Factors Affecting Programming Skill Development among Third Year Computer Engineering Students: A Survey Based Study

Joseph Carlos Crisostomo[1], Ramjohn Levy De Mesa[2], Clarence Hernandez[3], Edmund David Tamayo[4], Monaliza Jimenez[5]

Computer Engineering Department, Bulacan State University, Philippines

DOI: https://doi.org/10.47772/IJRISS.2026.100400022

Received: 20 March 2026; Accepted: 04 April 2026; Published: 25 April 2026

ABSTRACT

This study focuses on what shapes programming growth among third-year Computer Engineering learners at Bulacan State University. Psychological traits, classroom structure, personal study routines, alongside AI tool usage and shape skill acquisition patterns observed in these individuals. A numeric method guides this study. The surveys reach 60 selected participants via purposeful selection techniques. Information flows from responses marked on a 4-point Likert scale questionnaire, later analysed using weighted mean, standard deviation, and Pearson correlation. Results show those who believe more strongly in themselves tend to perform better in coding coursework. Despite a positive view of classroom support, learners show hesitation during difficult tasks. When mistakes occur in programming work, discomfort tends to follow. Because understanding code becomes harder, some turn to artificial intelligence systems too frequently. As time passes, personal analytical growth slows under such dependency. Without frequent engagement beyond scheduled lessons, ability to locate faults weakens. Skill advancement depends strongly on individual initiative paired with repeated exercise. Progress requires sustained effort done without guidance from instructors. The study concludes that consistent, self-directed study habits and hands-on practice are critical for skill development. Recommendations include increasing out-of-class coding practice, conducting live skill assessments for future research, and expanding study samples to include other IT-related fields.

Keywords: programming skill development, self-confidence, computer engineering, study habits, artificial intelligence tools

INTRODUCTION

Computer programming has turned into an important skill in higher education and the contemporary job market. Despite its importance, many students struggle to develop programming skills. The top three reasons for the failure in the programming skill examination are insufficient time, self-inefficacy, and unmatched question-time allotment (Bringula et al., 2017). Additionally, unrealistic demands on introductory programming courses are being placed on new learners, creating more learning difficulties for learning skills (Luxton-Reilly et al., 2018). These learning difficulties illustrate that there are not only learning difficulties in programming because of its complexity but also because of learning methods and psychology, and even the education system.

Learning difficulties in developing programming skills may have a negative impact on students' academic performance, motivation, and confidence in pursuing courses of study where computer programming is required.

Though the dropout rates among computing students are not alarmingly high, it is suggested that the difficulty expected for the new learners has been unrealistic rather than intrinsic subject complexity (Luxton-Reilly et al., 2018). Low self-confidence is linked with losing focus as a way to cope with the shortcomings in programming (Bringula et al., 2017). This issue is still applicable to the Philippines recently, where companies are only meeting

55% of the talent demand with a high need for programmers (Watson, 2024). As a result, both students and the national technology sector experience lost opportunities for growth.

The researchers propose to identify and analyze the factors that affect the programming skill development among computer engineering students through a survey-based study. Through the analysis of variables like time management, self-confidence, and studying habits, the research aims to create data that reveals understanding of the students' learning experience. It can be used to assist educators in developing programming courses that are more accommodating, realistic, and facilitatory. It is possible to enhance the students' skill sets and self-confidence through the design of such courses. By managing these variables, skill sets can be bridged, performance can be enhanced, and the need for programmers can be met.

METHODOLOGY

Specifically, the researchers employed the following approaches:

Research Approach

The study used quantitative research methods. It allows the researchers to collect quantitative data and use statistical tools to identify relationships, patterns, and developments among variables.

Participants

A purposive sample comprising of 60 third-year computer engineering students from Bulacan State University's main campus. Only third-year computer engineering students were included because they have sufficient experience in programming subjects and can provide reliable responses related to the study.

Data Collection Instruments

The researchers used an online survey questionnaire, which the researcher created, to collect the necessary data. It will be given to the students through Google Forms. The questionnaire is designed to find out about the factors affecting programming skill development among computer Engineering students. At the start of the online survey forms, the respondent can immediately see the Informed Consent page which provides study participants with information about their rights to control their personal data before they enter the survey. The research objectives are fully described in this section which confirms that participation is optional while maintaining complete confidentiality according to the Data Privacy Act. The survey requires participants to confirm their agreement by selecting the "I Agree checkbox" which will allow researchers to obtain complete participant consent for data collection.

Instrument Reliability and Validity

To ensure the reliability of the research instrument, internal consistency testing was conducted using Cronbach's alpha. The computed Cronbach's alpha coefficient for the questionnaire was 0.82, indicating good reliability and acceptable internal consistency among the survey items. A computer engineering professional examined and verified the survey questions to make sure they were appropriate, relevant, and clear in relation to the study's objectives.

Data Analysis

The gathered data was organized and encoded using Microsoft Excel. Likert-scale replies were given numerical values. To ensure consistency in interpretation, items with negative wording were evaluated in reverse.

The statistical instruments utilized were as follows:

Frequency and Percentage – used to evaluate the demographic profile and category replies of the respondents.

Weighted Mean – used to calculate the average level of agreement about factors connected to programming.

Standard Deviation – to measure variability in responses

Pearson (r) – to determine the relationship between selected variables, such as self-confidence and programming performance

All statistical tests were conducted using a 0.05 level of significance. Relationships with p-values below 0.05 will be considered significant.

Ethical Considerations

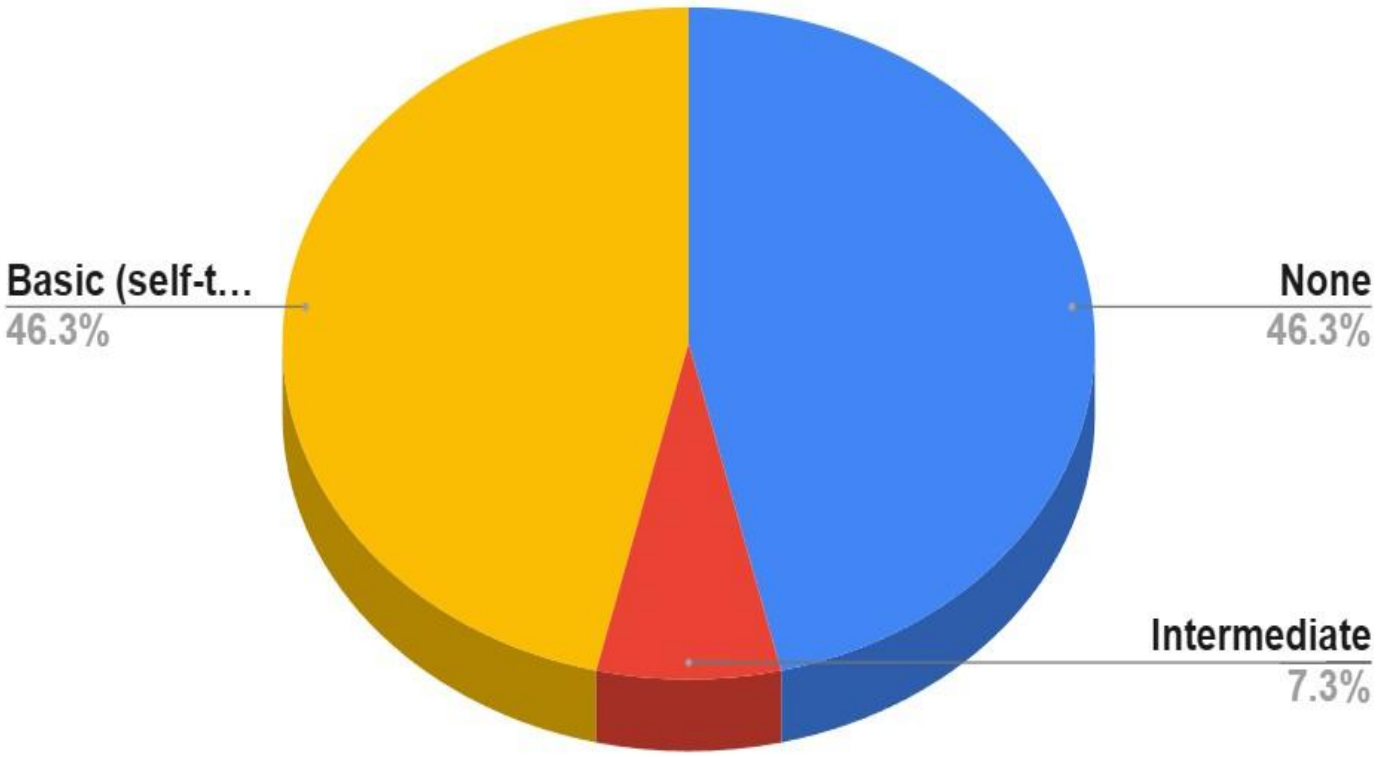
The study adhered to ethical guidelines. Respondents were guaranteed privacy, and they gave their full consent to participate. No private information was acquired. Every piece of information gathered had been used in compliance with the Data Privacy Act.

RESULTS

Respondents' Academic and Demographic Characteristics

The figure illustrates the foundational background of the respondents regarding their exposure to computer programming before entering their current degree program. This data is crucial for understanding the diverse starting points of students in a single cohort.

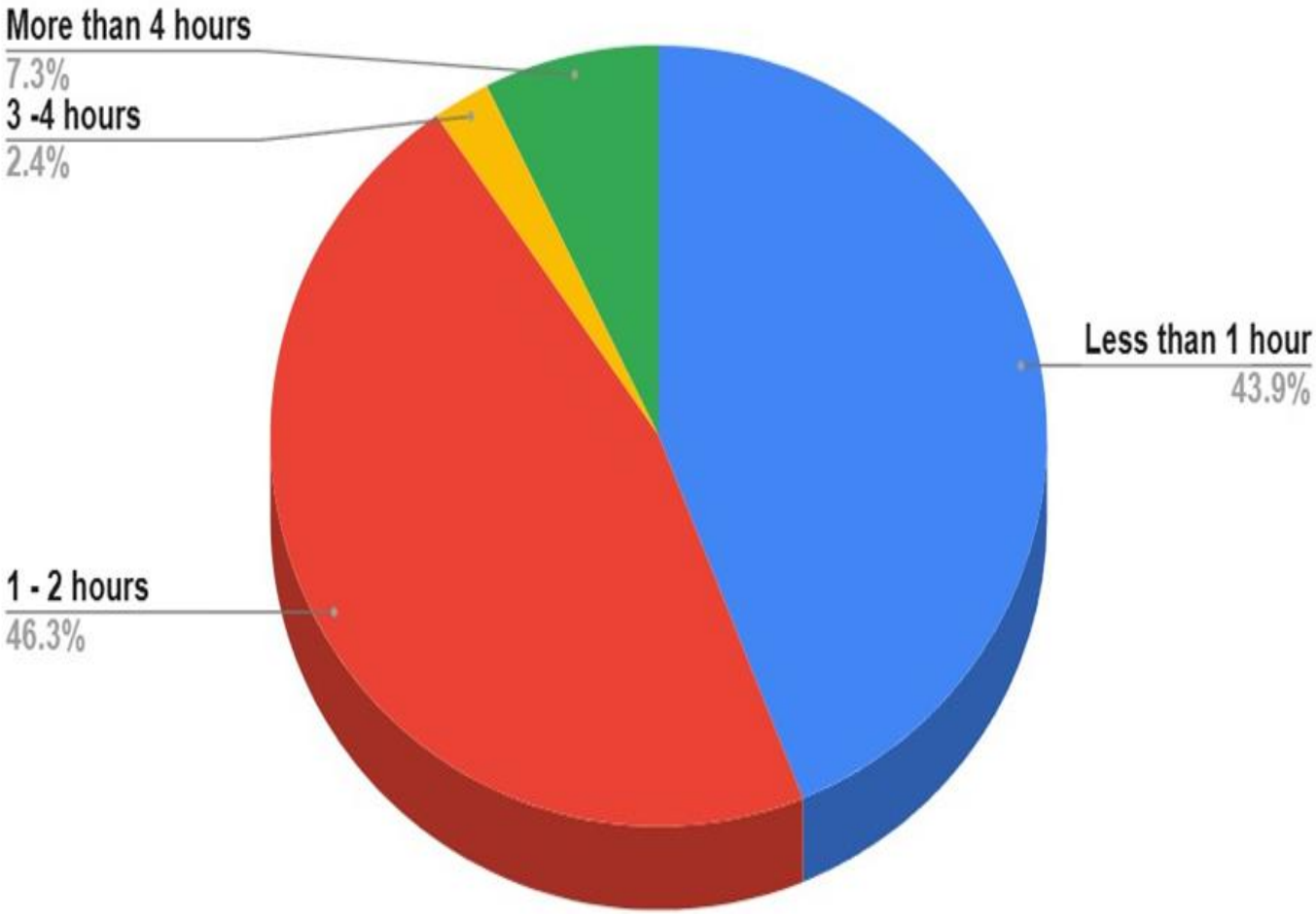
Figure 1: Prior Programming Experience Before College



The data shows an equal split between students with no prior experience (46.34%) and those with basic high school or self-taught backgrounds (46.34%). Only a small minority (7.32%) entered the program with intermediate skills. This suggests that nearly half of the 3rd-year population started their degree with a significant knowledge gap, which aligns with the literature stating that unrealistic expectations on introductory courses can add difficulty for new learners.

The figure provides a detailed breakdown of how students perceive their own capabilities and the external academic pressures they face. It tracks responses ranging from "Strongly Disagree" to "Strongly Agree" regarding time allotment and course expectations.

Figure 2: Average Daily Time Accommodated for Coding Practice

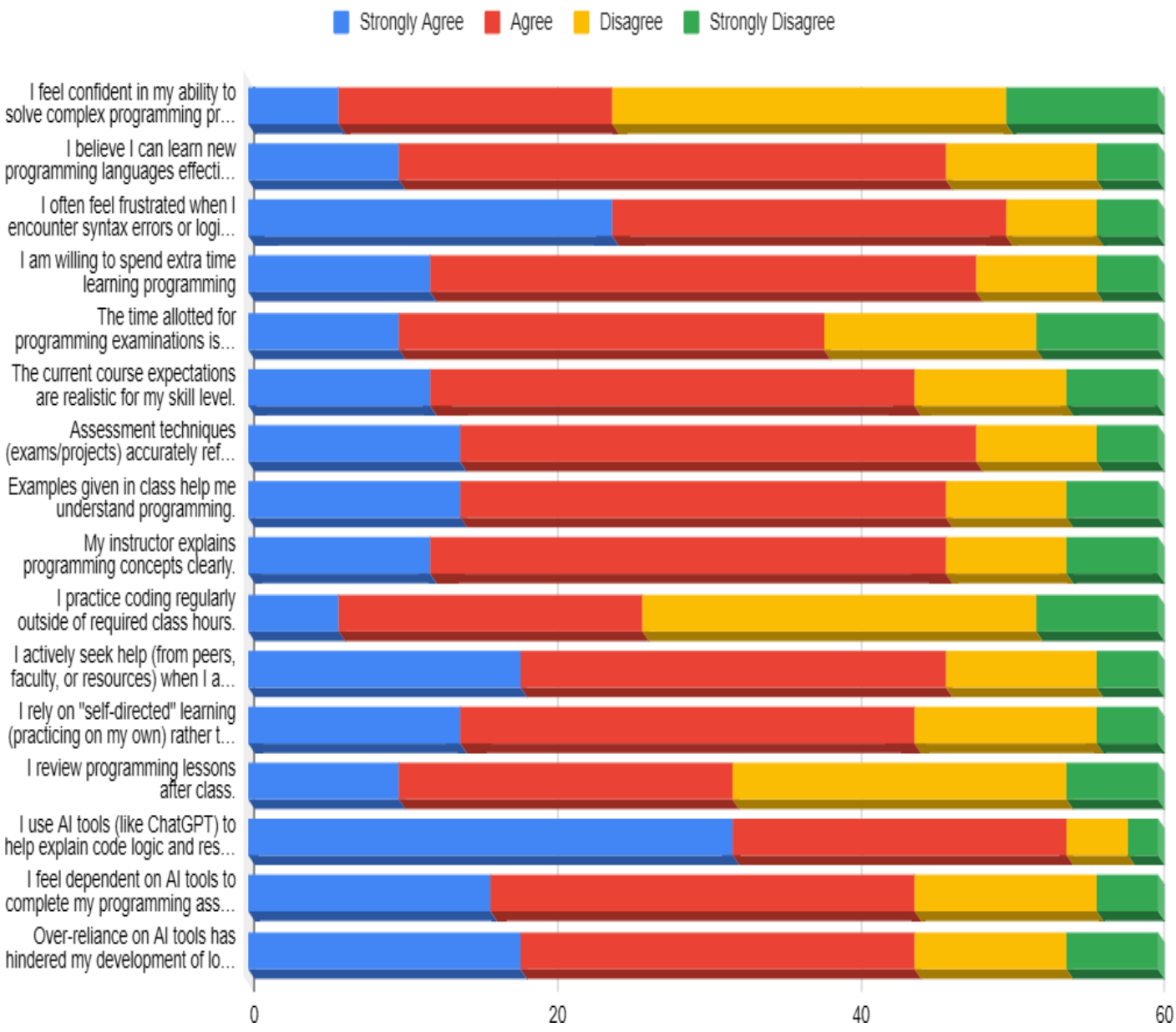


The results highlight a significant tension between academic requirements and student readiness. For instance, when asked if the time allotted for programming examinations is sufficient for the difficulty of the questions, the responses were widely distributed across the scale. This indicates that while some students feel they can cope with the allocated time, many students are struggling to cope with the requirements of a time-bound examination for a coding exam. This information is also in line with the aim of the research, which is to determine if existing examination methods are realistic given the current level of skill attained by third-year students.

Summary of Likert Scores

The figure below indicates a "Summary Distribution of Likert Scores" for the students involved in the survey. This information provides a visual representation of a tension that exists between academic requirements and levels of readiness for the students involved. By providing a visual representation of responses obtained across a range of questions asked, it provides a framework for analyzing the various academic, psychological, and behavioral determinants that impact the development of skill in programming for the students involved.

Figure 3: Summary Distribution of Likert Scores



Shown in Figure 3 is the spread of responses given by students on a four-level agreement scale. This particular arrangement uses values from one to four, where higher numbers indicate stronger alignment with statements posed. A score of four stands for full endorsement, three signals general consent, two reflects doubt, while one shows clear rejection. Scattered patterns appear when examining whether test duration matches question complexity in coding exams. What emerges is a mismatch - expectations set by coursework do not align neatly with student preparedness levels observed. Responses stretch across all available options, suggesting no uniform stance among participants. A significant spread in results indicates differing experiences - some learners handle the workload without difficulty, whereas others face challenges under timed coding tasks. Such patterns support the study's central aim: evaluating if present evaluation methods match the actual abilities of third-year pupils.

Presented here is a summary of responses gathered through a Likert-type questionnaire given to 60 learners in their third year of Computer Engineering study. Numbers shown reflect precise counts of replies, organized by theme. Under examination are four domains: elements tied to mental state, aspects linked to schooling and course layout, routines related to learning practices, along with effects stemming from artificial intelligence applications. Each value corresponds directly to how often a particular answer was selected within these categories. Data remains unchanged from original collection, reported without adjustment or interpretation.

Table 1. Tabulated Responses for Likert Scale

Questions	Strongly Agree	Agree	Disagree	Strongly Disagree	Total
A. Psychological Factors (Self-Confidence & Efficacy)					
I feel confident in my ability to solve complex programming problems	6	18	26	10	60
I believe I can learn new programming languages effectively	10	36	10	4	60
I often feel frustrated when I encounter syntax errors or logic bugs.	24	26	6	4	60
I am willing to spend extra time learning programming	12	36	8	4	60
B. Academic Factors& Course Design					
The time allotted for programming examinations is sufficient for the difficulty of the questions	10	28	14	8	60
The current course expectations are realistic for my skill level.	12	32	10	6	60
Assessment techniques (exams/projects) accurately reflect my programming ability.	14	34	8	4	60
Examples given in class help me understand programming.	14	32	8	6	60
My instructor explains programming concepts clearly.	12	34	8	6	60
C. Study Habits & Methods					
I practice coding regularly outside of required class hours.	6	20	26	8	60

I actively seek help (from peers, faculty, or resources) when I am stuck.	18	28	10	4	60
I rely on "self-directed" learning (practicing on my own) rather than just listening to lectures.	14	30	12	4	60
I review programming lessons after class.	10	22	22	6	60
D. Impact of AI Tools					
I use AI tools (like ChatGPT) to help explain code logic and resolve academic issues.	32	22	4	2	60
I feel dependent on AI tools to complete my programming assignments on time.	16	28	12	4	60
Over-reliance on AI tools has hindered my development of logical reasoning skills.	18	26	10	6	60

With exact numbers shown for each response - “Strongly Agree,” “Agree,” “Disagree,” and “Strongly Disagree” - per statement, clarity emerges on where agreement or split exists within the group. Where responses accumulate, patterns appear without assumption. For every line in the table, counts stand visible, leaving little room for interpretation drift. Because values are laid out plainly, what follows in analysis rests on something measurable. Following this, trends tied to challenges like unclear code structure, independent study habits, or reliance on artificial intelligence gain clearer context. Precision begins here, not later. What individuals selected becomes the base, not opinion. From these figures, conclusions about growth in coding ability draw closer to evidence than guesswork. Numbers do not smooth edges; they expose them. Insight grows where data sits undisturbed.

Factors Affecting Programming Skill Development Legend

The table outlines the interpretation guide for the 4-point Likert scale used in the survey instrument.

Table 2: Likert Scale Interpretation

Scale	Range	Verbal Interpretation
4	3.26 – 4.00	Strongly Agree
3	2.51 – 3.25	Agree
2	1.76 – 2.50	Disagree
1	1.00 – 1.75	Strongly Disagree

It defines the specific numerical ranges corresponding to the verbal interpretations of "Strongly Agree," "Agree," "Disagree," and "Strongly Disagree". This scale serves as the standard for translating the respondents' calculated weighted means into descriptive categories, allowing the researchers to accurately interpret the overall sentiment of the computer engineering students regarding their programming skill development.

Psychological Factors (Self-Confidence & Efficacy)

Table 3 presents the recalculated values, including weighted means and standard deviations, for the psychological factors influencing the students' programming skill development.

Table 3: Psychological Factors

Statement	WM	SD	Interpretation
I feel confident in my ability to solve complex programming problems	2.33	1.01	Disagree
I believe I can learn new programming languages effectively	2.87	0.82	Agree
I often feel frustrated when I encounter syntax errors or logic bugs.	3.17	0.88	Agree
I am willing to spend extra time learning programming	2.93	0.85	Agree
Overall Mean	2.83	0.89	Agree

The data indicates a noticeable gap between students' confidence and their willingness to learn programming. While respondents generally agree that they can learn new programming languages effectively (WM: 2.87, SD: 0.82) and are willing to spend additional time learning (WM: 2.93, SD: 0.85), they still disagree when it comes to confidence in solving complex programming problems (WM: 2.33, SD: 1.01). Furthermore, students reported a high level of frustration when encountering syntax errors and logic bugs (WM: 3.17, SD: 0.88), suggesting that these challenges act as psychological barriers. Overall, despite a positive attitude toward learning, limited confidence in problem-solving remains a key issue affecting programming skill development.

Academic Factors & Course Design

The table details the statistical results concerning academic factors and course design, specifically evaluating how the students perceive their learning environment and assessments.

Table 4: Academic Factors

Statement	WM	SD	Interpretation
The time allotted for programming examinations is sufficient for the difficulty of the questions	2.67	1.02	Agree
The current course expectations are realistic for my skill level.	2.83	0.94	Agree
Assessment techniques (exams/projects) accurately reflect my programming ability.	2.97	0.86	Agree

Examples given in class help me understand programming.	2.90	0.92	Agree
My instructor explains programming concepts clearly.	2.87	0.91	Agree
Overall Mean	2.85	0.93	Agree

The findings show that respondents generally perceive the academic environment as supportive and aligned with their learning needs. All indicators fall under the "Agree" category, including the sufficiency of exam time (WM: 2.67, SD: 1.02), realism of course expectations (WM: 2.83, SD: 0.94), and clarity of instruction (WM = 2.87, SD = 0.91). Notably, students believe that assessment techniques accurately reflect their programming ability (WM: 2.97, SD: 0.86). These results suggest that instructional design and teaching strategies are generally effective; however, variability in responses indicates that some students may still experience difficulty coping with academic demands.

Study Habits & Methods

The table summarizes the data related to the study habits and methods utilized by the students, measuring aspects like self-directed learning and practice frequency.

Table 5: Study Habits

Statement	WM	SD	Interpretation
I practice coding regularly outside of required class hours.	2.40	0.95	Disagree
I actively seek help (from peers, faculty, or resources) when I am stuck.	3.00	0.93	Agree
I rely on "self-directed" learning (practicing on my own) rather than just listening to lectures.	2.90	0.90	Agree
I review programming lessons after class.	2.60	0.97	Agree
Overall Mean	2.73	0.94	Agree

The results reveal inconsistencies in students' study behaviors. While respondents agree that they actively seek help when needed (WM: 3.00, SD: 0.93) and engage in self-directed learning (WM: 2.90, SD: 0.90), they disagree with practicing coding regularly outside of class hours (WM: 2.40, SD: 0.95). Additionally, reviewing lessons after class shows only moderate agreement (WM: 2.60, SD: 0.97). This indicates that although students demonstrate initiative in seeking support, the lack of consistent hands-on practice may hinder the development of programming proficiency, particularly in problem-solving and debugging.

Impact of AI Tools

The table displays the statistical findings regarding the impact of AI tools, focusing on student usage, dependency, and the perceived effects on their reasoning skills.

Table 6: Impact of AI Tools

Statement	WM	SD	Interpretation
I use AI tools (like ChatGPT) to help explain code logic and resolve academic issues.	3.40	0.75	Strongly Agree
I feel dependent on AI tools to complete my programming assignments on time.	2.93	0.94	Agree
Over-reliance on AI tools has hindered my development of logical reasoning skills.	2.93	0.99	Agree
Overall Mean	3.09	0.89	Agree

The data highlights a strong reliance on artificial intelligence tools among students. Respondents strongly agree that they use AI tools to assist with understanding code and completing tasks (WM: 3.40, SD: 0.75). At the same time, they acknowledge a level of dependency (WM: 2.93, SD: 0.94) and agree that over- reliance may negatively affect their logical reasoning skills (WM: 2.93, SD: 0.99). These findings suggest that while AI tools provide significant academic support, excessive use may limit the development of independent problem-solving abilities over time.

Here, Standard Deviation (SD) reflects variation between individual responses and the average opinion. When values are small, agreement is strong - most third-year Computer Engineering learners see themselves in a similar light regarding skill growth and evaluation trust. On the contrary, larger spreads point to differing realities: some struggle with intricate tasks or fear reliance on artificial intelligence, yet others show resilience. Such differences imply shared patterns coexist alongside personal histories - backgrounds where nearly half report no prior exposure, another near-half cite minimal training - that shape distinct paths through identical coursework. Though group tendencies emerge, uneven preparation ensures outcomes remain far from uniform across students.

The table presents the statistical analysis of the relationship between the 3rd-year computer engineering students' self-confidence in programming and their academic performance.

Table 7: Relationship Between Students' Self-Confidence and Academic Performance

Variables Correlated	Pearson r Value	p-value	Decision at 0.05 Level	Verbal Interpretation
Self-Confidence vs. Academic Performance	0.672	< 0.001	Reject Null Hypothesis	Significant Strong Positive Relationship

The table presents the statistical analysis of the relationship between the third-year computer engineering students' self-confidence in programming and their academic performance. Based on the computed Pearson correlation coefficient, a value of r = 0.672 was obtained, indicating a strong positive relationship between the two variables. This means that students who exhibit higher levels of self-confidence in their programming abilities are more likely to achieve better academic performance in their programming-related subjects.

The positive direction of the correlation suggests that as self-confidence increases, academic performance tends to improve correspondingly. Students who believe in their ability to understand programming concepts, solve coding problems, and overcome challenges are more likely to engage actively in learning tasks, persist through difficulties, and perform well in assessments. On the other hand, students with lower confidence levels may hesitate when facing complex problems, which can negatively affect their academic outcomes.

Furthermore, the computed p-value is less than 0.05, which is below the level of significance set for the study. This result leads to the rejection of the null hypothesis, confirming that the relationship between self-confidence and academic performance is statistically significant. Therefore, the findings provide sufficient evidence to conclude that self-confidence is an important factor influencing students' success in programming courses.

Self-belief shapes how long learners persist, how driven they appear, what abilities they build. This idea ties closely to Bandura's 1986 model. Seen through another lens, results echo recent observations in Frontiers in Education (2025): those confident in their problem-solving logic tend to achieve higher outcomes when learning code.

DISCUSSION

The findings of this study provide deeper insights into the factors influencing programming skill development among Computer Engineering students, particularly highlighting the significant role of psychological, academic, and behavioral variables.

The most notable finding is the strong positive correlation between self-confidence and academic performance ($r = 0.672$, $p < 0.001$). This supports the theory of Albert Bandura, which emphasizes that self-efficacy plays a crucial role in shaping students' motivation, persistence, and academic success. Students who believe in their ability to solve programming problems are more likely to engage actively in learning and perform better in assessments.

Despite the generally positive perception of the academic environment, the findings reveal a gap between instructional support and actual student capability. While students agreed that instructors explain concepts clearly and that assessments reflect their abilities, many still reported difficulty in solving complex programming problems and frequent frustration with syntax errors. This indicates that even with effective teaching, students may struggle without sufficient confidence and hands-on experience.

Study habits also play a critical role in programming skill development. Although students reported engaging in self-directed learning, the results showed that many do not practice coding regularly outside required class activities. This inconsistency limits the development of problem-solving and debugging skills, which are essential for programming mastery. Regular practice is necessary to reinforce theoretical knowledge and improve coding fluency.

Furthermore, the use of artificial intelligence (AI) tools presents both advantages and challenges. Students acknowledged that AI tools help them understand concepts and complete programming tasks more efficiently. However, they also recognized that excessive dependence on these tools may weaken their independent problem-solving skills. This indicates the need for balanced and guided use of AI in programming education.

Overall, the results indicate that programming skill development is influenced by a combination of psychological readiness, consistent practice, and responsible use of technological tools. While academic support systems are generally effective, they must be complemented by strategies that strengthen student confidence and promote active engagement in coding practice.

CONCLUSION

This study concludes that programming skill development among Computer Engineering students is significantly influenced by psychological, academic, and behavioral factors, with self-confidence emerging as the most critical determinant of academic performance. The strong positive correlation ($r = 0.672$, $p < 0.001$)

confirms that students with higher levels of self-efficacy are more likely to succeed in programming courses.

Although the academic environment is generally perceived as supportive, the findings reveal that many students continue to struggle with complex programming tasks due to low confidence and insufficient practical experience. Inconsistent coding practice outside classroom requirements further limits the development of essential problem-solving and debugging skills.

Additionally, while artificial intelligence tools provide substantial support in learning programming concepts, excessive reliance on these technologies may negatively affect students' ability to think critically and solve problems independently.

Therefore, programming skill development should not rely solely on instructional design but must also focus on enhancing students' self-efficacy, promoting consistent hands-on practice, and encouraging the responsible use of AI tools. Addressing these factors is essential to improving student performance and preparing graduates to meet industry demands.

RECOMMENDATION

To optimize the study habits of the learners and enhance the course materials used by the educators, several recommendations are proposed. These recommendations aim to improve the scope of the study and enhance the programming skills and self-efficacy of the learners:

- Include the other fields like information technology and computer science to make the sample of the study broader.
- Utilize probability sampling methods in future studies.
- Practice consistently by doing coding outside class activities to expand the programming comprehension.
- Measure the program proficiency via exams or live skill assessments to provide more accurate results.
- Include other factors like environmental factors and availability of equipment to make better limitations on the study.
- Develop guidelines for the responsible use of AI tools to prevent over-dependence.

REFERENCES

1. Bhandari, P. (2023, June 22). What is quantitative research? | Definition, uses & methods. Scribbr. https://www.scribbr.com/methodology/quantitative-research/
2. Bringula, R. P., Aviles, A. D., Batalla, M. Y. C., Borebor, M. T. F., Uy, M. a. D., & Diego, B. E. S. (2017). Factors affecting failing the programming skill examination of computing students. International Journal of Modern Education and Computer Science, 9(5), 1–8. https://doi.org/10.5815/ijmecs.2017.05.01
3. Dwiantoro, B., Sujana, Y., & Hatta, P. (2025). The role of artificial intelligence in programming education and its impact on the learning process. Indonesian Journal of Informatics Education, 9(1). https://doi.org/10.20961/ijie.v9i1.103884
4. Farooq, U., & Anwar, S. (2024). Students motivation to learn programming: A systematic review. In Proceedings of the IEEE Frontiers in Education Conference. https://doi.org/10.1109/FIE61694.2024.10893105
5. Gurer, M. D., & Tokumaci, S. (2020). Factors affecting engineering students' achievement in computer programming. International Journal of Computer Science Education in Schools, 3(4), 23–34. https://doi.org/10.21585/ijcses.v3i4.74

6. Harimurti, R., Ekohariadi, E., Munoto, M., B, I. G. P. A., & Winanti, E. T. (2019). Analysis of programming skills concept in developing problem solving skills. Jurnal Pendidikan Teknologi Dan Kejuruan, 25(1), 43–51. https://doi.org/10.21831/jptk.v25i1.22638
7. Kuo, Y., & Kuo, Y. (2025). Learning programming: exploring the relationships of self-efficacy, computational thinking, and learning performance among minority students. Frontiers in Education, 10. https://doi.org/10.3389/feduc.2025.1623415
8. Luxton-Reilly, A., Simon, N., Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: a systematic literature review. Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ACM ITiCSE 2018 Companion)., 55–106. https://doi.org/10.1145/3293881.3295779
9. Misa. (2025, July 2). Significance of Quantitative Research : 9 Essential insights. Researchmate.net. https://researchmate.net/quantitative-research/
10. Sitompul, S. S., Kustono, D., Suhartadi, S., & Setyaningsih, R. M. (2017). The relationship of the learning of tourism marketing, hard skills, soft skills and working quality of the graduates of Tourism Academy in Medan. International Journal of Social Sciences and Educational Studies, 3(4). https://doi.org/10.23918/ijsses.v3i4p124
11. Sweller, J. (1988b). Cognitive load during problem solving: Effects on learning. Cognitive Science, 12(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4
12. Watson, M. (2024, October 11). Job prospects for programmers in the Philippines. Full Scale. https://fullscale.io/blog/job-prospects-for-programmers-in-the-philippines/