Jiang, B. et al. (Eds.) (2025). Proceedings of the 33rd International Conference on Computers in Education. Asia-Pacific Society for Computers in Education

The Effects of Difficulty-Based Question Adaptivity in Formative Assessment on Introductory Programming Learning

Jagadeeswaran THANGARAJ[a*], Monica WARD[a] & Fiona O'RIORDAN[b]
[a]*School Of Computing, Dublin City University, Ireland*
[b]*Teaching and Learning Unit, CCT College Dublin, Ireland*
*Jagadeeswaran.Thangaraj2@mail.dcu.ie

Abstract: Introductory programming is an essential component of computer science education, yet many students find it challenging. An important factor in students' success in programming courses is their motivation and self-confidence. Numerous formative assessment systems have been designed to increase students' self-confidence. These systems are capable of automatically evaluating student work and delivering immediate feedback. Students gain confidence as a result of formative assessment's automatic feedback, which enables them to better evaluate their own work and make improvements. Many formative assessment systems ask the same questions of students with varying skill levels. Therefore, the objectives of this paper are: first, to introduce an adaptive strategy in formative assessment that poses the questions using difficulty levels; second, to determine whether this adaptive formative assessment framework can help students feel more confident by introducing common code errors, and third, to evaluate how well students performed following an adaptive formative assessment activity. An experimental study with 62 undergraduate Non-CS participants (novices) was conducted to accomplish these objectives. In-depth analyses were performed on the self-rated confidence, test results and completion time. According to the findings, novices feel significantly better confident after taking the adaptive formative assessment quizzes. However, the self-rated confidence declines over the advanced topics of programming. Additionally, students who took part in the revision adaptive formative assessment exercise performed better on their test. Findings showed that employing difficulty-based question adaptivity in formative assessment was more likely to motivate students and increase their self-confidence.

Keywords: Adaptive assessment, Computer programming, Formative feedback, Introductory programming, Novice students, Self-confidence.

1. Introduction

Introductory programming is an essential subject for students to feel confident about their computer science studies (Rum and Ismail 2017). Novice programmers find difficulty in: grasping the fundamental concepts of programming structure; learning programming language syntax; and identifying errors and troubleshooting program code (Kadar et al., 2021). Consequently, learning programming necessitates practice and simultaneous mastery (Islam et al., 2019). In general, novice programmers are ecstatic to be aware of their progress and the specifics of their errors in order to help guide them in the proper direction, and increase their confidence in their programming abilities (Zhang et al., 2024). Formative assessment is providing feedback to a student with the goal of changing their way of thinking or acting to enhance learning (Shute, 2008). Beyond assessment, it increases novice programmers' self-confidence and meta cognitive awareness (Lishinski and Yadav 2021). Although feedback mechanisms have been well studied, limited is known about novices' confidence in using formative feedback to learn introductory programming (Barra et al., 2020). Another limitation is that all these systems ask the same questions to all students (Luxton-Reilly et al., 2023). Each learner has strengths, limitations, and favorite subject areas within the framework of the learning environment. Adaptive assessment process assesses the students with different abilities with different sets of questions. By starting with easier tasks and progressively adjusting difficulty, systems maintain novices within an optimal challenge zone. Such adaptivity encourages early success, builds confidence, and sustains persistence. Recent advances in Item Response Theory/Computerized Adaptive Testing (IRT/CAT), Bayesian Knowledge Tracing (BKT), and Deep Knowledge Tracing (DKT) provide tools for such design (Sharpnack et al., 2024). These models are gaining attention in recent days and these provide a wide range of advantages in teaching diverse students (Louhab et al., 2018). Integrating embeddings of code edits, unit-test outcomes, and error traces allows the system to deliver granular scaffolds, such as debugging prompts or step-wise guidance (Zhao et al., 2023). BKT offers interpretable mastery estimates for specific knowledge components (KCs). This transparent KC feedback helps novices understand progress and reduces frustration, a key factor in sustaining motivation (Shi et al., 2022). These models allow supporting the students with feedback to make them understand the concepts of computer sciences and prevent from the cold start problem (Velde et al., 2021). According to the learner’s response pattern, adaptive systems allow them to learn new concepts in real time by modifying questions across different levels (Marwan et al., 2020; Vesin et al., 2022). These timely supports keep learners engaged while gradually raising task difficulty. This allows them to overcome cold start problems and gradually refine the difficulty profile and personalize the learning pathway (Pankiewicz, 2021; Velde et al., 2021). Research shows that learners' confidence is a powerful predictor of outcomes including skill development, though cognitive capacity and prior preparation are important factors in programming success (Shi et al., 2022). Thus, the improvement of programming self-confidence should be a top goal when developing curricula and assessments (Zhao et al., 2023). By continuously aligning task complexity with learner readiness, the system fosters confidence, sustains engagement, and supports gradual mastery of programming concepts. The objective of this paper is to use a difficulty based adaptivity in formative assessment to increase students’ level of confidence in learning introductory programming and their actual performance on programming tests. Using the adaptive formative assessment, the following research questions have been investigated:

1. Can adaptive formative assessment improve the self-confidence of novice programmers in:
    - a) accurately predicting the outcomes of fundamental programming concepts?
    - b) effectively identifying and correcting errors in Python programming?
2. How effective is adaptive formative assessment in facilitating novices’ understanding of distinct conceptual components in programming?
3. To what extent can adaptive formative assessment encourage novice learners to improve their programming skills?

2. Difficulty-based Adaptive Formative Assessment

The knowledge level of the learners increased by varying the order of the assessment questions in adaptive approach (Heitmann et al., 2018). Adaptive formative assessments are customized to each student individually based on their responses to previous test items (Papanastasiou, 2021). Self-regulated learning abilities, inventiveness, and self-efficacy of students can be used to adaptively generate assessments that are customized to each student in terms of question difficulty, assessment length, and question types (e.g., multiple choice, fill-in-the-blank, or short response) (Yang et al., 2022). Using adaptive assessment, which organizes a collection of questions into three cognitive levels according to complexity (easy, moderate, and difficult), programming adaptive testing evaluates students’ knowledge in programming courses (Chatzopoulou and Economides, 2010). This helps to create meaningful feedback for students and supports their learning process for different levels of difficulty. Two characteristics were deemed crucial while intending to develop adaptive formative assessments of programming tasks. For error messages to be properly understood, feedback needs to be quick and detailed. Second, students must be given the chance to discover their mistakes after making a number of attempts on different questions. Due to the nature of these elements, it is necessary to create assessments large enough so that students may repeat assessments without encountering the same questions twice.

This paper presents how to create an adaptive formative assessment in an efficient manner so that students can grasp the material and get a motivation of how proficient they are with computer programming. While considering the limitations of automatic assessment, this framework emphasizes the importance of achieving comparable difficulty for questions and maximizes the potential for randomization for exercise tasks. These things could be topics, questions, a variety of errors, etc. The goals relate to questions with varying degrees of difficulty. Bloom's taxonomy offers a well-established cognitive framework that can inform automated question design for programming learning (Fuller et al., 2007). Research indicates that higher-order cognitive tasks, such as those aligned with Bloom's "Apply" level, exhibit higher discrimination indices, meaning they more effectively differentiate between learners of varying abilities (Hamamoto et al., 2020). This supports the categorization of tasks into difficulty levels corresponding to Bloom's taxonomy, facilitating the estimation of item difficulty parameters in IRT models. Difficulties in programming are classified as Bloom's taxonomy of programming (Sobral, 2021). This model categorizes programming tasks into three cognitive levels: Remember (Easy), Understand (Moderate) and Apply (Hard). This research classified a list of questions in three cognitive levels based on the complexity (like easy, moderate, and difficult) (Louhab et al., 2018). The adaptive sequencing mechanism is informed by Knowledge Space Theory (Ihichr et al., 2024) which enabled branching rules based on prerequisite structures among programming concepts. Here easy questions assess the basic concepts, moderate questions assess comprehensive knowledge and difficult questions do the applications of the knowledge (Vie et al., 2017). If a student successfully responds to a moderate question on this assessment when it starts, the subsequent question is hard. If not, the easy questions will be asked as indicated in Figure-1. It goes on until the system forecasts the competency level of the students (Simon-Campbell et al., 2018). This study has led us to create multiple choice question quizzes to introduce common programming errors that are an excellent method to increase student self-confidence by engaging and helping them remember the programming content (Ross et al., 2018).

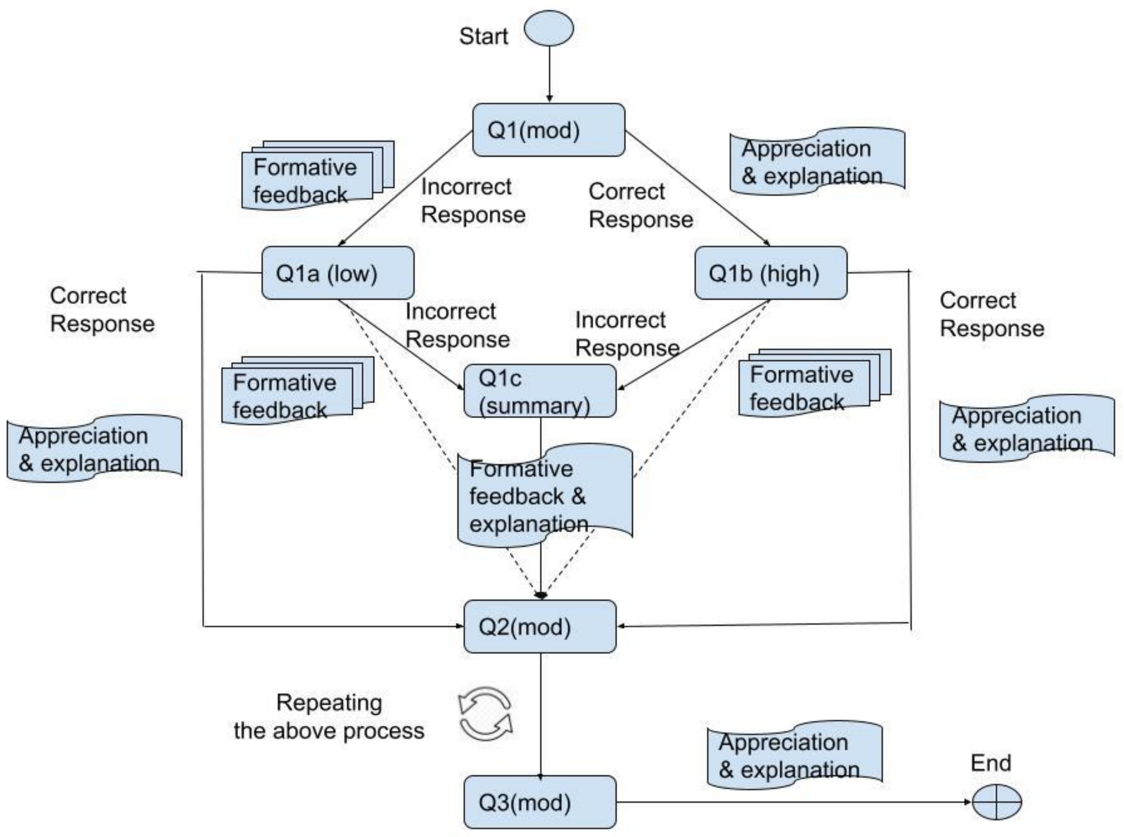

Figure-1. Adaptive strategy

3. Research method and Data collection

To answer the research questions, this research developed a set of quizzes for basic topics of introductory programming. Because of its convenience and syntactical simplicity, Python is a popular programming language used in introductory programming classes (Johnson et al., 2020). The topics covered include variables, operators, conditionals, loops, and a few concepts related to functions (Xinogalos et al., 2020).

Population. Dublin City University's undergraduate students from the Introductory programming and advanced modules were participated. The data includes 252 students' programming quiz attempts of 62 students that they submitted at the end of each quiz session.

Survey Questions. In addition to quiz questions, each quiz included survey questions. At the conclusion of each quiz, they filled out the survey. Four different factors were measured by a survey consisting of 21 items with use of the post-survey attitude of self-confidence.

- Self-confidence in programming understanding (16 items in total)
- Self-confidence of understanding common code errors (3 items in each quiz)
- Quiz difficulty level

4. Results

4.1 Research Question-1

The first research question is, *"Can adaptive formative assessment improve the self-confidence of novice programmers a) accurately predicting the outcomes of fundamental programming concepts b) effectively identifying and correcting errors in Python programming?"*. This question focuses on whether the difficulty-based adaptive formative assessment increases their confidence in their ability to comprehend the basic concepts of programming. In formative assessment, Bloom's difficulty level-based adaptive model is employed to capture relationships between concepts with confidence. We compared the self-confidence levels before and after practicing all of the assessment quizzes. Additionally, we gathered novice students' confidence in four subjects before and following all the quizzes: their ability to learn computer programming, create new programs, comprehend how programs operate, and recognize programming errors. The question was, 'How confident are you in your ability to do the following, using a number between 0 and 10?'.

Table-1. *Comparison of the novices' confidence before and after their quiz attempts*

Confidence in	Mean		Std. Dev		Difference		t
	Pre	Post	Pre	Post	Mean	SD	
Learning computer programming	4.63	6.23	2.21	1.59	1.59	2.60	4.832
Designing new programs	2.74	4.77	1.87	2.01	2.03	2.72	5.879
Understanding how programs operate	3.98	6.05	2.25	1.64	2.06	2.74	5.921
understanding programming errors	4.26	6.19	2.22	1.61	1.93	2.50	6.092

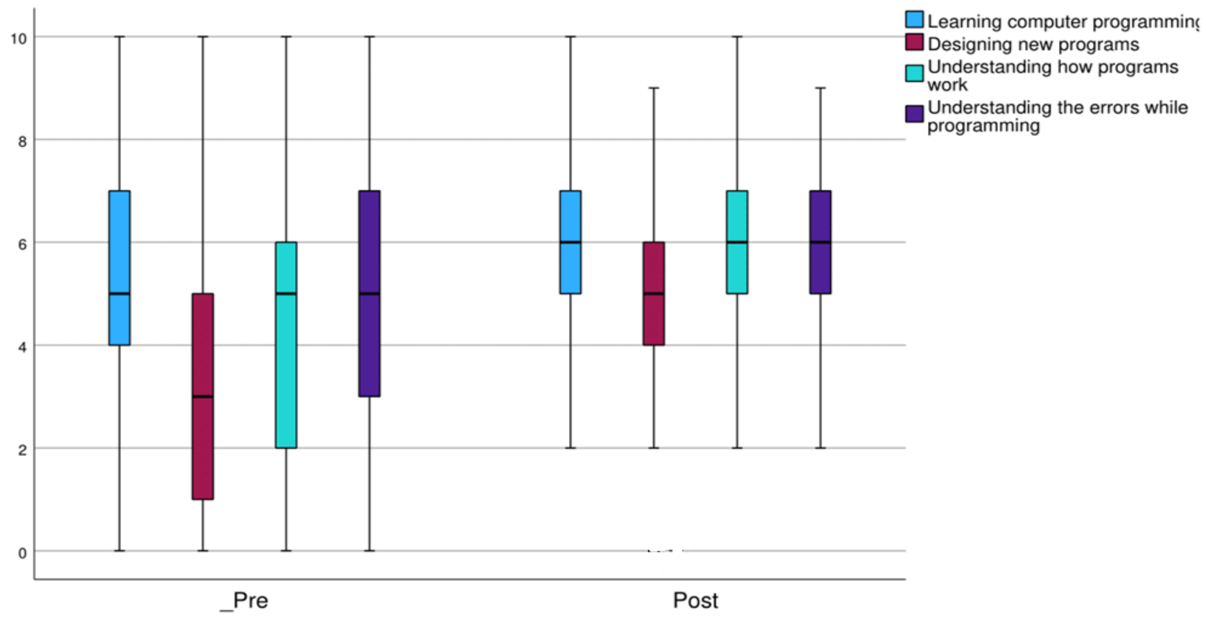

Figure-2. Mean difference on self-confidence

Table-2. *Novices Likert scale responses on Python basic concepts*

No	Question (I know..)	Mean	SD
Q1	how to use print statements in Python programming.	4.21	0.87
Q2	how to use the Python programming variables.	3.58	0.90
Q3	what the operators (+, -, *, /, >,<, =) mean in Python programming.	3.71	1.02
Q4	how to use input functions and arithmetic operations.	3.41	0.86
Q5	how to use variables and their type conversions.	3.54	1.06
Q6	the orders of operator precedence in Python programming.	3.57	0.92
Q7	what the comparative operators (>,<, >=, <=, == & !=) mean.	4.20	0.98
Q8	how to use if/else statements in Python programming.	3.78	1.08
Q9	the syntax and indentation errors while using if/else statements.	3.13	1.05
Q10	what the logical operators (AND, OR, NOT) mean.	4.00	0.97
Q11	how to use a while loop in Python programming.	3.82	1.12
Q12	what process before or after increment or decrement happens in the while loop of Python programming.	3.00	0.94
Q13	how to use functions in Python programming.	2.83	1.13
Q14	how arguments work functions of Python programming.	2.83	1.03
Q15	how to pass arguments in function calls of Python programming.	2.86	1.06
Q16	what are default arguments in functions of Python programming.	2.80	1.13

The mean difference between their confidence levels before and after the quiz attempts is depicted in Figure-2. It demonstrates that their confidence levels have significantly increased. Before the quizzes, the confidence scores ranged from 0 to 10, and after the quizzes, they rose to between 2 and 10. Additionally, the mean value has grown. To determine the impact of adaptive formative assessment in these subjects, a paired-sample *t-test* was used. Table-1 indicates that their confidence levels before and after the quiz exercises differed statistically significantly. There was a statistically significant difference between the pre-quiz (M = 4.63 out of 10, SD = 2.212) and the post-quiz (M = 6.23 out of 10, SD = 1.593) for learning computer programming (t = 4.832, p < 0.001 [two-tailed]). Accordingly, the post-quiz mean was higher by 4.832 with a 95% confidence interval between 0.34 and 0.88. The corresponding *t* values for creating new programs, comprehending how programs work, and recognizing programming errors are 5.879, 5.921, and 6.092 (p < 0.001 [two-tailed]). This indicates that the adaptive formative assessment has significant beneficial differences in these contexts.

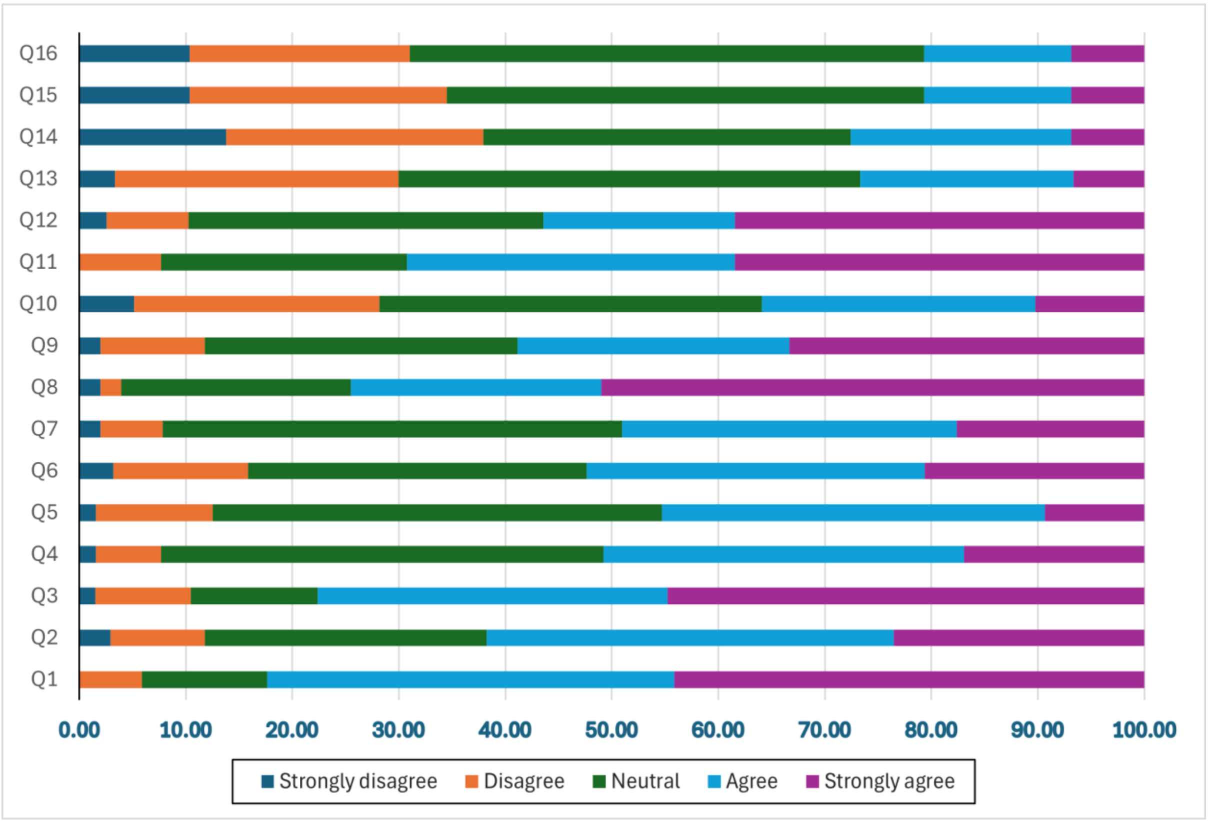

Figure-3. All students' feedback (in %) on understanding errors

After every quiz, we also asked them about their confidence with a list of basic concepts as shown in Table-2. Using a self-rated Likert scale, the responses were accepted that had five possible scores: strongly disagree (1), disagree(2), neutral(3), agree(4), and strongly agree(5). The students completed five quizzes and provided a self-rated confidence response of 252 (82.5%) out of 292 attempts. Their distribution of confidence across the topics shown in the Figure-3. These findings indicate that they feel more confident after they have completed the adaptive formative assessment quizzes covering the fundamental subjects with mean scores of 3.5 or higher. However, the self-rated confidence declines over the advanced topics of programming such as functions as shown in Figure-3. This may reflect students' limited prior exposure to abstract concepts and the cognitive demands of integrating multiple new skills.

4.2 Research Question-2

The second research question is, *How effective is adaptive formative assessment in facilitating novices' understanding of distinct conceptual components in programming?*. We have compiled the percentage of correct answers for different questions of varying complexity between Non-CS and CS cohorts as shown in Table-3. The corresponding graph illustrated in Figure-4 compares how two cohorts performed across five quizzes, broken down by question difficulty (Easy, Moderate, Hard). Analyzing these scores across CS and Non-CS cohorts helps evaluate whether this design is equitable, adaptive, and informative. Non-CS students (represented in dotted lines) are more balanced and consistent, especially in Moderate and Easy questions across most quizzes. Quiz-4 (Loop) is a significant turning point because, as a result of the quiz content playing, Non-CS did better than CS in every category. Up until Quiz-5, Non-CS students perform better than CS students on moderate questions. CS students (represented in continuous lines) consistently perform well on hard questions, particularly in Quiz-5. Quiz-5 might be easier for CS students, given the cluster of 100s.

Table-3. *Correct responses percentage by CS vs Non-CS for Each Quiz and Difficulty Level*

Difficulty	Moderate		Easy		Hard	
Quiz	CS	Non-CS	CS	Non-CS	CS	Non-CS
1	88.89	77.28	58.33	50.00	87.94	84.82
2	90.64	86.00	72.22	78.59	89.68	86.79
3	70.42	88.51	83.33	59.56	71.37	66.99
4	60.50	90.62	50.00	79.17	82.42	89.20
5	90.48	78.68	100.00	69.20	97.14	86.63

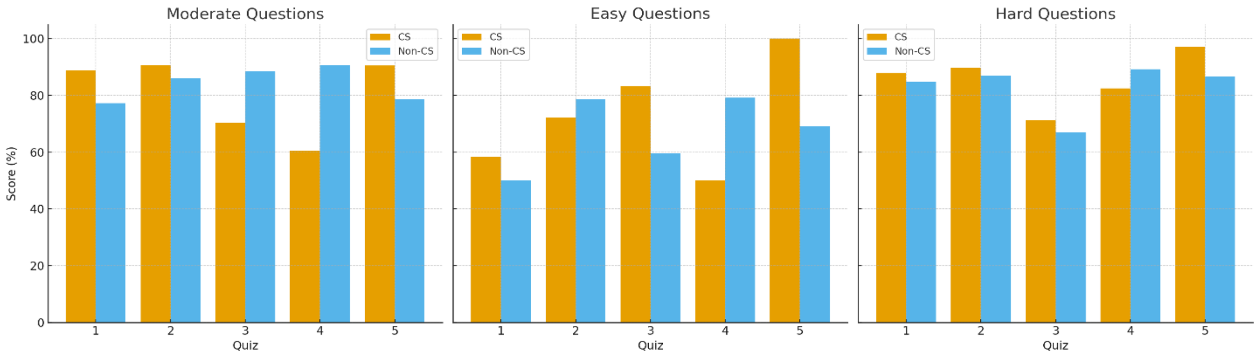

Figure-4. Correct responses percentage by Difficulty levels across quizzes

4.3 *Research Question-3*

The next research question is, *To what extent can adaptive formative assessment encourage novice learners to improve their programming skills?*. To address this research question, we then chose to examine if there were significant variations in scores between the two groups of students who attended and those who did not. Therefore, this study offered a quiz before the exam in order to determine the effect of the adaptive formative assessment on exam scores. Prior to the exam, 84 out of 121 students took the quiz. We examined the differences in exam scores between students who took the quiz and those who did not. To determine the difference in scores between participants and non-participants, an independent *t - test* was used. Table-4 shows the mean difference of the exam score and completion time between participants and non-participants. These findings show that the adaptive quiz significantly influences both scores and completion times between quiz participants and non-participants.

Table-4. *Mean difference between quiz participants and non-participants*

Aspect	**Participant**	**N**	**Mean**	**SD**
Score	Yes	84	6.33	1.500
	No	37	6.05	1.914
Time Taken	Yes	84	15.4014	6.79762
	No	37	16.6420	5.26143

5. Discussion

This study was driven to explore how an adaptive formative assessment might increase novices' confidence in their ability to learn programming. We employed two different kinds of explicit interventions to address the research questions, such as students' self-rated confidence and their actual exam scores following the quizzes. By exploring novices' self-rated confidence and evaluating whether adaptive formative assessment would help them, this study found that there was a statistically significant difference between the pre and post self-confident values. According to the findings, the majority of students entered with high confidence, while just a small percentage entered with low confidence. This implies that the majority of students think that learning from mistakes through adaptive formative assessment aids in their comprehension of fundamental concepts of programming. Additionally, it shows how adaptive assessments improved their comprehension of errors.

Overall, the self-rated confidence, exam score, and completion time have a significant positive correlation. As a conclusion, adaptive formative assessment can be employed to teach novices introductory programming concepts through increasing their self-confidence and helping them learn from their mistakes. However, adaptive formative assessment may produce unintended negative outcomes when difficulty calibration is misaligned—for example, assigning overly simple tasks may reduce engagement, while overly challenging ones can harm motivation and confidence. Additionally, abrupt shifts in task difficulty may impose excessive cognitive load, particularly for novices who are still grappling with foundational concepts. These risks underscore the importance of designing adaptive systems that balance responsiveness with stability. As they can effectively aid in the learning of programming, one approach we proposed in this contribution is to develop formative quizzes with adaptive techniques and difficulty levels. As a result, learning opportunities have expanded, increasing students' confidence, and understanding the common code errors. Because the questions in the evaluation system are only shown dependent on the responses to earlier questions. Therefore, proficient learners do not require more time. It can be a viable teaching and learning tool for introductory programming.

5.1 Limitations & Future work

Here, a key factor in adaptive formative assessment is programming complexity. However, the measure of programming complexity was tied to the adaptive system's classification of questions as easy, moderate, or hard. While this provided a practical framework, it does not fully capture the nuanced cognitive demands of different programming concepts (e.g., syntax vs. algorithmic reasoning, or loops vs. recursion). As a result, some tasks classified at the same difficulty level may not have been experienced equally across students. Second, the study relied on self-reported confidence ratings. Although these measures provide valuable insights into student perceptions, they may not always align with demonstrated competence. Students may overestimate or underestimate their understanding, which introduces the risk of bias in interpreting adaptive assessment outcomes. Third, cohort differences (e.g., CS vs. non-CS students) and prior exposure to programming were not fully controlled. This may explain observed variability in confidence, particularly on advanced topics, where background knowledge could strongly influence both performance and self-assessment. Finally, the adaptive system did not incorporate more sophisticated models such as item response theory or learner modeling to account for differences across cohorts, languages, or problem types. This limits the generalizability of the system's difficulty calibration. Future work could enhance with different question suggestions utilizing other methods, including item response theory (Yang et al., 2022).

Also, Generating questions according to difficulty levels using Artificial Intelligence (AI) tools is also a major potential for adaptive formative assessment in programming. Some researchers focus on leveraging AI to create programming questions (Doughty et al., 2024). The ability to classify the questions according to their level of difficulty is lacking, though. As a result, this study recommends using AI to categorize the questions based on their degree of difficulty.

References

Barra, E., López-Pernas, S., Alonso, , Sánchez-Rada, J. F., Gordillo, A., & Quemada, J. (2020). Automated assessment in programming courses: A case study during the covid-19 era. *Sustainability*, *12*(18). Retrieved from https://www.mdpi.com/ 2071-1050/12/18/7451 doi: 10.3390/su12187451

Chatzopoulou, D., & Economides, A. (2010, 08). Adaptive assessment of student's knowledge in programming courses. *J. Comp. Assisted Learning*, *26*, 258-269. doi: 10.1111/j.1365-2729.2010.00363.x

Doughty, J., Wan, Z., Bompelli, A., Qayum, J., Wang, T., Zhang, J., ... & Sakr, M. (2024). A comparative study of AI-generated (GPT-4) and human-crafted MCQs in programming education. In *Proceedings of the 26th Australasian Computing Education Conference* (pp. 114-123).

Fuller, U., et al. (2007). Bloom's taxonomy in computing education. *ACM SIGCSE Bulletin, 39*(3), 10-14.

Hamamoto Filho, P. T., Silva, E., Ribeiro, Z. M. T., Hafner, M. L. M. B., Cecilio-Fernandes, D., & Bicudo, A. M. (2020). Relationships between Bloom's taxonomy, judges' estimation of item difficulty and psychometric properties of items from a progress test: a prospective observational study. *Sao Paulo medical journal - Revista paulista de medicina*, *138*(1), 33–39. https://doi.org/10.1590/1516-3180.2019.0459.R1.19112019

Heitmann, S., Grund, A., Berthold, K., Fries, S., & Roelle, J. (2018). Testing is more desirable when it is adaptive and still desirable when compared to note-taking. *Front. Psychol.*, *9*.

Ihichr, A., Oustous, O., Idrissi, Y., & Ait Lahcen, A. (2024). A systematic review on assessment in adaptive learning: Theories, algorithms and techniques International Journal of Advanced Computer Science and Applications, 15. https://doi.org/10.14569/IJACSA.2024.0150785

Islam, N., Sheikh, G., Fatima, R., & Alvi, F. (2019, 10). A study of difficulties of students in learning programming. *Journal of Education & Social Sciences*, *7*, 38-46. doi: 10.20547/ jess0721907203

Johnson, F., McQuistin, S., & O'Donnell, J. (2020). Analysis of student misconceptions using python as an introductory programming language. In *Proceedings of the 4th conference on computing education practice.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3372356.3372360 doi: 10.1145/3372356.3372360

Kadar, R., Wahab, N., Othman, J., Shamsuddin, M., & Mahlan, S. (2021). A study of difficulties in teaching and learning programming: A systematic literature review. International Journal of Academic Research in Progressive Education and Development, 10. https://doi.org/10.6007/IJARPED/v10-i3/11100

Lishinski, A., & Yadav, A. (2021). Self-evaluation interventions: Impact on self-efficacy and performance in introductory programming. *ACM Transactions on Computing Education (TOCE)*, *21*, 1 - 28.

Louhab, F. E., Bahnasse, A., & Talea, M. (2018). Towards an adaptive formative assess ment in context-aware mobile learning. *Procedia Computer Science*, *135*, 441-448. (The 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life) doi: https://doi.org/10.1016/j.procs.2018.08.195

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., . . . Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings companion of the 23rd annual acm conference on innovation and technology in computer science education* (p. 55–106). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3293881.3295779 doi: 10.1145/3293881.3295779

Luxton-Reilly, A., Tempero, E., Arachchilage, N., Chang, A., Denny, P., Fowler, A., . . . Ye, X. (2023, 01). Automated assessment: Experiences from the trenches. In (p. 1-10). doi: 10.1145/3576123.3576124.

Marwan, S., Gao, G., Fisk, S., Price, T. W., & Barnes, T. (2020). Adaptive immediate feedback can improve novice programming engagement and intention to persist in computer science. In *Proceedings of the 2020 acm conference on international computing education research* (p. 194–203). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3372782.3406264 doi: 10.1145/3372782.3406264.

Mohammed, M., & Omar, N. (2020). Question classification based on Bloom's taxonomy cognitive domain using modified TF-IDF and word2vec. *PloS one*, *15*(3), e0230442. https://doi.org/10.1371/journal.pone.0230442.

Pankiewicz, M. (2021). Assessing the cold start problem in adaptive systems. In *Proceedings of the 26th acm conference on innovation and technology in computer science education v. 2* (p. 650). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3456565.3460053 doi: 10.1145/3456565.3460053.

Papanastasiou, E. (2021). Adaptive assessment. In R. Gunstone (Ed.), *Encyclopedia of science education* (pp. 1–2). Dordrecht: Springer Netherlands. Retrieved from https://doi .org/10.1007/978-94-007-6165-0 3-4 doi: 10.1007/978-94-007-6165-0_3-4.

Ross, B., Chase, A.-M., Robbie, D., Oates, G., & Absalom, Y. (2018). Adaptive quizzes to increase motivation, engagement and learning outcomes in a first year accounting unit. *International Journal of Educational Technology in Higher Education*, *15*(1), 30.

Rum, S. N. M., & Ismail, M. A. B. (2017). Metacognitive Support Accelerates Computer Assisted Learning for Novice Programmers. *J. Educ. Technol. Soc.*, *20*, 170-181. Shute, V. J. (2008). Focus on Formative Feedback. *Review of Educational Research*, *78*(1), 153-189. doi: 10.3102/0034654307313795.

Sharpnack, J., Hao, K., Mulcaire, P., Bicknell, K., LaFlair, G., Yancey, K. & von Davier, A.A. (2025). BanditCAT and AutoIRT: Machine Learning Approaches to Computerized Adaptive Testing and Item Calibration. Proceedings of Large Foundation Models for Educational Assessment, 264:121-135 https://proceedings.mlr.press/v264/sharpnack25a.html.

Shi, Y., Pardos, Z., & Khanna, R. (2022). A code-based knowledge tracing model for programming. In Proceedings of the 15th International Conference on Educational Data Mining (EDM), 266–277.

Simon-Campbell, E., & Phelan, J. (2018). Effectiveness of an adaptive quizzing system as a self-regulated study tool to improve nursing students' learning. *International Journal of Nursing & Clinical Practices*, *5*. doi: 10.15344/2394-4978/2018/290.

Sobral, S. (2021). Bloom's taxonomy to improve teaching-learning in introduction to programming. *International Journal of Information and Education Technology*, *11*, 148- 153. doi: 10.18178/ijiet.2021.11.3.1504.

Velde, M., Sense, F., Borst, J., & Rijn, H. (2021). Alleviating the cold start problem in adaptive learning using data-driven difficulty estimates. *Computational Brain & Behavior*, *4*. doi: 10.1007/s42113-021-00101-6.

Vesin, B., Mangaroska, K., Akhuseyinoglu, K., & Giannakos, M. (2022). Adaptive assessment and content recommendation in online programming courses: On the use of elo-rating. *ACM Trans. Comput. Educ.*, *22*(3). Retrieved from https://doi.org/ 10.1145/3511886 doi: 10.1145/3511886

Vie, J.-J., Popineau, F., Bruillard, , & Bourda, Y. (2017, 02). A review of recent advances in adaptive assessment. In (Vol. 94, p. 113-142). doi: 10.1007/978-3-319-52977-6_4.

Xinogalos, S., Tomáš Pitner, Savić, M., & Ivanovic, M. (2020). First programming language in introductory programming courses, role of.. doi: 10.1007/978-3-319-60013 -0_217-1.

Yang, A. C., Flanagan, B., & Ogata, H. (2022). Adaptive formative assessment system based on computerized adaptive testing and the learning memory cycle for personalized learning. *Computers and Education: Artificial Intelligence*, *3*, 100104. doi: 10.1016/j.caeai.2022.100104.

Zhang, X., Li, Y., & Chen, J. (2024). Enhancing rural students' computer science self-efficacy in a robotics-based language arts course. *Education and Information Technologies, 29*(4), 5123–5140.

Zhao, L., Yang, J., & Wang, Y. (2023). Difficulty-aware knowledge tracing with code embeddings for novice programming learners. Journal of Educational Data Science, 5(1), 22–41.