


# An Experience with Adaptive Formative Assessment for Motivating Novices in Introductory Programming Learning

Jagadeeswaran Thangaraj ✉ 

School of Computing, Dublin City University, Ireland

Monica Ward ✉ 

School of Computing, Dublin City University, Ireland

Fiona O’Riordan ✉ 

CCT College, Dublin, Ireland

---

## Abstract

This study presents empirical research that uses adaptive formative assessment framework in addition to traditional lectures to motivate novice students in an introductory programming course. The primary goal of this work is to provide guidance for the creation of adaptive formative assessments in Python programming language to inspire novice students. The experiment is based on lessons learned from the literature and pedagogical theories that support learning through assessment and scaffolding. This study investigates how the experiment helped the novices, whether it increased their confidence, whether it assisted in identifying and correcting common errors, and whether it covered the material on learning modular programming components. It reports on extensive survey results of over 265 attempts of 90 students taking CS1 (introductory programming) that included five quizzes covering fundamental concepts. The students responded favorably to the experiment, and results are also included.

**2012 ACM Subject Classification** Applied computing → Education; Social and professional topics → Computing education; Social and professional topics → Student assessment

**Keywords and phrases** Assessment and feedback, Computer programming, CS1, Formative assessment, Introductory programming, Novice students

**Digital Object Identifier** 10.4230/OASICS.ICPEC.2024.6

**Funding** I want to thank the research committee of School of Computing at Dublin City University (DCU) for funding this research project.

## 1 Introduction

Any course in a higher education institution worldwide that is concerned with software development requires programming modules. By introducing syntax and semantics, these modules aim to impart fundamental knowledge of programming languages. These modules are crucial for students to feel confident in their study in Computer Science (CS). Students who are learning their introductory programming must also become familiar with the often-hard syntax of the language, numerous data types and its operations, the effects of various statements on variables, and control flow. Novice programmers are those taking their first computer programming courses or those with no prior programming experience. E.g. First year computer science degree students, second level students such as junior or senior cycle years. Independent components of programming will increase the difficulties of novices in learning programming [23]. There are a number of activities introduced to motivate novice students in programming modules in addition to traditional lecture and practical sessions. The recommended pedagogical activities are pair programming, peer instructions, live coding, collaborative learning and assessment and feedback systems [7, 30].



© Jagadeeswaran Thangaraj, Monica Ward, and Fiona O’Riordan;  
licensed under Creative Commons License CC-BY 4.0

5th International Computer Programming Education Conference (ICPEC 2024).

Editors: André L. Santos and Maria Pinto-Albuquerque; Article No. 6; pp. 6:1–6:12

OpenAccess Series in Informatics



OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 6:2 An Experience with Adaptive Formative Assessment

Every lecture in the classroom is given by a lecturer and includes exposition lectures, live coding to solve practical problems in real time and practical exercises using programming languages in lab sessions. The traditional lecture session occupied the majority of the practice. Also, students felt frustrated that they had to repeatedly go over concepts in lab session from they had already learned in lecture [4]. Furthermore, it does not help students to grasp fundamental ideas. Conversely, they found the live coding and practical portion especially enjoyable since it allowed the students to witness the theories and concepts being used to build a program [25]. However, novice students are unable to interpret program code and have a lack of understanding of individual elements of programming due to their lack of programming experience [24]. The assessment system of programming helps in this scenario as it is an essential component of education that promotes learning [26]. Formative assessment is one of the approaches for effective programming learning that aims to increase student understanding and learning by providing feedback on students' submissions [23]. Taking this into consideration, this research created a formative assessment framework that aims at increasing students' confidence by familiarising novice students with independent components of programming. As an expansion of our earlier research [32], this paper explores the development of a formative assessment framework for this purpose, describes the experiment, and offers insights into its performance and future directions.

### **2** Exploring Formative Assessments' Potential to Improve Programming Learning

One strategy for efficient programming learning is formative assessment including feedback [31]. Formative feedback is information provided to a learner with the intention of altering their behavior or way of thinking to improve learning [27]. Formative feedback gives teachers the chance to personalize their responses, motivate student interest, gather data on each student's progress, encourage reflective thought, and encourage self-directed learning [33, 21]. Therefore, formative assessment refers to routine, interactive evaluations of student development and comprehension that are used in educational settings to identify learning needs and adapt training [15].

#### **2.1** Enhancing Programming skills through Learning from Errors

When writing programs, both novice and experienced programmers make errors – just not to the same extent. Different kinds of errors demotivate novices to engage in learning to program. Error detection and correction play a central role in programming learning: it is an essential cognitive and pedagogical component of learning to program [20]. Programming errors can appear in a variety of forms such as syntax and logical errors [6, 3, 2]. A program error that breaks language conventions and stops execution is known as a syntax error [3]. When a syntax error occurs in a compiled language, the compiler will usually produce an error message that points to the incorrect program lines. Most of the time, program compilers provide information on the kind and location of syntax errors. Compiler error messages can also be used to identify and fix syntax and semantic errors during program compilation or execution. However, for novice programmers, these error messages can occasionally be difficult to understand and make them confused [5]. Therefore, some research suggested improving error messages to make them easier to interpret and understand [10, 8]. They demonstrated that these enhancement messages were more useful than compiler messages for figuring out common syntax and semantic errors [5, 8].

In addition, students suffer more with logical errors than syntax errors [3]. There are three possible explanations for these logical mistakes: Algorithm, misunderstanding, and false information [12]. Another factor, called “Programmer misconception”, contributes to the misunderstanding of control flow in logic that led to the erroneous code [18]. As a result, the code can deliver inaccurate results. It is also quite challenging to identify and correct logical errors. Various studies have identified common logical errors that people make frequently and offered suggestions for how to avoid them [3, 12]. Thus, helping students understand their programming errors is another effective technique to teach them programming, as mistakes are an incredible means of instruction for programming courses [14, 36]. Therefore, our goal is to familiarise students with these types of errors ahead to make it easier for them to understand. It enables students to comprehend the common code errors they make as well as compiler error messages.

## 2.2 Enhancing Formative assessment with adaptive strategy

Assessments that are customized to each student individually based on their responses to previous test items are referred to as adaptive assessments [22]. Students who complete adaptive assessments have their ability level determined by the answers they provide; the most illuminating problems are then shown to the students to measure their abilities more accurately [13]. When taking an adaptive assessment, every student will experience it differently from another [34]. In a traditional assessment, every student works on the same set of tasks with varying degrees of difficulty. Students can work on the predetermined tasks in any sequence because they are predetermined. On the other hand, in an adaptive assessment, every student works on a customised set of tasks since the questions are chosen by an algorithm that considers the answers that each student has previously provided. Therefore, it varies from traditional assessment in that each participant is asked a separate set of questions rather than all the same ones [34]. As a result, students work on different questions that take different amounts of time to complete [13]. The system chooses the next test from a pool of available tests based on the the students’ performance [35].

## 2.3 Encouraging Learning and Proficiency in programming through Adaptive Formative assessment

The process of learning programming involves starting from beginning and using a completion method, such as changing or finishing program code [4]. When it comes to cognitive skills, programming is more advanced than rote memorisation; to complete programming activities, students must grasp particular concepts and understand how to apply them together [35]. Self-efficacy of students can be used to adaptively generate assessments that are customised to each student in terms of question difficulty, assessment length, and question types (e.g., multiple choice, fill-in-the-blank, or short response) [35]. An answer key and a number of choices make up the two components of a multiple-choice question (MCQ) [1]. A query or a remark is typically made by the stem. To proceed with the stem, the learner must choose the best or accurate option. MCQs are insufficient for evaluating a student’s coding proficiency in programming modules since they do not encourage learners to write their own code, even when they are at an advanced level. The ability to retain programming concepts and increase engagement, however, can be useful. The assessment length and question difficulty can also be modified. Using adaptive assessment, which organizes a collection of questions into three cognitive levels according to complexity (easy, moderate, and difficult), adaptive formative assessment evaluates students’ knowledge in programming courses [9]. Consequently, rather than supplying resources that are universally applicable, adaptive assessment systems could customize instruction and evaluation by considering each student’s uniqueness.

## 6:4 An Experience with Adaptive Formative Assessment

■ **Table 1** An example question with answer choices and accompanying feedback.

Question	Choices	Feedback
What will be the output of following code?  <code>var = "computer" print(var[5::1])</code>	computer	<b>Incorrect!</b> It prints a range of items starting from index 5 with step 1. [ter]
	ter	<b>Correct!</b> <code>print( var[5 :: 1] )</code> - it prints a range of items starting from index 5 with step 1. [ter]
	compu	<b>Incorrect!</b> if the index is "0" <code>print(var[0::1])</code> , then it prints compu.
	u	<b>Incorrect!</b> if the index is "5" <code>print(var[5])</code> , then it prints u.

### 2.4 Research Questions

This study intends to investigate the scaffolding and support that formative assessment quizzes might provide for students learning to program. Using the adaptive formative assessment quizzes, this study will particularly investigate the following research questions.

**RQ-1:** Does formative assessment help to build self-confidence in novice programmers in learning basic concepts of programming?

**RQ-2:** Does formative assessment help to understand and correct the errors in order to improve their programming skills?

**RQ-3:** Does formative assessment help novices effectively learn the independent components of programming concepts?

## 3 Development of Adaptive Formative Assessment Framework

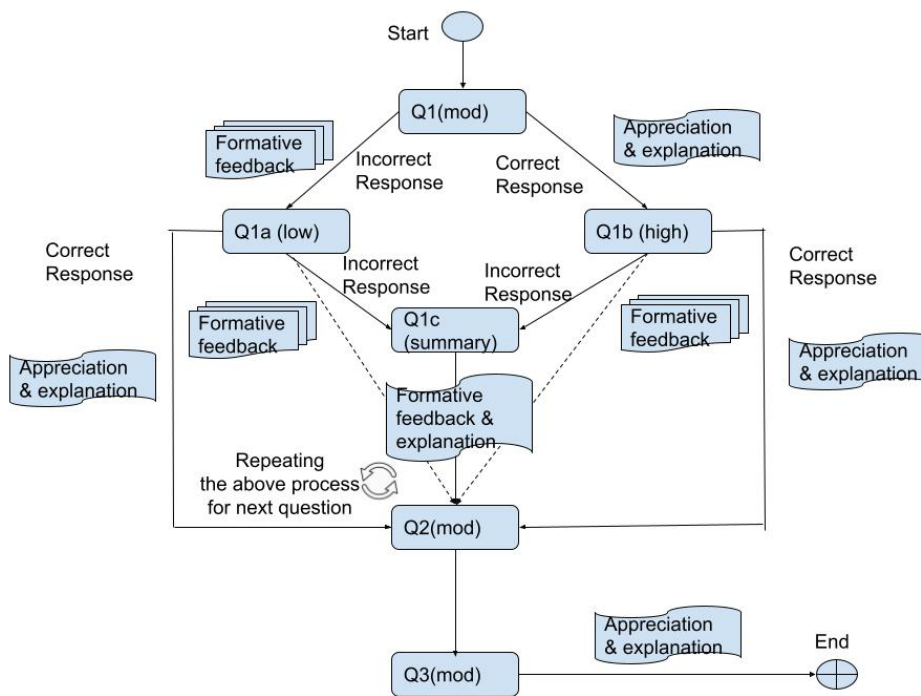
This study has led us to create formative assessment quizzes to introduce common programming errors. These quizzes are an excellent method to increase student confidence and introduce more frequent errors while programming. We have a list of questions with various answers in these quizzes. As they offer feedback for each selection, these quizzes assist in fostering their learning. While the wrong answer feedback helps them locate the appropriate response, the right answer feedback acknowledges their responses. Students can learn from their incorrect responses and determine the correct response. As a result, it is a system that progresses and aids in their ability to learn from mistakes. This study examines if the formative assessment increases participants' confidence in their capacity to understand the fundamental ideas behind programming. They assist students in comprehending the common code errors they make as well as compiler error messages.

### 3.1 Quiz Implementation

Google Forms is utilized to implement the quizzes because, according to [11], it can be an effective tool for formative assessment and for promoting active learning. Furthermore, integration with all learning management systems is possible. Each quiz aims to educate students about common code errors they made when studying the assigned topics. A sample question is shown in Table 1. Feedback will be given to students for each potential response, which will help them better comprehend the errors and help them to understand easily as shown in Table 1. Feedback are customised messages that are similar to enhanced error messages [5]. Every incorrect reaction offers advice on how to respond. Students can select the best alternative based on the feedback.

### 3.2 Adaptive Approach

Difficulty levels have been added to learning objects in the model. These learning objects could be topics, questions, a variety of errors. The goals relate to questions with varying degrees of difficulty. Difficulties in programming are classified as Bloom’s taxonomy of programming [29]. In this model, we classified a list of questions in three cognitive levels based on the complexity (like easy, moderate, and difficult) [17]. Here easy questions assess the basic concepts, moderate questions assess comprehensive knowledge and difficult questions do the applications of the knowledge [34]. If a student successfully responds to a moderate question on this assessment, 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 [28]. A sample classification is described as Table 2.



■ **Figure 1** Adaptive approach.

■ **Table 2** Summary of print statement question in adaptive model.

Purpose	Difficulty low	Difficulty moderate	Difficulty high	Summary
String notation and character traverse	var = 'Amazon' print(var[4])	var = 'Computer' print(var[5 :: 1])	var = 'Python' print(var[4 :: -1])	var = 'James Bond' print(var[3]) print(var[5 :: 1]) print(var[5 :: -1])

### 3.3 Questions Development

Python is an established programming language that is utilized in introductory programming courses due to its convenience and syntactical simplicity [16]. Variables, operators, conditionals, loops, and functions were all covered in the introductory programming course. We created quizzes corresponding with these topics that familiarise modular parts of programming. Each question intends to introduce some of the most common errors made by novices [3, 36]. Details of the questions are shown in Table 3. Quiz topics included syntax errors, logical errors, and common misconceptions among novice students [2, 12, 18]. Every quiz has at least five questions of a moderate difficulty. Depending on the responses, the question moves automatically from a moderate level to a high level or low level as shown in Figure 1. It goes to a summary question that describes the relevant concept to provide proficiency level if they are unable to respond to all of the questions. With the help of the lecturer, we conducted these quizzes periodically during teaching sessions to build novice's confidence as well as to capture their barriers in programming.

■ **Table 3** Summary of questions of each quiz.

Quiz no	Topic	Modular parts	Objectives
1	Print & operators	Print statement Arithmetic operators Assignment operator	Familiarising syntax errors in print statement Familiarising syntax errors in arithmetic expressions Familiarising syntax errors in assignment
2	Variables, input & operations	Input function Operators & precedence Higher arithmetic operator	Familiarising variables and type conversion when read a value from keyboard Familiarising operators & precedence in expressions Familiarising higher precedence arithmetic operators (% , // , **)
3	If/Else statements	The if/elif/else structure with comparative operators Operators & precedence Errors in If/else	Familiarising if/else process & comparative operators Familiarising comparative expression using AND, OR, NOT Familiarising syntax error & indentation error in assignment
4	While loop	While loop process Introducing multiple while loops & infinitive executions Mixed loops	Introducing while loop elements, process before or after increment or decrement Familiarising multiple loops and syntax errors & Non-terminating loops Familiarising multiple loops (while & if/else) & Syntax or logical errors
5	Strings & Functions	String representation Functions Global & Local variables in function	Familiarising different string array representations Familiarising values passing to parameters & syntax errors Familiarising logical errors in assignment of global or local variables

## 4 Research Methodological paradigm

This research combines both qualitative and quantitative elements. This instance is constrained by the CS1 module during the academic year 2023-24, and different student cohorts of first year undergraduate Computer Science students. In quantitative research, the interven-

tion technique is used to address reliability and validity. A brief, voluntary, anonymous survey is used in conjunction with formative assessment to gain insight of students perceptions on how they perceived and experienced it. It also includes qualitative elements and makes use of a range of data sources and data collection methods. It was conducted via an anonymous “Google forms” questionnaire.

#### 4.1 The Population

The University’s Introductory programming modules provided the data for this investigation. The data includes 267 students’ programming quiz attempts that they submitted at the end of each quiz session. Some students attempted many surveys, in relation to the total number of quiz attempts. These participants were from non-CS major course.

#### 4.2 Data collection strategies

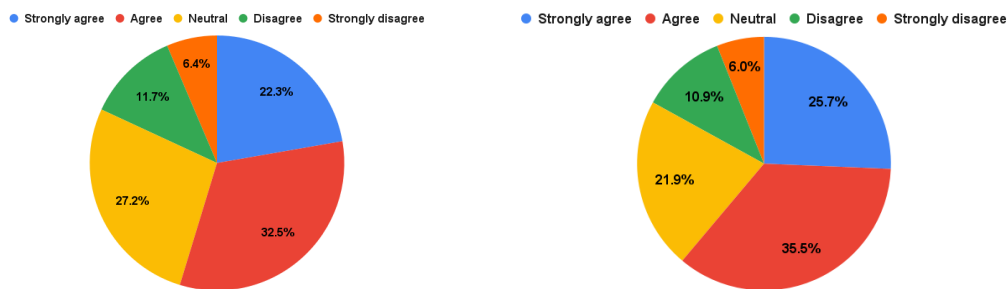
Data was gathered using a survey consisting of both closed-ended and open-ended questions to obtain both qualitative and quantitative data [19]. In addition to traditional teaching and practical sessions, regular quizzes were provided during the study periods. This gave the chance to take quizzes and consider what they had learned. For this study, a short optional and anonymous survey was employed to get an idea of how learners viewed and experienced the quizzes for introductory programming at the end of each quiz. At the end of each quiz, we conducted a survey about how it effectively helped them to learn programming. The respondents were questioned about how they felt about formative assessment quizzes of each programming topic. Open-ended questions for qualitative data and closed-ended Likert scale questions for quantitative data were both used in the survey form. The Likert scale had five possible scores: strongly disagree, disagree, neutral, agree, and strongly agree. The surveys provide both quantitative and qualitative information about the effects of this intervention and perceptions about the use of formative assessment in learning programming. The student questionnaires and their reflective writing assignments provide the qualitative data. Every piece of qualitative data is anonymous.

### 5 Results & Discussion

In order to collect more thorough data for analysis, this study gave students quizzes at regular intervals. Each quiz contained survey questions in addition to other quiz questions. Following an intervention survey, all analysis was completed.

■ **Table 4** Post Survey Likert Questions 1 and 2 (N=265).

Question	Strongly Dis-agree:1	Disagree:2	Neutral:3	Agree:4	Strongly Agree:5	Mean	SD
Do these quizzes increase your self-confidence in learning programming?	17	31	72	86	59	3.52	1.14
Do these quizzes help to understand and correct errors in Python?	16	29	58	94	68	3.64	1.15



■ **Figure 2** Overall feedback on self-confidence. ■ **Figure 3** Overall feedback on understanding & correcting errors.

### 5.1 RQ-1: Increasing self-confidence

To answer the RQ-1, it included a Likert question, “Do these quizzes increase your self-confidence in learning programming?”, at the end of the quizzes. The responses ranged from ‘Strongly disagree’ to ‘Strongly agree’. The whole survey data results are presented in Table 4. It offers a thorough understanding of the students’ feelings regarding their level of self-confidence in handling these quizzes. Responses for ‘Strongly agree’ and ‘Agree’ ( $\approx 55\%$ ) were higher than ‘Disagree’ part ( $\approx 18\%$ ) as shown in Figure 2. As a result, this study discovered that the adaptive formative assessment quizzes helped them increase their self-confidence.

### 5.2 RQ-2: Understand & Correct the errors

This study asked, “Do these quizzes help to understand and correct errors in Python?”. The responses ranged from ‘Strongly disagree’ to ‘Strongly agree’. The whole survey data results are presented in Table 4. Huge responses ( $\approx 61\%$ ) were received as ‘Agree’ as a result as in Figure 3. Their responses for these two questions show a clear shift from their post-quizzes average of 3.52 to an average of 3.64 (with less volatility) as shown in Table 4. These results indicate that the adaptive formative assessment quizzes helped them comprehend the common programming errors.

### 5.3 RQ-3: Students’ perception on learning modular parts

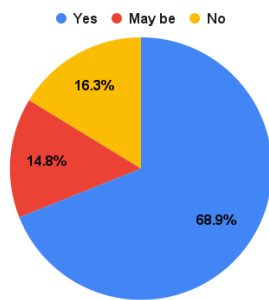
This study asked, “Do these quizzes help to better understand basic concepts of Python language?”. The responses were ‘Yes’, ‘May be’ and ‘No’. The result is presented in Table 5. Huge responses ( $\approx 68\%$ ) were received as ‘Yes’ as a result as shown in Figure 4. Based on the surveys, it demonstrates that the novice students believe the quizzes assisted in grasping fundamental programming principles and significantly increased students’ confidence in learning programming after attending. This study also asked, “Is the feedback you receive for each question helpful in finding the correct answers and understanding errors?”. The responses were ‘Yes’, ‘May be’ and ‘No’. These responses were also highly positive as shown in Table 5.

Quantitative data alone does not provide the full picture of the learning experience. Finding out what students think and feel about formative assessment as a computer programming learning activity is critical. According to sentimental analysis, they delighted in gaining knowledge by taking quizzes in various models and they also valued these quizzes for various reasons as stated in the comments below.

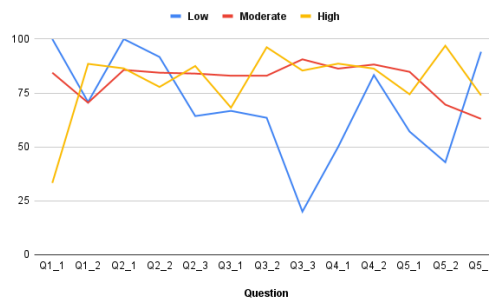


■ **Table 5** Post Survey Likert Question 3 (N=267).

Question	No: 1	May be: 2	Yes: 3	Mean	SD
Do these quizzes help to better understand basic concepts of Python language?	40	45	182	2.53	0.74
Is the feedback you receive for each question helpful in finding the correct answers and understanding errors?	43	39	185	2.52	0.76



■ **Figure 4** Overall feedback on understanding basic concepts.



■ **Figure 5** The correct answer frequency (in %) of difficult levels during the quiz attempts.

*...It introduced me to new elements of python...made me realise what i didn't know...was good to refresh my brain...very helpful exercises...I think they are much better than the way the lectures are being taught...It helped to recall... Maybe do the quizzes in the lectures to fully understand what is being taught...*

Responses for understanding modular concepts (M = 2.53, SD = 0.74) out of four questions were given a higher weight than other responses with a small effect size as shown in Table 5. It is evident from their feedback that these quizzes enabled them to review their programming expertise and make necessary revisions. Some students claimed that taking the quizzes had taught them some new material.

### 5.4 Low vs Moderate vs High – Difficulty levels

The frequency of correct responses to questions at varying degrees of difficulty is shown by the outcome chart that we have plotted in Figure 5. It demonstrates that the majority (≈81%) of students correctly answer questions at a moderate difficulty level. On the other hand, students who attempt questions with low difficulty levels tend to give less (≈70%) correct answers. Additionally, students who tackle highly difficult questions give more (≈80%) correct answers. Eventually, students get more detailed feedback to their summary questions, which helps them understand the modular parts of programming better. Moreover, it demonstrates that adaptive formative assessments helped them understand the basic concepts better.

## 6 Conclusion and Future Work

The experience presented in this paper describes the use of adaptive formative assessment in introductory programming to help novice students overcome the challenges of learning to program, particularly in identifying and understanding more frequent errors. The results

are clearly confirming that this approach helped novice students become more confident in programming and dispel some of the misconceptions surrounding it. As a conclusion, we argue that adaptive formative assessment quizzes motivate students to evaluate and learn from their mistakes, which in turn encourages them to learn computer programming. It can be a viable teaching and learning tool for computer programming. The reflection makes it easier to comprehend the basic concepts. The students claim that learning through formative assessment quizzes has improved their comprehension and increased their confidence in learning programming. They also claim to be satisfied and indicate that they would repeat this teaching technique again, as evidenced by their high level of loyalty. This study will further investigate whether the adaptive formative assessment quizzes could help novice students learn more effectively in other programming languages. Consequently, we intend to incorporate this technique into other languages like Irish, Portuguese and Spanish.

---

### References

- 1 Pedro Henriques Abreu, Daniel Castro Silva, and Anabela Gomes. Multiple-choice questions in programming courses: Can we use them and are students motivated by them? *ACM Transactions on Computing Education (TOCE)*, 19(1):1–16, 2018.
- 2 Alireza Ahadi, Raymond Lister, Shahil Lal, and Arto Hellas. Learning programming, syntax errors and institution-specific factors. In *Proceedings of the 20th Australasian Computing Education Conference, ACE '18*, pages 90–96, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3160489.3160490.
- 3 Nabeel Alzahrani and Frank Vahid. Common logic errors for programming learners: A three-decade literature survey. In *2021 ASEE Virtual Annual Conference Content Access*, Virtual Conference, July 2021. ASEE Conferences. URL: <https://peer.asee.org/36814>.
- 4 Zahra Atiq and Michael Loui. A qualitative study of emotions experienced by first-year engineering students during programming tasks. *ACM Transactions on Computing Education*, 22, March 2022. doi:10.1145/3507696.
- 5 Brett Becker, Graham Glanville, Ricardo Iwashima, Claire McDonnell, Kyle Goslin, and Catherine Mooney. Effective compiler error message enhancement for novice programming students. *Computer Science Education*, pages 1–28, September 2016. doi:10.1080/08993408.2016.1225464.
- 6 Anat Ben-Yaacov and Arnon HersHKovitz. Types of errors in block programming: Driven by learner, learning environment. *Journal of Educational Computing Research*, 61(1):178–207, 2023. doi:10.1177/07356331221102312.
- 7 Neil C. C. Brown and Greg Wilson. Ten quick tips for teaching programming. *PLOS Computational Biology*, 14(4):1–8, April 2018. doi:10.1371/journal.pcbi.1006023.
- 8 Tessa Charles and Carl Gwilliam. The effect of automated error message feedback on undergraduate physics students learning python: Reducing anxiety and building confidence. *Journal for STEM Education Research*, 6(2):326–357, August 2023. doi:10.1007/s41979-022-00084-4.
- 9 Dimitra Chatzopoulou and Anastasios Economides. Adaptive assessment of student’s knowledge in programming courses. *J. Comp. Assisted Learning*, 26:258–269, August 2010. doi:10.1111/j.1365-2729.2010.00363.x.
- 10 Paul Denny, James Prather, Brett A. Becker, Catherine Mooney, John Homer, Zachary C Albrecht, and Garrett B. Powell. On designing programming error messages for novices: Readability and its constituent factors. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, CHI '21*, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3411764.3445696.
- 11 Mireille Djenno, Glenda M. Insua, and Annie Pho. From paper to pixels: using google forms for collaboration and assessment. *Library Hi Tech News*, 32(4):9–13, 2022. doi:10.1108/LHTN-12-2014-0105.

- 12 Andrew Ettles, Andrew Luxton-Reilly, and Paul Denny. Common logic errors made by novice programmers. In *Proceedings of the 20th Australasian Computing Education Conference, ACE '18*, pages 83–89, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3160489.3160493.
- 13 Takayuki Goto, Kei Kano, and Takayuki Shiose. Students’ acceptance on computer-adaptive testing for achievement assessment in japanese elementary and secondary school. *Frontiers in Education*, 8, 2023. doi:10.3389/feduc.2023.1107341.
- 14 Heather J. Hoffman and Angelo F. Elmi. Do students learn more from erroneous code? exploring student performance and satisfaction in an error-free versus an error-full sas® programming environment. *Journal of Statistics and Data Science Education*, 29(3):228–240, 2021. doi:10.1080/26939169.2021.1967229.
- 15 Seyed M. Ismail, D. R. Rahul, Indrajit Patra, and Ehsan Rezvani. Formative vs. summative assessment: impacts on academic motivation, attitude toward learning, test anxiety, and self-regulation skill. *Language Testing in Asia*, 12(1):40, 2022. doi:10.1186/s40468-022-00191-4.
- 16 Fionnuala Johnson, Stephen McQuistin, and John O’Donnell. Analysis of student misconceptions using python as an introductory programming language. In *Proceedings of the 4th Conference on Computing Education Practice, CEP '20*, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3372356.3372360.
- 17 Fatima Ezzahraa Louhab, Ayoub Bahnasse, and Mohamed Talea. Towards an adaptive formative assessment in context-aware mobile learning. *Procedia Computer Science*, 135:441–448, 2018. The 3rd International Conference on Computer Science and Computational Intelligence (ICCS CI 2018) : Empowering Smart Technology in Digital Era for a Better Life. doi:10.1016/j.procs.2018.08.195.
- 18 Davin Mccall and Michael Kölling. Meaningful categorisation of novice programmer errors. In *Proceedings - Frontiers in Education Conference, FIE*, volume 2015, October 2014. doi:10.1109/FIE.2014.7044420.
- 19 Donna Mertens. *Research and Evaluation in Education and Psychology: Integrating Diversity with Quantitative, Qualitative, and Mixed Methods 5th edition*. SAGE Publications, Inc, June 2019.
- 20 Anastasia Misirli and Vassilis Komis. Computational thinking in early childhood education: The impact of programming a tangible robot on developing debugging knowledge. *Early Childhood Research Quarterly*, 65:139–158, 2023. doi:10.1016/j.ecresq.2023.05.014.
- 21 Gunilla Näsström, Catarina Andersson, Carina Granberg, Torulf Palm, and Björn Palmberg. Changes in student motivation and teacher decision making when implementing a formative assessment practice. *Frontiers in Education*, 6, 2021. doi:10.3389/feduc.2021.616216.
- 22 Elena Papanastasiou. *Adaptive Assessment*, pages 1–2. Springer Netherlands, Dordrecht, 2021. doi:10.1007/978-94-007-6165-0\_3-4.
- 23 Yizhou Qian and James Lehman. Students’ misconceptions and other difficulties in introductory programming: A literature review. *ACM Trans. Comput. Educ.*, 18(1), October 2017. doi:10.1145/3077618.
- 24 Sjaak Smetsers Renske Weeda and Erik Barendsen. Unraveling novices’ code composition difficulties. *Computer Science Education*, 0(0):1–28, 2023. doi:10.1080/08993408.2023.2169067.
- 25 Anders Schlichtkrull. An experience with and reflections on live coding with active learning. In *International Computer Programming Education Conference*, 2023. URL: <https://api.semanticscholar.org/CorpusID:260777639>.
- 26 Nicole Shanley, Florence Martin, Nicole Collins, Manuel Perez-Quinones, Lynn Ahlgrim-Delzell, David Pugalee, and Ellen Hart. Teaching programming online: Design, facilitation and assessment strategies and recommendations for high school teachers. *TechTrends*, 66, April 2022. doi:10.1007/s11528-022-00724-x.
- 27 Valerie J. Shute. Focus on formative feedback. *Review of Educational Research*, 78(1):153–189, 2008. doi:10.3102/0034654307313795.

## 6:12 An Experience with Adaptive Formative Assessment

- 28 E'loria Simon-Campbell and Julia Phelan. Effectiveness of an adaptive quizzing system as a self-regulated study tool to improve nursing students' learning. *International Journal of Nursing & Clinical Practices*, 5, August 2018. doi:10.15344/2394-4978/2018/290.
- 29 Sonia Sobral. Bloom's taxonomy to improve teaching-learning in introduction to programming. *International Journal of Information and Education Technology*, 11:148–153, March 2021. doi:10.18178/ijiet.2021.11.3.1504.
- 30 Sonia Sobral. *Strategies on Teaching Introducing to Programming in Higher Education*, pages 133–150. Springer, Cham, March 2021. doi:10.1007/978-3-030-72660-7\_14.
- 31 Qing Sun, Ji Wu, Wenge Rong, and Wenbo Liu. Formative assessment of programming language learning based on peer code review: Implementation and experience report. *Tsinghua Science and Technology*, 24:423–434, August 2019. doi:10.26599/TST.2018.9010109.
- 32 Jagadeeswaran Thangaraj, Monica Ward, and Fiona O'Riordan. The impact of using formative assessment in introductory programming on teaching and learning. *10th International Conference on Higher Education Advances (HEAd'24)*, Valencia, June 2024.
- 33 Fabienne S. Van der Kleij and Lenore Adie. Formative assessment and feedback using information technology. *Second Handbook of Information Technology in Primary and Secondary*, pages 601–615, 2018. doi:10.1007/978-3-319-71054-9.
- 34 Jill-Jênn Vie, Fabrice Popineau, Éric Bruillard, and Yolaine Bourda. *A Review of Recent Advances in Adaptive Assessment*, volume 94, pages 113–142. Springer International Publishing, February 2017. doi:10.1007/978-3-319-52977-6\_4.
- 35 Albert C.M. Yang, Brendan Flanagan, and Hiroaki Ogata. Adaptive formative assessment system based on computerized adaptive testing and the learning memory cycle for personalized learning. *Computers and Education: Artificial Intelligence*, 3:100104, 2022. doi:10.1016/j.caeai.2022.100104.
- 36 Zihe Zhou, Shijuan Wang, and Yizhou Qian. Learning from errors: Exploring the effectiveness of enhanced error messages in learning to program. *Frontiers in Psychology*, 12, 2021. doi:10.3389/fpsyg.2021.768962.