how IBT influences students' learning experiences and outcomes in programming education.

4.2.1. Perceptions of Issue-Based Teaching

> *Student 1: "At the beginning of the semester, programming seemed like an abstract concept far removed from my daily life. However, as the course progressed, I discovered its accessibility and practical applications. The instructor's issue-based teaching approach was instrumental in bridging the gap between theory and practice, enabling me to grasp programming concepts and apply them to solve real-world problems".*
>
> *Student 2: "The flexibility offered by the instructor in allowing us to select topics that resonated with our interests was a game-changer. This approach empowered me to design my own project, tailoring it to a context I found engaging. As the course unfolded, I was able to adapt the programming examples taught in class to my chosen subject, making the learning process both relevant and rewarding".*

Student feedback demonstrates that IBT plays a pivotal role in enhancing learning motivation and fostering design thinking among non-CS students. The approach offers two key benefits: First, practical application—IBT enables students to apply programming concepts to real-world problems, thereby increasing their understanding of its practical value. This approach transforms programming from an abstract concept into a tangible tool for problem-solving, making it more accessible and relevant to students' experiences. Second, personal engagement—by encouraging students to address personally interesting problems, IBT motivates them to actively explore and develop their programming skills. This self-directed approach not only deepens their understanding of programming concepts but also cultivates critical problem-solving abilities.

As a result, IBT positions students as active agents in their own learning journey. Through this process, students gradually build confidence in their programming abilities, develop a deeper appreciation for the subject, and acquire valuable skills that extend beyond the classroom. The approach effectively bridges the gap between theoretical knowledge and practical application, making programming more engaging and meaningful for non-CS students.

4.2.2. Impact on Programming Self-Efficacy

> *Student 3: "This course has not only fulfilled my long-standing desire to acquire interdisciplinary and practical skills but has also ignited a passion for embracing new challenges. It has exceeded my expectations in bridging the gap between theory and real-world application".*
>
> *Student 4: "I've noticed a significant improvement in my problem-solving approach. When encountering errors, I now calmly analyze the source, which has made me more attentive during coding. This process has gradually cultivated a more methodical and confident approach to overcoming programming challenges".*
>
> *Student 5: "The instructor's perspective resonated with me—while engineers excel at designing functions, we, as humanities professionals, bring a unique creative dimension to programming. This insight has helped me appreciate the value of my background in this technical field".*
>
> *Student 6: "My perception of programming languages has completely transformed. What once seemed intimidating now appears accessible and intriguing. This newfound confidence has sparked a desire to explore and learn additional programming languages in the future".*
>
> *Student 7: "The course has inspired me to set ambitious goals for my programming journey. I'm now planning to delve into C++ and Java, with a focus on developing personal websites and applications. This represents a significant shift in my career aspirations and skill set".*

The student responses collectively demonstrate the effectiveness of IBT in programming education for non-CS students. IBT has successfully enhanced motivation and engagement with programming, improved problem-solving skills and resilience, fostered appreciation for interdisciplinary perspectives, transformed perceptions of programming accessibility, and inspired long-term learning goals and career considerations. These qualitative outcomes align closely with the study's quantitative findings, reinforcing the positive impact of IBT on programming self-efficacy, goal identification, and satisfaction with learning outcomes among non-CS students. The approach has not only made programming more accessible and relevant but has also empowered students to see its broader applications in their respective fields, potentially influencing their future academic and career paths.

4.2.3. Learning Satisfaction

Student 8: "Achieving the desired outcome instilled in me a tremendous sense of accomplishment and ignited my passion for programming".

Student 9: "The ability to apply my newfound knowledge to produce tangible results brought an indescribable sense of achievement".

Student 10: "Connecting programming to real-life topics has been the greatest motivator in my learning journey. The joy I felt upon completing my own program and proudly demonstrating the app I built to my friends was unparalleled".

Student 11: "Acquiring a new skill outside my major field has given me an incredible sense of accomplishment, especially the feeling of creating something from scratch".

Student 12: "Developing a program on a topic I'm passionate about feels both innovative and deeply fulfilling".

Student 13: "The moment I completed the program, I was overwhelmed with excitement; the sense of achievement was immeasurable".

The IBT method has proven highly effective in fostering interest and engagement in programming among non-CS students. By enabling learners to achieve concrete goals through real-world problem-solving, IBT significantly enhances the learning experience. This approach integrates programming with everyday topics, demonstrating its real-world relevance and appeal, which in turn makes the subject more accessible and engaging for students from diverse academic backgrounds. One of the key strengths of IBT lies in its ability to cultivate a sense of personal achievement and boost motivation. Students experience a profound sense of accomplishment when completing self-directed projects, and the tangible outcomes of their work significantly enhance their learning motivation. This approach not only helps students develop technical skills but also nurtures a sense of pride and creativity, making the learning process more rewarding and meaningful.

Furthermore, IBT promotes cross-disciplinary development by helping students recognize the value of programming skills beyond their primary field of study. This approach encourages learners to appreciate the potential for interdisciplinary applications, broadening their perspectives and preparing them for the increasingly interconnected nature of modern professional environments. In conclusion, the IBT method offers a comprehensive approach to programming education that goes beyond technical skill development, fostering a more holistic and engaging learning experience for non-CS students.

5. Discussion

5.1. Issue-Based Practice in Enhancing Programming Self-Efficacy

As quantitative findings, the empirical results and student feedback demonstrate that for non-CS learners, our research model demonstrates strong explanatory power for self-satisfaction with learning outcomes, accounting for 21.1% of the variance. The primary influencing factors are programming self-efficacy (H3, β = 0.359 ***, t = 3.581) and current programming performance (H7, β = 0.197 *, t = 2.137). Notably, current

programming performance is positively impacted by programming self-efficacy, mediated through programming goal identification (H2, β = 0.247 *, t = 2.567; H6, β = 0.224 *, t = 1.952; r^2 = 5.2%). This finding suggests that engaging students in projects on topics they find personally relevant or interesting can effectively stimulate self-efficacy and lead to high levels of self-satisfaction.

As qualitative insights, analysis of student feedback indicates that IBT significantly enhanced programming self-efficacy among non-CS students. Numerous students reported that IBT enabled them to apply programming concepts to real-world problems, thereby strengthening their understanding and confidence in these skills. This pedagogical approach not only fostered students' learning motivation but also encouraged them to select project topics aligned with their personal interests. Such autonomy resulted in a more active exploration of programming concepts. Moreover, students consistently reported increased composure and organization when approaching problem-solving tasks, suggesting that IBT effectively cultivated their problem-solving abilities and self-assurance.

This teaching method allowed students to perceive programming not as an abstract concept, but as a practical tool applicable within their respective fields. Overall, IBT not only elevated students' programming self-efficacy but also promoted their interest in coding and the establishment of long-term learning objectives, demonstrating its potential in interdisciplinary education. These findings offer valuable insights for interdisciplinary programming education, suggesting that educators should prioritize cultivating students' self-efficacy through issue-oriented, learner-centered pedagogical strategies, rather than solely relying on traditional performance metrics.

5.2. Self-Efficacy's Role in Learning Satisfaction

As quantitative findings, in the context of IBT, programming self-efficacy emerges as the strongest predictor of self-satisfaction with learning outcomes (H3, β = 0.359 ***, t = 3.581), surpassing even the impact of current programming performance (final project grade) (H7, β = 0.197 *, t = 2.137). This result underscores the significant benefit of allowing students to choose and work on topics of interest, particularly for non-CS learners. It suggests that the process of personally solving self-defined topics and problems can provide a greater sense of accomplishment than traditional grade-based assessments.

Moreover, as qualitative insights, student feedback indicates that IBT is highly effective in enhancing programming self-efficacy and learning satisfaction among non-CS students. Numerous students reported that the opportunity to select and solve problems of personal interest provided them with a profound perceived achievement and fulfillment (e.g., students 8, 10, and 12). These responses clearly demonstrate that IBT, by allowing students to choose topics of interest and personally address challenges, can offer a greater sense of accomplishment than traditional grade-based assessments. This approach not only elevates students' self-efficacy but also significantly enhances their learning satisfaction, which aligns with the quantitative analysis results indicating that programming self-efficacy is the strongest predictor of learning satisfaction.

These findings align with and extend Bandura's [6] social cognitive theory, particularly in the context of programming education for non-CS students. The strong relationship between programming self-efficacy and learning satisfaction demonstrates how mastery experiences, provided through IBT, can significantly enhance students' belief in their capabilities.

5.3. Past Performance and Self-Efficacy

Interestingly, this study found that past programming performance (e.g., previous assignments and midterm project scores) does not significantly impact programming self-efficacy (H1, β = 0.159, t = 1.712). Student feedback indicates that issue-based teaching enables students to choose topics of interest and apply their knowledge in assignments, fostering a sense of ownership and relevance. Furthermore, self-efficacy in programming is dynamically shaped by multiple factors, including instructional emphasis on practical relevance [31]. Our study demonstrates that IBT, by allowing students to select personally

meaningful topics, enhances engagement and motivation to achieve self-determined programming objectives. This approach appears to be more effective in building self-efficacy than relying on past performance metrics. Moreover, our finding that programming self-efficacy is a stronger predictor of learning satisfaction than traditional performance metrics aligns with Tsai et al.'s [4] work, which found that self-efficacy significantly impacted programming performance. However, our study extends this by demonstrating the particular importance of self-efficacy for non-CS students in an IBT context.

5.4. Practical Implication for Programming Education

This study investigates the impact of IBT on programming self-efficacy among non-CS students, offering several significant contributions to the field of programming education.

Implications for Research Question 1: This study demonstrates that IBT significantly enhances programming self-efficacy among non-CS students. The empirical results indicate that IBT effectively cultivates students' confidence in their programming abilities, particularly for those from non-technical backgrounds. Our research demonstrates that IBT is highly effective in enhancing programming self-efficacy among non-CS students. This study reveals that programming self-efficacy is a stronger predictor of learning satisfaction and project success than traditional performance metrics such as grades. This finding underscores the importance of fostering self-efficacy in programming education, particularly for students from non-technical backgrounds.

Implications for Research Question 2: This study demonstrates that IBT significantly enhances the connection between programming skills and real-world applications for non-CS students. The empirical evidence suggests that IBT effectively bridges the gap between theoretical programming concepts and their practical implementation in real-world scenarios, particularly for students from non-technical backgrounds. Our research provides valuable insights into the effectiveness of IBT as a pedagogical approach. By allowing students to work on projects related to their interests and real-world issues, IBT not only enhances motivation and engagement but also helps students overcome their apprehensions about programming. This approach proves particularly beneficial for non-CS students, fostering deeper engagement with the subject matter and promoting interdisciplinary thinking. IBT is particularly effective in fostering self-efficacy among non-CS students. The issue-based approach enables these students to connect programming concepts with real-world problems they find personally relevant, thereby enhancing their confidence in their programming abilities. This method proves especially beneficial for students from non-technical backgrounds, as it provides a contextual framework that makes programming more accessible and meaningful to their diverse academic pursuits.

Implications for Research Question 3: This study demonstrates that IBT exerts beneficial effects on the interrelationships among programming self-efficacy, goal alignment, and learning satisfaction within a non-CS context. The empirical evidence suggests that IBT facilitates positive interactions between these variables, enhancing the overall learning experience for students from non-technical backgrounds. The result highlights the mediating role of programming goal identification between self-efficacy and performance. While programming self-efficacy does not directly influence current programming performance, it significantly impacts goal identification, which in turn positively affects performance. This cascade effect emphasizes the importance of helping students set clear, achievable goals in their programming journey.

5.5. Theoretical Contributions to Programming Education

This study offers several significant theoretical contributions to the field of programming education:

1. Novel theoretical framework: This research advances a novel theoretical framework for understanding how issue-based teaching (IBT) enhances programming self-efficacy among non-Computer Science students. This framework provides a foundation for

exploring the relationships between innovative pedagogical approaches and learning outcomes within interdisciplinary contexts.

2. Extension of existing theories: Our findings extend Bandura's [6] social cognitive theory by demonstrating how IBT can provide mastery experiences and vicarious learning opportunities that enhance self-efficacy in programming education. This extension is particularly relevant for non-CS students in interdisciplinary settings.
3. Integration of self-efficacy and goal alignment: This study elucidates the interplay between programming self-efficacy, goal alignment, and learning satisfaction within the context of IBT. This integrated perspective offers new insights into the mechanisms by which innovative teaching methods can impact learning outcomes.
4. Contextualization for non-CS students: While previous research has explored problem-based learning for non-CS students, our study is unique in its focus on issue-based teaching specifically for this population. Our findings suggest that IBT may be particularly effective for building self-efficacy in non-CS students by allowing them to connect programming concepts to personally relevant real-world issues.
5. Implications for future research: This theoretical framework lays the groundwork for future studies exploring the long-term impacts of IBT on students' career choices and professional development in programming and related fields. It also provides a basis for investigating how this approach can be scaled and adapted across different educational contexts and disciplines.

These theoretical contributions not only advance our understanding of programming education for non-CS students but also offer valuable insights into curriculum design and pedagogical approaches in interdisciplinary education.

6. Conclusions and Limitations

This study provides valuable insights into programming education, particularly within interdisciplinary contexts. The findings demonstrate that issue-based teaching (IBT) significantly enhances programming self-efficacy among non-Computer Science students. The results indicate that programming self-efficacy is a stronger predictor of learning satisfaction and project success than traditional performance metrics. IBT effectively connects programming skills with real-world applications by allowing students to choose topics of interest. Furthermore, programming goal identification mediates the relationship between self-efficacy and performance, highlighting the importance of helping students set clear, achievable goals. These findings suggest that educators should prioritize cultivating students' self-efficacy through issue-oriented, learner-centered approaches rather than solely relying on traditional performance metrics. However, this study has several limitations. The research was conducted at a single university, potentially limiting the generalizability of the results. Future research directions could include investigating how this approach can be scaled and adapted across different educational contexts and disciplines, exploring how IBT affects students with diverse learning styles and backgrounds, and providing a more comprehensive understanding of IBT's effectiveness in interdisciplinary programming education.

Author Contributions: Conceptualization, C.-Y.T.; methodology, C.-Y.T.; validation, C.-Y.T., T.-H.C. and C.-H.C.; formal analysis, C.-Y.T.; writing—original draft preparation, C.-Y.T.; writing—review and editing, C.-Y.T., T.-H.C. and C.-H.C. All authors have read and agreed to the published version of the manuscript.

Funding: The first author is partially supported by the National Ministry of Education, ROC (project number: PGE1122499, PGE107043).

Institutional Review Board Statement: Not applicable.

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

Data Availability Statement: Data sharing is not applicable to this article.

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

Appendix A Questionnaire

- Programming Self-Efficacy
  1. I can use Python even if no one is available to show me how.
  2. I can use Python even if I have never had experience with similar programs.
  3. I can use Python as long as I have examples to refer to.
  4. I can use Python if someone demonstrates it to me once before I try.
  5. I can use Python as long as I have someone to ask when I encounter problems.
  6. I can use Python if someone teaches me how to use it at the start.
  7. I can use Python if I have plenty of time to complete tasks with it.
  8. I can use Python as long as it has online help functions.
  9. I can use Python if someone briefly shows me how to operate it first.
  10. If I have experience with similar programs, I can use a new programming language like Python.
- Self-Satisfaction with Learning Outcomes
  1. I am satisfied with the results I have achieved in my learning so far.
  2. I am pleased with the outcomes I have achieved in my learning so far.
  3. I feel good about the results I have achieved in my learning so far.
  4. I find the results I have achieved in my learning so far to be valuable.
- Programming Goal Identification
  1. The programming goals I have set are impossible to achieve.
  2. It seems unrealistic to achieve the programming goals I have set.
  3. The programming goals I have set may need to be adjusted depending on progress.
  4. To be honest, I don't really care whether I achieve the programming goals I have set.

References

1. Hahn, C.L. Research on issues-centered social studies. In *Handbook on Teaching Social Issues (NCSS Bulletin 93)*; Evans, R.W., Saxe, D.W., Eds.; National Council for the Social Studies: Washington, DC, USA, 1996; pp. 25–41.
2. Corlu, M.S.; Capraro, R.M.; Capraro, M.M. Introducing STEM education: Implications for educating our teachers for the age of innovation. *Educ. Sci.* **2014**, *39*, 74–85.
3. Bybee, R.W. *The Case for STEM Education: Challenges and Opportunities*; National Science Teachers Association: Arlington, VA, USA, 2013.
4. Tsai, M.J.; Wang, C.Y.; Hsu, P.F. Developing the computer programming self-efficacy scale for computer literacy education. *J. Educ. Comput. Res.* **2019**, *56*, 1345–1360. [CrossRef]
5. Kafai, Y.B.; Burke, Q. The importance of design in learning. In *Design, Make, Play: Growing the Next Generation of STEM Innovators*; Honey, M., Kanter, D.E., Eds.; Routledge: New York, NY, USA, 2014; pp. 1–20.
6. Bandura, A. *Social Foundations of Thought and Action: A Social Cognitive Theory*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1986.
7. Compeau, D.R.; Higgins, C.A. Computer self-efficacy: Development of a measure and initial test. *MIS Q.* **1995**, *19*, 189–211. [CrossRef]
8. Li, Y.; Chen, X. The impact of goal alignment on learning satisfaction in higher education: A focus on programming courses. *Comput. Educ.* **2021**, *162*, 104090.
9. Askar, P.; Davenport, D. An investigation of factors related to self-efficacy for Java programming among engineering students. *Turk. Online J. Educ. Technol.* **2009**, *8*, 26–32.
10. Ismail, N.Z.; Razak, M.R. The challenges of learning programming subject in online distance learning (ODL) environment at UiTM Pahang. *GADING J. Sci. Technol.* **2021**, *4*, 27–31.
11. Singh, S. Identifying Learning Challenges faced by Novice/Beginner Computer Programming Students: An Action Research Approach. In Proceedings of the 6th Software Engineering Education Workshop (SEED 2022), CEUR Workshop Proceedings, Virtual, Japan, 6 December 2022; Volume 3330.
12. Chen, G.; Guo, W. Emotional intelligence can make a difference: The impact of principals' emotional intelligence on teaching strategy mediated by instructional leadership. *Educ. Manag. Adm. Lead.* **2019**, *47*, 927–947. [CrossRef]
13. Gao, Y.; Wang, S.; Chen, X. The role of programming self-efficacy in predicting programming performance among non-CS majors. *Comput. Educ.* **2022**, *179*, 104468.
14. Mudambi, R.; Zhao, M. Enhancing student engagement through issue-based teaching: Implications for programming education. *Educ. Train.* **2022**, *64*, 415–429.

15. Locke, E.A.; Latham, G.P. *A Theory of Goal Setting & Task Performance*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1990.
16. Bandura, A. Social cognitive theory of self-regulation. *Organ. Behav. Hum. Decis. Process.* **1991**, *50*, 248–287. [CrossRef]
17. Lim, C.; Kim, H. The relationship between goal setting, self-efficacy, and programming performance: A study of computer science students. *J. Educ. Comput. Res.* **2020**, *58*, 765–785.
18. Ramsden, P. A performance indicator of teaching quality in higher education: The Course Experience Questionnaire. *Stud. High. Educ.* **1991**, *16*, 129–150. [CrossRef]
19. Uçar, S.; Sungur, S. The role of learning satisfaction in predicting academic performance: A study on STEM education. *Int. J. STEM Educ.* **2017**, *4*, 12.
20. Usher, E.L.; Pajares, F. Sources of self-efficacy in school: Critical review of the literature and future directions. *Rev. Educ. Res.* **2008**, *78*, 751–796. [CrossRef]
21. Doménech-Betoret, F.; Abellán-Roselló, L.; Gómez-Artiga, A. Self-Efficacy, Satisfaction, and Academic Achievement: The Mediator Role of Students' Expectancy-Value Beliefs. *Front. Psychol.* **2017**, *8*, 1193. [CrossRef]
22. Webster, J.; Martocchio, J.J. The differential effects of software training previews on training outcomes. *J. Manag.* **1995**, *21*, 757–787. [CrossRef]
23. Ames, C. Classrooms: Goals, structures, and student motivation. *J. Educ. Psychol.* **1992**, *4*, 261–271. [CrossRef]
24. Kim, K.J.; Kwon, B.D. The effects of achievement goals on self-regulated learning, flow, and achievement in an online game-based learning environment. *J. Educ. Comput. Res.* **2012**, *46*, 233–254.
25. Hair, J.F.; Black, W.C.; Babin, B.J.; Anderson, R.E.; Tatham, R.L. *Multivariate Data Analysis*, 6th ed.; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2006.
26. Chin, W.W. The partial least squares approach to structural equation modeling. In *Modern Methods for Business Research*; Marcoulides, G.A., Ed.; Lawrence Erlbaum Associates: Mahwah, NJ, USA, 1998; pp. 295–336.
27. Kock, N.; Hadaya, P. Minimum sample size estimation in PLS-SEM: The inverse square root and gamma-exponential methods. *Inf. Syst. J.* **2018**, *28*, 227–261. [CrossRef]
28. Hair, J.F.; Hult, G.T.M.; Ringle, C.M.; Sarstedt, M. *A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM)*; Sage: Thousand Oaks, CA, USA, 2017.
29. Hair, J.F.; Risher, J.J.; Sarstedt, M.; Ringle, C.M. When to use and how to report the results of PLS-SEM. *Eur. Bus. Rev.* **2019**, *31*, 2–24. [CrossRef]
30. Rigdon, E.E.; Ringle, C.M.; Sarstedt, M. Structural modeling of heterogeneous data with partial least squares. *Rev. Mark. Res.* **2010**, *7*, 255–296.
31. Abdunabi, R.; Hbaci, I.; Nyambe, T. Predicting perceived programming self-efficacy for information system students. In Proceedings of the ISCAP Conference, Albuquerque, NM, USA, 1–4 November 2023; Colorado State University: Fort Collins, CO, USA, 2023.