Designing Autograders For Open-Ended Assignments In An Introductory Programming CS Coursera Course

8/05/2024

Kevin Alvarenga (Duke University)

Susan Rodger, Kristin Stephens-Martinez, Yesenia Velasco (Duke University)

Nikita Agarwal (University of Wisconsin - Madison)

Arunima Suri, University of Illinois Urbana - Champaign

1. Motivation

Open-ended assignments help students to be creative and enhance their critical thinking skills. These assignments encourage students to apply their existing knowledge in innovative ways. However, it is time consuming for instructors to manually grade these open-ended assignments. It is beneficial, especially in large scale courses, to have autograders that can provide immediate feedback and allow students to reflect on their project in a timely manner. This efficiency not only enhances the student learning but also alleviates the workload on instructors, enabling them to focus more on creating the lecture content. While there are many benefits to using autograders, most of these autograders are meant for assignments with a given single goal/end state. This can work well in higher level CS courses where there is one main objective, or one way to solve a specific problem. However, more open-ended assignments, which allow for more creativity, are common in introductory Computer Science courses. There is a gap in research on autograders for these types of assignments, and it can make it difficult for instructors to grade these assignments.

2. Background / Related Work

2.1 The Use of Open-Ended Assignments

Significant research has been done on using open-ended assignments in introductory Computer Science classes to see how they can impact student motivation, self-efficacy, engagement, grades, and more. There are varying definitions for the term "open-ended assignment", but there is a general agreement on what it stands for. A general consensus is that an open-ended assignment allows students to make their own choices and decisions about aspects of the project, which results in each student having their own solution to a general problem [7]. The use of open-ended assignments is motivated by many different learning theories, including the idea that students build knowledge by looking at prototype models, the constructivist learning theory, and student metacognition [10]. Open-ended assignments have shown to have a positive impact on student motivation, confidence level, and satisfaction [7]. They also have had no impact on student grades, self-efficacy, or attention span. Thus, the research shows that open-ended assignments, at the very least, do not have potential negative effects. [7].

If implemented correctly, open-ended assignments can be beneficial in the learning environment. There is still the potential issue of students lacking interest, which can affect their learning. In 2018, researchers were curious if letting students choose the assignment would aid with their learning. The researchers examined an introductory coding class and noticed that many students did not enjoy the class as the assignments did not align with their interests. To address this, they designed 5 open-ended assignments for students to choose from and noticed that their enjoyment and strength on core concepts increased [3]. However, it was also mentioned that the grading process for these assignments required much work and was tedious for instructors. The researchers noticed that incorporating some form of choice was beneficial for the students in their learning. However, they also acknowledged that there is still room for improvement of the actual grading process. Therefore, the use of choice in these courses has the potential to provide benefits to the student, but there is still no path to designing an autograder capable of grading these assignments.

2.2 The Efficacy of Autograders

Autograders have been helpful in courses with a large roster of students as they reduce the time needed for grading, give clear feedback, and can evaluate complex equations much quicker. Also, autograders can be used to help guide students to improved solutions with less errors, by providing feedback or hints on where to move forward. Research shows that generating hints for open-ended assignments can be difficult due to variability in programs. However, researchers were able to use student data to generate hints specifically for the student and have helped them reach a better solution [5]. This helps maintain the core of open-ended assignments and ensures that students still receive the help they need. Even though this seems like a reasonable solution, there has been less research conducted on the student side of things. It has been found that student perceptions on autograders can affect how well they perform, as well as the experience they have. These students have stated that they would have had a better experience with the autograder if they could directly cater their code for it. [4]. However, this negates the benefits of using open-ended assignments as the main purpose is to promote creativity and new formats.

2.3 Student Perspectives on Autograders

Students often believed that autograders used simple techniques such as general matching and keyword matching [6]. Combined with false negatives, correct solutions being marked as incorrect, students started to develop a distrust of the autograders for their course. This made it frustrating for the students as it made them question their coding skills and standing in the course. However, false positives, incorrect solutions marked as correct, did not reveal the same negative effect on the student's view on the course. Also, students indicated that if they were informed of the autograder's design, then they could cater their code to it and work on more assignments. This indicates an increase in course engagement, therefore, for our study we will ask students for their perspectives on how our autograders work through reflection surveys.

2.4 False Positive/Negative Impact on Students

When developing an autograder, it is best to minimize false positives and false negatives to ensure students are receiving accurate feedback. In an introductory programming course, it revealed that false positives were more damaging than false negatives on a student's learning experience [6]. With false positives, students would less likely admit that their code was incorrect and move on without engaging with course material. However, with false negatives, students would ignore this and engage more with the course to get a correct scoring. So, false negatives encouraged students to reattempt and use the course materials more, even if they felt their code was correct. Therefore, this shows that how a student views the accuracy of an autograder can, on some level, impact their engagement level with the course.

2.5 Limiting Student Submissions

Autograders can vary in how students are able to submit their code. One example would be the number of submissions, or the time in between submissions. It was found in an introductory programming course, that having submissions be tied to tokens that take 24 hours to generate, promoted students to engage with the course materials more [2]. This format moved students away from the trial-and-error mindset, motivating students to take the time to debug and find new solutions. This increased course engagement and students were overall satisfied with the autograder.

A different approach was taken in 2022, where researchers developed an algorithm that could grade code based on different aspects such as lines of code, spacing, variable names, string count, etc. [9]. This helped with a course where all assignments had visual and graphical output, making it difficult for an autograder to correctly grade them. This algorithm focused on classifying the elegance of a student's code, which, while important, is not a sufficient criteria to use for grading open-ended assignments. This algorithm did not take into account the actual output of a student's code which is the main concern when it comes to these types of assignments.

Given the previous research out there, it is clear that there lacks an integration of autograders into open-ended assignments for introductory computer science classes. Since these classes are having larger and larger student enrollment, assignments simply cannot be graded by hand. Therefore, it is critical that autograders be introduced for open-ended assignments as open-ended assignments are an important way to ensure that students are able to be creative and stay motivated as they complete assignments. In other words, the main problem lies in a lack of existence of autograders that can handle student creativity, so our work aims to fill this gap through building and documenting autograders for open-ended assignments [8].

3. Design Process

My main role for this project was to use Python and Pytest, a testing framework, to develop autograders for open-ended assignments. These assignments were for an introductory college-level programming Coursera course taught in python. My team was able to design multiple autograders and tested them in the Coursera environment. The main objective was to ensure that the assignments can be open-ended but also still feasible to grade. We wanted to ensure that students had creative freedom in their learning and were still given support to improve. Currently, there isn't much research on designing autograders for open-ended assignments, which is where our process comes in.

3.1 Open-Ended Assignment Example

This Coursera course uses open-ended coding assignments for students to learn Python and programming skills. An example of an open-ended assignment would be having students design a store with certain items for sale. In this assignment, students are required to code functions that allow the user to grab certain items for their cart, get the price of each individual item, and apply coupons to some items. Therefore, this assignment, besides having a couple required functions, is fully dependent on the student and their choice of item and pricing. An approach a student can take is to first decide the items and their price. Their next step would be to decide the format of presenting the item and price to the user. This format could be a numbered list, a regular list, or even just a one line statement. They could also print out pictures of the items to the terminal if they choose to do so. The required functions only need to return the desired value, but the student can print out the values in any form they wish. So, this assignment remains open-ended since the student can decide how their store presents and sells the items of their choosing. As long as they have the required functions, they can add any other function or anything to their program that still allows for it to execute.

3.2 Development Process

Each lab/assignment had a list of requirements that a student needed to follow. These requirements were pretty basic as we still wanted students to be creative with their programs. From the student view, the requirements were separated into two categories, required and exceeds expectations. The required test cases made up 80% of the possible points, while the remaining 20% came from the exceeds expectation category. A student only needs to get 80% to pass the specific lab/assignment.

The first step was to design sample student solutions, where there was a mixture of correct and incorrect solutions. The incorrect solutions were designed based on common errors students could make in their program. These solution files are also focused on capturing edge cases that the autograder would have to consider. So it was best to create these solution files before the testers to ensure the testers can properly handle the sample solutions. These solution files were constantly changing and new ones were created throughout the development process.

In order to build the testers, we had to keep the requirements in mind and we split our test cases into three categories: pretests, required, and exceptional. Pretests are run before the student's program gets tested with the listed requirements. These tests are meant to check for errors, like compile errors, lack of input, too much input, or just general bugs that can cause the program to not execute properly. If a student fails even one of the testers here, then they receive a score of 0, with specific feedback on why this occurred. The required and exceptional test cases reflect the requirements of the lab and all of this was through the use of Pytest.

After all student solutions and testers were created, we will assess them in an admin view. We execute all solution files in the terminal and see if the error message received matches the expected error message for that solution file. Once all solutions have matched the expected error message, then we move into the student view. We test our autograder and submit student solutions to see if the error messages are consistent. Once this is completed, then we move into the code review process where more edits are made to refine the autograder.

3.3 Autograders I Have Developed

Decision Advisor with Randomness

This autograder was a group task where I worked with the other two undergraduate team members. This lab was focused on student's getting used to boolean expressions and the use of random in their program. Since the lab incorporated random, the criteria to pass this lab was kept to a minimum as we cannot specifically test for randomness. Instead we tested to see if they used the correct conditionals and if they are asking for the right number of inputs.

Generate Art with While Loop

This lab is focused on students obtaining and verifying user input to create an art output. Once the student verifies the input, they will use a while loop to continuously build output lines. The autograder for this lab followed the format mentioned above but there was a difference. There were a lot of edge cases and variability in the art output that students could make. Students could have interruptions in their art, they could do multiple lines, they could do one line, or they can use a for loop instead of a while loop. Therefore, I had to avoid assumptions about the students output and focus on analyzing all parts of the output and code. The testers would look for all small parts and see if they all exist at the same time.

Debugging

This autograder was also assigned as a group task but this one was a little different from the other autograders. For this lab, students were given an incorrect program and were asked to remove all bugs. These bugs include compile errors, logic errors, and general errors that can prevent the program from executing correctly. We still used the same format to test the student solutions.

Generate Art with While and Indexing - Part 3

This lab focused on students verifying user input and using parts of it to build some art. The student will take the string input and use indexing to grab specific information from it, like the art symbol, number of occurrences, and the use of newline characters. The autograder for this lab is similar to the other lab focusing on generating art, in that there aren't many assumptions. However, the main difference is that the autograder needed to test for the use of if statements instead of a while loop. This lab had more criteria to follow and there was much more overlap in the testers, so the ordering of the testers was crucial here.

Transforming Words and Practicing the Seven Steps

This lab was focused on students transforming words into pig latin and text message slang. The conversions depended on the placement of vowels and consonants. The autograder for this lab had the most sample student solutions as students were given four functions to write. Any error that was applicable in one function was applicable in the other three. There was a lot of overlap in the criteria for the functions that made it a little complicated in writing the exceptional testers.

For loops and Range

This lab focused on having students build a store with certain items that users can pick from and get the price for. The main difference with building this autograder is that I needed to figure out what items the student is using. To do this, I tested one of their functions and passed in all possible items and saw which ones the student was returning. Once grabbing these items, I used them as input to all testers in the three test files. This ensures that we are properly assessing the student's program with the items they decided to use.

4. Interested Research Question

Currently, the Coursera courses are still in development and will not be launched until a later date. Therefore, we cannot collect nor perform data analysis. However, once the courses are launched there is a particular area of interest that I would like to focus on. The research question of interest is: How does a student's perspective on the reliability of the autograders affect their willingness to revise and complete the open-ended assignments in a CS1 course?

4.1 Motivation

Open-ended assignments in CS1 courses give students the opportunity to test their skills in a creative manner. However, this creates a lot of variability in student programs and would require for an autograder to be adjusted to this. These open-ended assignments would need to have some basic criteria for an autograder to follow, which again can vary in design. Also, having open-ended assignments be graded by an autograder, would mean that for a student to make improvements, they must trust and use the feedback given. The feedback can vary in its usefulness for open-ended assignments as there is much it would need to test and possibly ignore. Therefore, it is essential for the student to build a sense of trust with the autograder as it would help promote a learning environment that is less stressful and more focused on engaging with the course materials.

Moreover, student perspectives on the reliability of the autograders can help pinpoint areas of weakness. By analyzing student feedback, instructors can find these areas and make the necessary adjustments to improve them. Addressing these weak points can lead to improvements in the accuracy and reliability of these autograders for open-ended assignments. Such improvements can enhance the learner experience for students which can alter their perspective on the autograders. Altering their perspectives on the autograders can be beneficial as it could have an effect on the student’s level of engagement and willingness to complete assignments in the course. Analyzing these student perspectives on the autograder can provide valuable information on the autograder strengths and weaknesses, while also revealing how the efficacy the autograders affect course engagement.

4.2 Methods

To gain student perspectives and experiences, we implement a variety of surveys throughout the course. First, we discuss the use of reflection surveys after each assignment. Next, we discuss the use of an end of course survey to gain a general feeling of the course. Finally, we discuss how survey data with Coursera student data can be used to see how student perspective can impact their learning and performance in the course.

4.2.1 Reflection Surveys

To avoid recency bias, students will be prompted to take an optional survey through Qualtrics or Google Forms (haven't decided yet) for reflection upon each assignment completion. This survey is meant to grab their perspective of the efficacy of the autograder for that particular assignment. The following four questions are asked:

Autograder Accuracy: To gather views on the accuracy of the autograder, a Likert style question is asked, with responses ranging from very accurate to very inaccurate: "How accurately do you feel the autograder evaluated your assignment?" This question is also helpful in determining a student's level of trust in the autograders for this course. This can be kept in mind when analyzing their responses to future questions and how they use the autograders.

Ease of Use: It's also crucial to gather any notices of difficulty in using the feedback provided from the autograder. This can determine if an autograder needs to be reworked or any common weak points. The following question is asked with a similar Likert style format: "How difficult was it to understand and use the feedback provided by the autograder?"

Assignment Length: To gather a student's level of satisfaction with the respective autograder, another Likert style question is asked: "How long did it take you to complete this assignment?" This question has time ranges of hours and it is self-reported, which we can compare with the Coursera data.

General Thoughts: If a student decides to provide more information about their experience that wasn't asked above, then they can provide it here in a text box.

4.2.2 End of Course Survey

To focus more on student's experiences, an end of course survey will be given out (in the same manner as the reflection surveys) where students will be asked to share their thoughts on the course.

Autograder Satisfaction: To gather a general view on the course autograders the following question is asked: "How satisfied are you with the performance of the autograders in this course?"

Bad Assignment Experience: We want to collect data regarding bad experiences a student had with a particular assignment/autograder. The question asked for this section would be: "Was there an assignment where you felt the autograder performed badly? If so, which one?" This is to see if there was a point where a student had a bad experience. We can use this as a checkpoint to see if their engagement (based on Coursera data) was altered after this encounter.

Usefulness to CS Courses: This section is meant to see how students view the efficacy of autograders in their learning and other student's learning. The first question here asks "Do you feel the autograders were fair and useful to your learning?" This will be dependent on their experience with the autograders and it gives insight into their perspective on the autograders. The second question: "Do you feel that autograders were reliable and can be used for future courses?" This is a direct question focused on gathering the students' perspective on the reliability of the autograders.

Final Thoughts: Students will be asked a final question where they can provide their thoughts in a text box. This is a space for students to elaborate on experience or provide general feelings about the course or the autograders.

4.3 Analysis of Student Responses

Coursera provides student data like number of submissions per assignment, time in between submissions, how long a student took on assignments, and their overall performance in the course. By analyzing the student responses to the reflection and end of course surveys, we can see if there is a pattern between perspective on reliability and performance.

5. Future Work

The first course will be launched this fall where students can learn the basic programming concepts. We will collect data from Coursera and perform data analysis in the spring (2025). The analysis of this data will be used to answer this research question and the research questions of my teammates. Also, the development of new autograders for the other three courses will continue and they will undergo a similar process in deployment and analysis.

6. References

1. Gupta, Surendra, and Shiv Kumar Dubey. "Automatic assessment of programming assignment." Computer Science & Engineering 2.1 (2012): 67.
2. Juho Leinonen, Paul Denny, and Jacqueline Whalley. 2022. A Comparison of Immediate and Scheduled Feedback in Introductory Programming Projects. In Proceedings of the 53rd ACM Technical Symposium on Computer Science Education - Volume 1 (SIGCSE 2022), Vol. 1. Association for Computing Machinery, New York, NY, USA, 885–891. https://doi.org/10.1145/3478431.3499372
3. Sohail Alhazmi, Margaret Hamilton, and Charles Thevathayan. CS for All: Catering to Diversity of Master’s Students through Assignment Choices. ACM Technical Symposium on Computer Science Education, 2018.
4. Silas Hsu, Tiffany Wenting Li, Zhilin Zhang, Max Fowler, Craig Zilles, and Karrie Karahalios. Attitudes Surrounding an Imperfect AI Autograder. CHI Conference, 2021.
5. Thomas W. Price, Yihuan Dong, and Tiffany Barnes. Generating DataDriven Hints for Open-Ended Programming. International Educational Data Mining Society, 2016.
6. Tiffany Wenting Li, Silas Hsu, Max Fowler, Zhilin Zhang, Craig Zilles, and Karrie Karahalios. 2023. Am I Wrong, or Is the Autograder Wrong? Effects of AI Grading Mistakes on Learning. In Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1 (ICER '23), Vol. 1. Association for

Computing Machinery, New York, NY, USA, 159–176. https://doi.org/10.1145/3568813.3600124

7. Sadia Sharmin, Daniel Zingaro, and Clare Brett. Weekly Open-Ended Exercises and Student Motivation in CS1. Koli Calling, 2020.
8. Sadia Sharmin, Daniel Zingaro, Lisa Zhang, and Clare Brett. Impact of Open-Ended Assignments on Student Self-Efficacy in CS1. ACM Conference on Global Computing Education, 2019.
9. Sirazum Munira Tisha, Rufino A. Oregon, Gerald Baumgartner, Fernando Alegre, and Juana Moreno. An Automatic Grading System for a High School-Level Computational Thinking Course. International Workshop on Software Engineering Education for the Next Generation, 2022.
10. Tammy Vandegrift. Encouraging Creativity In Introductory Computer Science Programming Assignments. 2007 Annual Conference Exposition, 2007