Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025
Paper ID #47321

Positive Student Impacts of an Unlimited, Randomized Self-Assessment Quiz Per Chapter: Study Habits, Self-Efficacy, and Learning Outcomes

Dr. Annie Hui, zyBooks, A Wiley Brand

Annie Hui is a zyBooks assessment specialist. She has 15 years of experience teaching computer science, information technology, and data science courses, in both in-person and online modes. She has taught in Northern Virginia Community College and George Mason University. She specializes on course design to maximize student engagement and success.

Dr. Nkenge Wheatland, zyBooks, A Wiley Brand

Nkenge Wheatland is a Sr. Manager for Content Development in Computer Science at zyBooks, where she overseas the development and maintenance of CS titles. She earned a B.S in Computer Science at the University of Maryland, Baltimore County (UMBC) and an M.S. and Ph.D. in Computer Science at the University of California, Riverside (UCR).

Chelsea L Gordon, zyBooks, A Wiley Brand

Chelsea Gordon received her PhD in Cognitive Science at University of California, Merced in 2019. Chelsea works as a research scientist for zyBooks, a Wiley company that creates and publishes interactive, web-native textbooks in STEM.

Ms. Efthymia Kazakou, zyBooks, A Wiley Brand

Efthymia Kazakou is Sr. Assessments manager at zyBooks, a startup spun-off from UC Riverside and acquired by Wiley. zyBooks develops interactive, web-native learning materials for STEM courses. Efthymia oversees the development and maintenance of all zyBo

Neil Thawani, zyBooks, A Wiley Brand
Michael Goldwasser, Saint Louis University

Michael Goldwasser is a Professor of Computer Science at Saint Louis University and served six years as the Director of the Computer Science program followed by six years as the inaugural chair of a new Department of Computer Science. Dr. Goldwasser's research interests involve the design and analysis of data structures and algorithms and computer science education. He is the author of three traditional undergraduate textbooks with Pearson and Wiley, and four interactive textbooks with zyBooks, where he also serves as a part-time Content Author.

Dr. Yamuna Rajasekhar, zyBooks, A Wiley Brand

Yamuna Rajasekhar is Director of Content, Authoring, and Research at zyBooks, a Wiley Brand. She leads content development for the Computer Science and IT disciplines at zyBooks. She leads the authoring and pedagogy team at zyBooks, developing innovative learning solutions that drive measurable student success. She is also an author and contributor to various zyBooks titles. She was formerly an assistant professor of Electrical and Computer Engineering at Miami University. She received her M.S. and Ph.D. in Electrical and Computer Engineering from UNC Charlotte.

Positive Student Impacts of an Unlimited, Randomized Self-Assessment Quiz Per Chapter: Study Habits, Self-Efficacy, and Learning Outcomes

Many introductory computer science and engineering students excessively struggle with course content, leading to reduced self-efficacy, learning outcomes, and retention in the major. Since engineering concepts build upon earlier material, developing a strong foundation with earlier material is key to reducing excessive struggle with later material. Research shows that enabling a student to assess their progress in the learning material is a way to improve study habits, as well as improve self-efficacy. In this study, an online exam system delivered a self-assessment quiz as the last section of each chapter used in the course's online textbook on the zyBooks platform. A student could take the self-assessment quiz anytime and as many times as desired; the student could see their overall score after submitting a quiz attempt. Students received a few course points for attempting each self-assessment quiz. This pilot study is based on an introductory computer science (CS) course offered during Fall 2024 with approximately 124 students total and 17 self-assessment quizzes. Each quiz contained question types common in CS exams: Multiple choice (5-8 per quiz) and parameterized code writing questions (2-3 per quiz) with immediate auto-grading. The code writing questions were randomized per exam attempt, enabling a different challenge for repeat attempts. The goal of the pilot was to measure the impact on students' study habits, self-efficacy, and learning outcomes. Students completed a 25-item survey regarding knowledge of course content and self-efficacy, at the start and end of the course. At the end of each chapter, students were offered the self-assessment quiz, followed by a brief survey on the insights the student gained about their understanding of the material, and impact on study habits and self-efficacy. This paper presents exploratory analyses examining students' self-assessment quiz usage patterns through the course, quantifying students' engagement with the self-assessment quizzes, and gathering insights into whether students found the self-assessment quizzes helpful, enjoyable, and worthy of inclusion in the course.

Introduction

To many students, learning to program can be initially overwhelming as the students are required to learn and master concepts as well as programming syntax. Laying a strong foundation for programming skills is critical to prepare a student for success in an introductory programming course. Having plenty of opportunities to practice coding is essential to enable students to learn programming concepts effectively. A lack of effective study strategies combined with an overload of concepts and not enough time lead to reduced retention and higher drop rates in introductory programming courses [1].

Several research teams have considered the impacts of providing students in introductory programming courses the opportunity for practice and self-assessment via short exercises. Brusilovsky and Sosnovsky [2] describe the development of their QuizPACK system for generating randomized variants of coding exercises, and the availability of that system for students' practice and self-assessment; in a follow-up study [3] they describe two approaches for increasing students' use of the platform. Chung and Hsiao [4] describe an intervention in which students were presented one multiple choice question for review each day, and how students' short-term and long-term patterns of persistent engagement with the platform related to later exam scores. Rodríguez-Vidal et al. [5], [6] analyze 12 years of collected data regarding students' use of self-assessment coding exercises and their influence on final grades; an earlier study of the same system AulaWeb system was described by García-Beltrán and Martinez [7].

Researchers have also examined the role of self-regulated learning and engagement on computer science student performance. Bergin et al. [8] describe a study of 35 undergraduate students in an introductory programming course, using the Motivated Strategies for Learning Questionnaire to collect data on students' learning values and cognitive and metacognitive strategies, and how those responses relate to students' academic performances. Faulkner et al. [9] identify specific self-regulated learning strategies that are manifested in the context of first-year programming students. Ilves et al. [10] examine the use of graphical visualization to support students' self-regulated learning in online computer science courses.

In this study, interactive self-assessments are applied to an introductory computer science course. The self-assessments designed for this study are intended to be a diagnostic tool offering students the opportunity to practice and check their comprehension of the material while studying. The self-assessments (1) help students recall the concepts they have learned, (2) reinforce students' understanding of important topics and concepts, (3) strengthen students' skills on essential techniques of problem solving, and (4) identify any topics that could benefit from additional reading and practice. Offering self-assessments in a low-stakes environment enables students to take ownership of their learning and attempt the assessments multiple times until they feel confident.

Design

Assessment environment

The zyBooks assessment system is designed to offer a comprehensive and flexible approach to evaluating student learning. It supports the creation of multiple-choice questions, allows for the import of existing questions from test banks, and includes highly randomized, scaffolded, auto-graded activities known as Challenge Activities (CAs). Additionally, the system facilitates the creation of programming labs to assess practical coding skills. Each assessment has a set duration, and students are permitted unlimited attempts to encourage mastery through practice.

The system also includes a detailed reporting feature, enabling instructors to track student performance at both the assessment and question level for each submission. Upon completing an assessment, students can immediately view their scores, promoting timely feedback and continuous improvement.

Design of self-assessment questions

Our self-assessments are carefully designed to focus on essential concepts and provide a straightforward evaluation of the topics being assessed. To ensure comprehensive coverage of essential knowledge, we first identify the learning objectives of each section. No two self-assessments are identical, as each is tailored to the unique content being assessed.

Each self-assessment question is a standalone item consisting of a few short sub-questions. These questions assess a range of skills, including concepts (e.g., defining terminology), application of concepts (e.g., describing a scenario using a concept), logical deduction (e.g., predicting the outcome of a program), and problem-solving (e.g., completing a program).

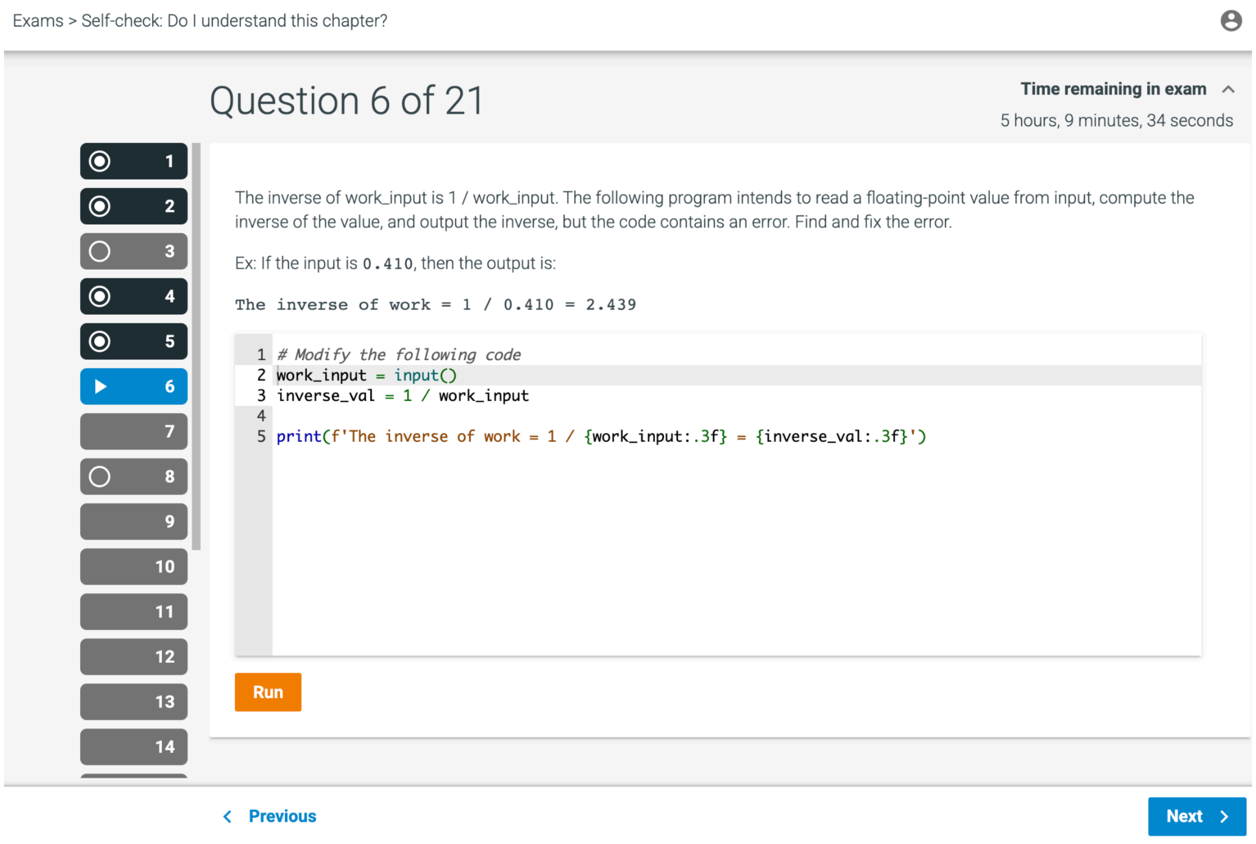
While self-assessment questions are derived directly from the section content, their format is sometimes adapted to fit the assessment's needs. For example, a matching activity used in a zyBooks section might be restructured as a multiple-choice question in the self-assessment quiz. This straightforward approach enables students to focus on the most important concepts in a section and encourages them to review the content for better recall and understanding.

Concepts and application of concepts are primarily assessed through multiple-choice questions. These questions include a correct answer alongside distractors that reflect common misconceptions, requiring students to apply critical thinking to identify the correct response. Logical deduction and problem solving skills are assessed through CAs. The randomized nature of CAs ensures that students’ application of skills is not tied to a specific scenario, meaning success demonstrates mastery of the underlying skill.

Furthermore, CAs have been found to reinforce students’ competence as a course progresses. As reported in a recent study we conducted [11], among students who consistently attempted CAs through the course, a low correlation was found between these students’ CA failure rates and the complexity of course materials. The same study also addressed how the highly randomized CAs adapts to students’ need for practice in their learning process. Namely, students may practice once and be done with a topic that is easy to them. They may also practice earnestly on a challenging topic before and after they pass the CA on the topic. Thus, including CAs in a self-assessment ensures a strong tie between students' learning efforts and their test outcomes.

The size and duration of self-assessment quizzes

In the programming books, one self-assessment quiz is provided for each chapter. Each self-assessment quiz consists of one question per section of the chapter. For example, the title "Programming in Python 3" contains 17 chapters and 88 sections, resulting in 17 self-assessment quizzes with a total of 88 questions. Similarly, the title "Programming in Java" consists of 16 chapters and 101 sections, yielding 16 self-assessment quizzes and a total of 101 questions. The self-assessment quizzes are customized to fit the order of the topics adopted in each instructor's course. Instructors may further customize the quizzes to meet their needs. To promote effective practice, students need to spend no more than 15 minutes per assessment question. Mastery is defined as being able to answer a question correctly within five minutes.


Exams > Self-check: Do I understand this chapter?
Question 6 of 21
Time remaining in exam
5 hours, 9 minutes, 34 seconds
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Run
Previous
Next

The inverse of work_input is 1 / work_input. The following program intends to read a floating-point value from input, compute the inverse of the value, and output the inverse, but the code contains an error. Find and fix the error.

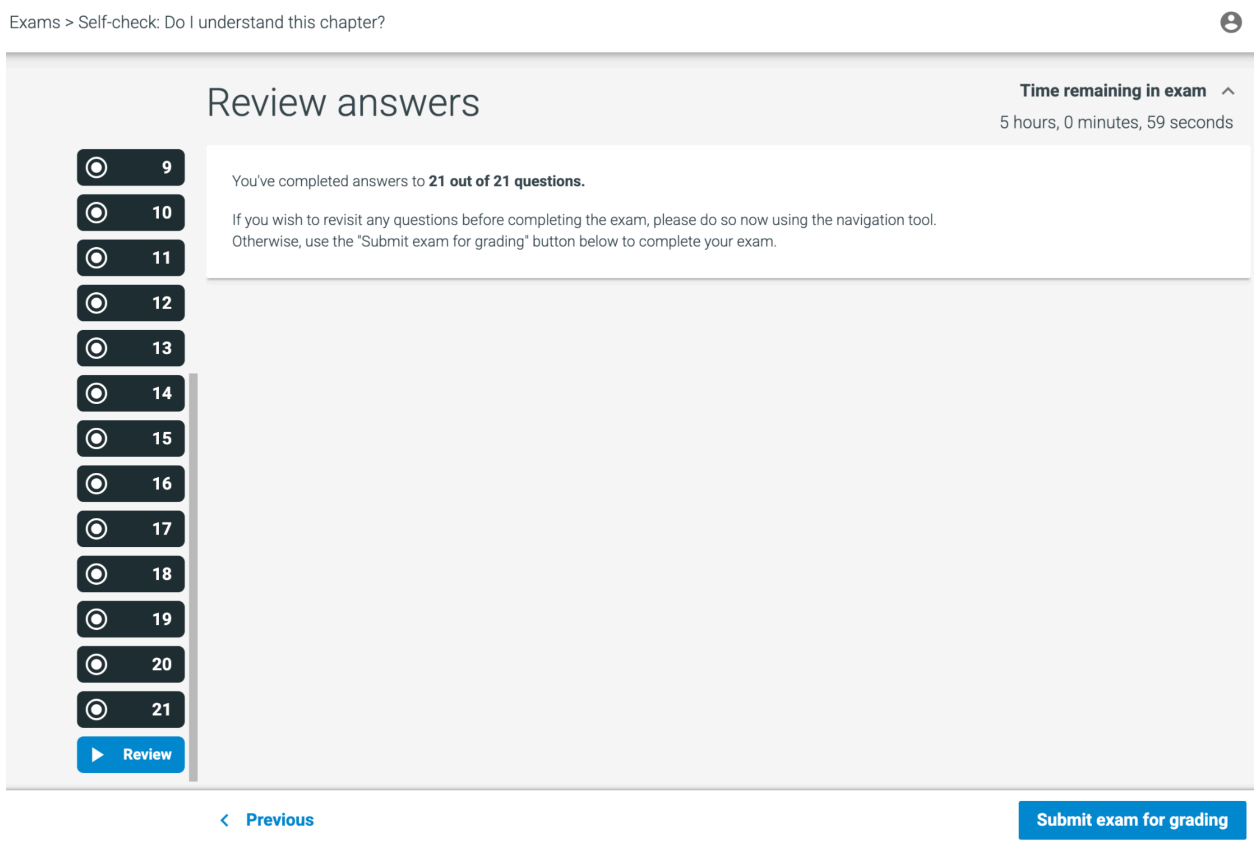
Ex: If the input is `0.410`, then the output is:

```
The inverse of work = 1 / 0.410 = 2.439
```

```python
# Modify the following code
work_input = input()
inverse_val = 1 / work_input

print(f'The inverse of work = 1 / {work_input:.3f} = {inverse_val:.3f}')
```

Figure 1: A self-assessment in progress. Students can skip questions and go back as needed.


Exams > Self-check: Do I understand this chapter?

Review answers

Time remaining in exam

5 hours, 0 minutes, 59 seconds

You've completed answers to **21 out of 21 questions.**

If you wish to revisit any questions before completing the exam, please do so now using the navigation tool. Otherwise, use the "Submit exam for grading" button below to complete your exam.

Figure 2: Final review page of a self-assessment.

Methods

The course in this case study is a semester-long, introductory course conducted in Python with a primary emphasis on programming. In the first three weeks, students are introduced to programming concepts and features such as I/O, syntax, basic data types, and arithmetic operations. Major programming constructs, namely, branches and loops, make up the next three weeks. Python-specific data types such as strings, lists, and dictionaries are covered next, followed by extensive coverage of functions. Advanced programming constructs and problem-solving techniques such as classes and recursions are covered towards the end of the course. A self-assessment quiz is available to the students at the end of every week. Students engage in intensive programming activities in the middle one-third of the course to solve problems that use programming constructs, data types, and functions. While students are exposed to a large variety of challenging problems, the self-assessments have no due dates, and are meant to be low stakes and student driven. The total number of students is 124, divided among three sections.

Results

Self-assessment usage data

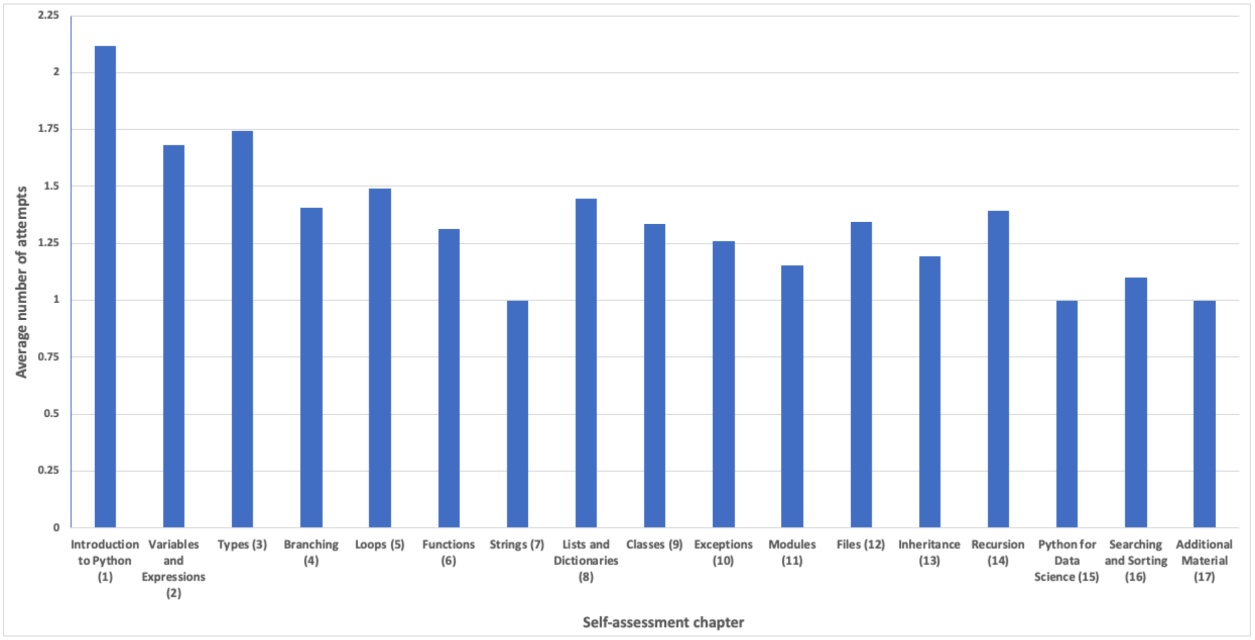


Figure 3: Average number of attempts per self-assessment.

Overall, we observed that each self-assessment was attempted at least once, and in some cases, the assessments were attempted multiple times per student. Figure 3 shows the average number of attempts for each self-assessment. This is promising as students seem to be using the self-assessments as part of their learning and practice. 69% of students attempted at least one out of the sixteen available self-assessments. We also observed that 60% students made multiple attempts overall.

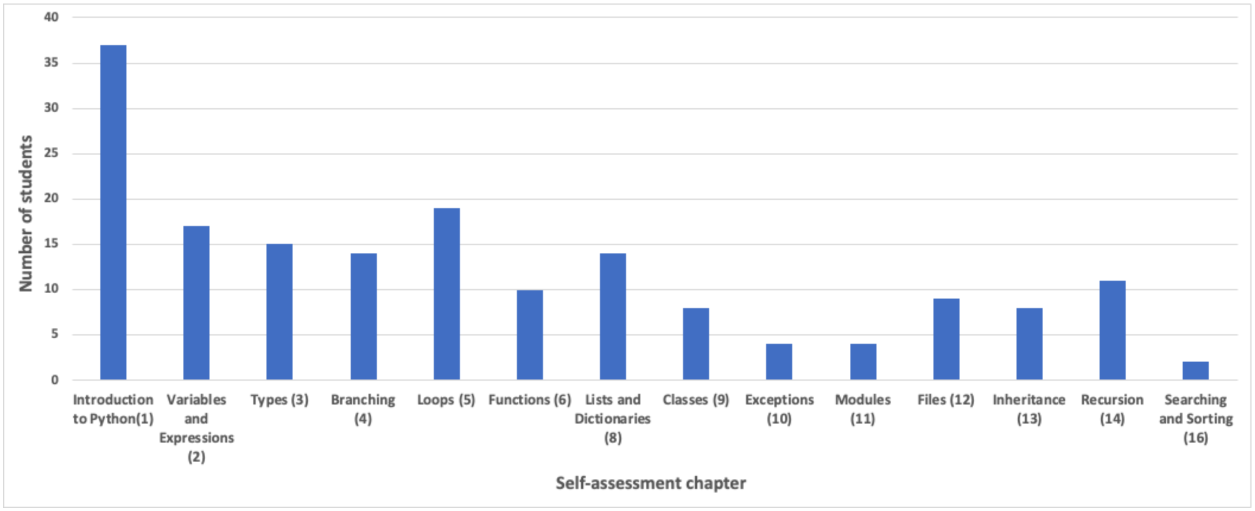


Figure 4: Number of student attempts per self-assessment.

Figure 4 shows how many students attempted each zyBook chapter's self-assessment more than once. As shown, the first chapter, Introduction to Python, had 37 total attempts by students. This was the highest number of attempts on a self-assessment. The chapter with the second-highest number of students (19) attempting the self-assessment more than once was Chapter 5, Loops, which is often a topic that students require additional practice on.

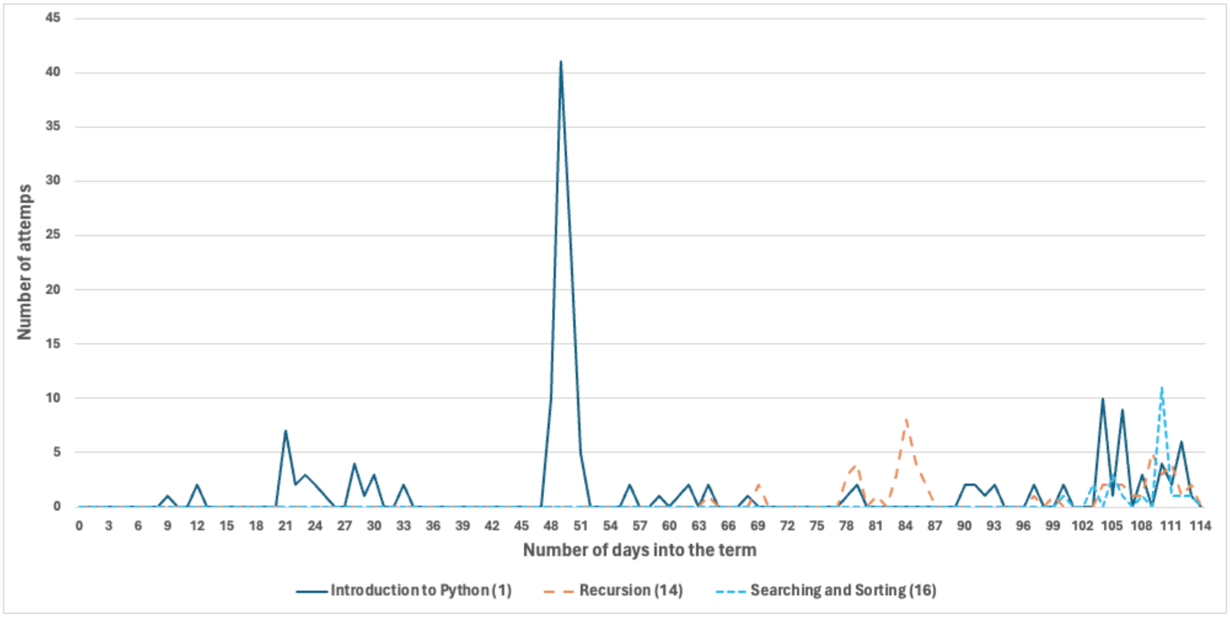


Figure 5: Self-assessments usage data over the course of the semester

Figure 5 shows how many self-assessments students attempted and when during the semester they completed those self-assessments for three chapters; Chapter 1: Introduction to Python, Chapter 14: Recursion, and Chapter 16: Searching and Sorting. As shown, midway through the semester, there was a significant surge in self-assessment attempts for Chapter 1: Introduction to Python. This surge in attempts between days 48 and 51 was right before the midterm, meaning students likely used the self-assessment for studying. Between days 82 and 87 of the semester, students began attempting the self-assessment in Chapter 14: Recursion. Recursion is typically one of the hardest topics in an introductory programming course and the surge in self-assessments indicates students needed more help to understand the concept. Similarly, students attempted the self-assessment in the final chapter, Chapter 16: Searching and Sorting, near the end of the semester, along with usage spikes in the other chapters, likely to study for the final exam. The trend line indicates a correlation between student self-assessment activity and the times during which those subjects were taught in class and also the assessments assigned during those times.

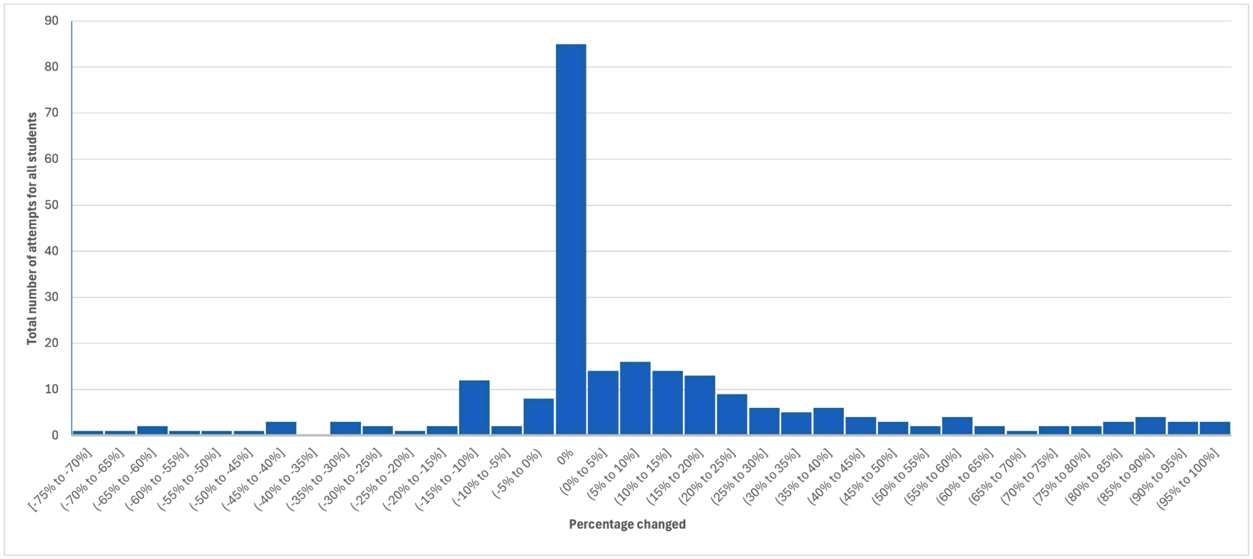


Figure 6: Score changes between successive self-assessment attempts

Figure 6 shows the percentage change when students attempted the same end-of-chapter self-assessment multiple times. Most scores did not change more than 5% between attempts, but most students who attempted the self-assessment more than once did receive higher scores on subsequent attempts.

The total possible score for end-of-chapter self-assessments ranged between 7 and 50 points. For those assessments that were worth less points, getting one or two questions incorrect could result in a higher percentage score decrease between subsequent attempts compared to when students completed assessments with higher possible scores.

End of term survey data

At the end of the course, students were asked to complete a survey regarding demographics and their experience with the self-assessments. 41 students completed the survey. Of those who responded, seven students reported that they completed self-assessments for most or all chapters. 31 students completed self-assessments for only some of the chapters. Three students reported not completing any self-assessments.

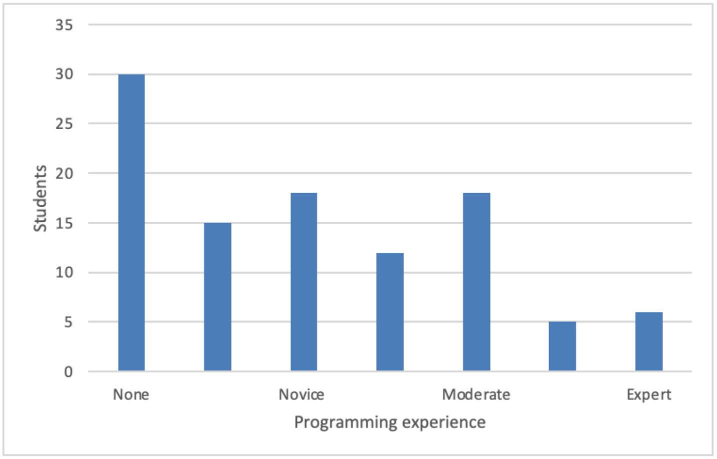


Figure 7: Self-reported prior experience

As shown in Figure 7, about 24% of students indicated no prior programming experience. Less than 5% of the students self identified as experts.

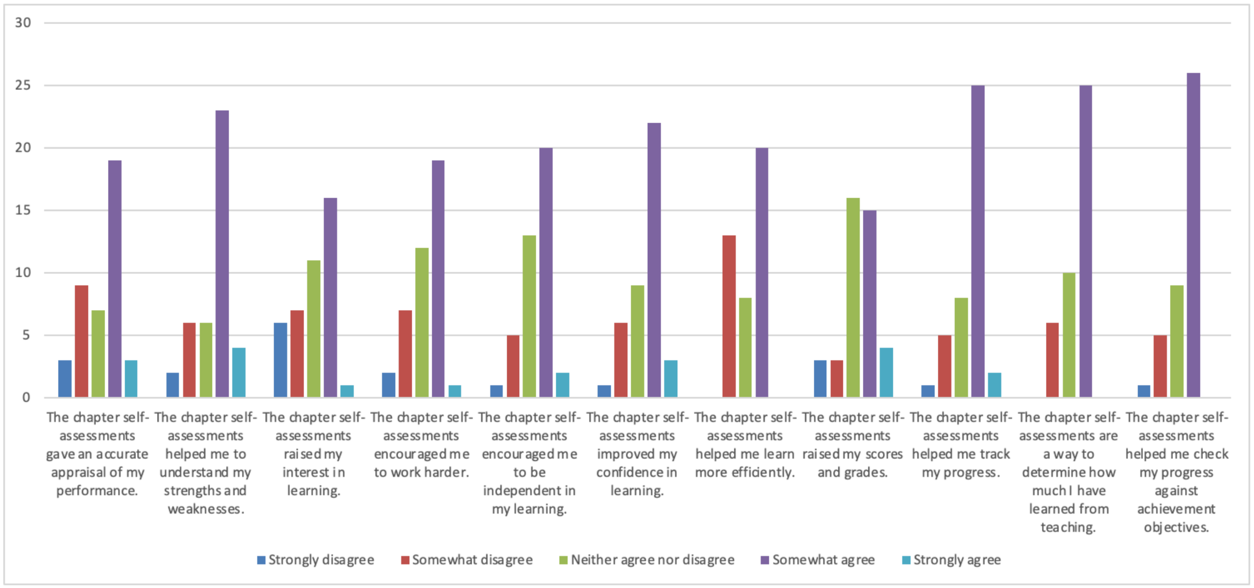


Figure 8: Student reflections

Students' reported reflections following the post-chapter self-assessment surveys are shown in Figure 8. Students completed a post-chapter survey after each self-assessment, reporting their reflections on measures of usefulness on a Likert scale with options: Strongly disagree, Somewhat disagree, Neither agree nor disagree, Somewhat agree, and Strongly agree.

The statements on the survey were:

1. The chapter self-assessments gave an accurate appraisal of my performance.
2. The chapter self-assessments helped me to understand my strengths and weaknesses.
3. The chapter self-assessments raised my interest in learning.
4. The chapter self-assessments encouraged me to work harder.
5. The chapter self-assessments encouraged me to be independent in my learning.
6. The chapter self-assessments improved my confidence in learning.
7. The chapter self-assessments helped me learn more efficiently.
8. The chapter self-assessments raised my scores and grades.
9. The chapter self-assessments helped me track my progress.
10. The chapter self-assessments are a way to determine how much I have learned from teaching.
11. The chapter self-assessments helped me check my progress against achievement objectives.

For almost every question with regard to the effectiveness and impact of chapter self-assessments, students answered "Somewhat agree." With regard to whether the chapter self-assessments helped raise their scores and grades, most students answered "Neither agree nor disagree."

Conclusion and Future Work

We observed a correlation between the self-assessment activity and the times during the semester that students likely had in-class assessments, such as a midterm and final exam, indicating that students used the self-assessments to study. Another interesting observation is a higher number of attempts for chapters like Loops, Lists & Dictionaries, Files, and Recursion. These chapters are arguably more difficult in an introductory Python course.

Most scores did not change significantly (between 0 and 5%) between successive attempts, but we observe that most scores increased between successive attempts. We also note that some scores decreased, but as some assessments were worth as many as 50 points and some were worth only 7 points, large percentage drops could result between successive attempts of assessments not worth many points.

This research is preliminary yet promising. The usage data warrants further investigating the impact of self-assessments in boosting the confidence of an early computer science student. We plan on extending the self-assessments tool beyond being diagnostic to further encourage good study habits and increase self-efficacy for students.

Bibliography

[1] A. Petersen, M. Craig, J. Campbell, and A. Tafliovich, "Revisiting why students drop CS1," in *Proc. 16th Koli Calling Int. Conf. Comput. Educ. Res.*, 2016, pp. 71–80, doi: 10.1145/2999541.2999552.

[2] P. Brusilovsky and S. Sosnovsky, "Individualized exercises for self-assessment of programming knowledge: An evaluation of QuizPACK," in *J. Educ. Resour. Comput.*, vol. 5, no. 3, 2005, Art. no. 6, doi: 10.1145/1163405.1163411.

[3] P. Brusilovsky and S. Sosnovsky, "Engaging students to work with self-assessment questions: a study of two approaches," in *Proc. 10th SIGCSE Conf. Innov. Tech. Comput. Sci. Educ.*, 2005, pp. 251–255, doi: 10.1145/1067445.1067514.

[4] C.-Y. Chung and I.-H. Hsiao, "Investigating patterns of study persistence on self-assessment platform of programming problem-solving," in *Proc 51st ACM Tech. Symp. Comput. Sci. Educ.*, 2020, pp. 162–168, doi: 10.1145/3328778.3366827.

[5] J. Rodríguez-Vidal and Á. García-Beltrán, "Impact of C-coding self-assessment exercises on exam performance: A study in engineering education," in *Comput. Appl. Eng. Educ.*, vol. 32, no. 2, 2024, Art. no. e22706, doi: 10.1002/cae.22706.

[6] J. Rodríguez-Vidal, R. Martínez, and Á. García-Beltrán, "C-programming self-assessment exercises versus final exams: 12 years of experience," in *Comput. Appl. Eng. Educ.*, vol. 31, no. 5, pp. 1272–1288, 2023, doi: 10.1002/cae.22639.

[7] A. García-Beltrán and R. Martínez, "Web assisted self-assessment in computer programming learning using AulaWeb," in *Int. J. Eng. Educ.*, vol. 22, no. 5, pp. 1063–1069, 2006.

[8] S. Bergin, R. Reilly, and D. Traynor, "Examining the role of self-regulated learning on introductory programming performance," in *Proc. 2005 Int. Workshop Comp. Educ. Res.*, pp. 81–86, doi: 10.1145/1089786.1089794.

[9] K. Falkner, R. Vivian, and N.J.G. Falkner, "Identifying computer science self-regulated learning strategies," in *Proc. 2014 Conf. Innov. Tech. Comput. Sci. Educ.*, pp. 291–296, doi: 10.1145/2591708.2591715.

[10] K. Ilves, J. Leinonen, and A. Hellas, "Supporting self-regulated learning with visualizations in online learning environments," in *Proc. 49th ACM Tech. Symp. Comput. Sci. Educ.*, 2018, pp. 257–262, doi: 10.1145/3159450.3159509.

[11] J. Loeber, E. Kazakou, Y. Rajasekhar, A. Hui, and N. Collins, "An analysis on the effectiveness of randomized, auto-graded activities in introductory programming courses" in *Proc. 2025 Conf. Amer. Soc. Eng. Educ.* [Accepted]