[Work-in-progress] Towards AI-assisted CS1 classrooms: An experience report on the design of learning activities and assessments in engineering computing courses

Ayush Pandey

School of Engineering
University of California Merced.
Email: ayushpandey@ucmerced.edu

July 25, 2025

Abstract
With generative AI tools commonly available to students, instructors must update the in-class activities and assessments accordingly such that AI can be integrated as a pair programmer. In this experience report, we describe the outcome of AI-assisted active learning in CS1 courses and novel assessment designs to maximize creative exploration. Conventional CS1 assessments can be solved with iterative prompting of generative AI tools like ChatGPT or completed on the fly with Github Copilot. To address this, we present the design of a new assessment strategy in introductory programming courses where each student works on an open-ended problem for their summative assessment. We design generalized scaffolds (project proposal, schematic development, pseudocode, integration of files, and graphs) for these open-ended assessments so that each student completes a project of desired complexity. On implementation, we find that students pursued independent projects driven by their interests and passions. This approach encouraged creative work and enables new assessment methods in CS1 (introductory programming) courses.

1 Introduction

It is well established in computer science (CS) education literature [1], that learning-by-doing and rigorous practice are effective for students to gain programming expertise. Consequently, the formative and summative assessments in CS courses often take the form of programming tasks. These assignments usually have fixed inputs and outputs, making them amenable to autograders — software-based automated grading. Autograders provide immediate feedback to students, reduce teachers' workload, and if implemented carefully, can be fair to all students. These properties make autograders effective for fostering a mastery learning framework [2] in CS education. For example, students can keep on working on mastering a skill in an autograded formative loop until they get the desired grade.

To enable richer hands-on experiences, instructors may choose to assign open-ended projects. Learning-oriented assessment studies [3] have shown that such open ended assignments are close to students' interests and can lead to an increase in student engagement and agency.

1.1 The changing landscape of CS1 assessment

With the general availability of generative AI, summative assessments in CS1 have to be designed with the understanding that the students can use these tools to complete the exams. Rather than switching to timed in-person or similar "stricter" settings to prevent AI use, CS educators must innovate the assessments such that generative AI tools can be used collaboratively. Open-ended projects offer a path forward. Research has shown that projects as the main summative assessment in engineering and computing courses can have a greater positive impact on student learning and career preparation than conventional exams (needs citations).

1.2 The case for open-ended projects

With open-ended projects, the idea is to go one step further. If a project problem statement is carefully designed by the instructor and provided to the students, generative AI tools could have a much easier time finishing the project, especially in CS1. Identifying a problem, exploring possible solutions, and defining what a good real-world implementation would look like are the creative elements of any project. Therefore, I designed open-ended project-based summative assessment exams for two CS1 courses.

The open-ended projects can be designed to be creative and exploratory in nature. I gave a set of constraints to the students for their Python projects but the project goals, specific objectives, and the technical approach were assigned as tasks for the students. The project constraints for the ME021 class were:

All students must work on an independent Python project. This project must demonstrate the following five key elements of Python that you learn in ME 021. An ideal project that grades 100 will

1. *Control the program flow with branching and loops*
2. *Uses correct data structures for optimal computations*
3. *Uses functions for modular code*
4. *Loads data from files (real-world data is preferable) and writes outputs to files (if needed)*
5. *Documents the flow of logic with comments, docstrings, and user-friendly messages*

Beyond these constraints, the students were given the independence to propose a project and develop it for their exam. Two main challenges with this style of summative assessment are: lack of clarity for the students on "what do I do?", and the design of a fair grading strategy and rubrics. To address the first challenge, I provided a list of project ideas to the students. The students were also encouraged to propose their own project ideas. Scaffolds were designed carefully for the project so that students could make consistent progress and achieve the best results.

1.3 Milestone assignments as project scaffolds

A total of 5 milestone assignments were designed as scaffolds for this project-based summative assessment. The first scaffold was a "project proposal" assignment. In this assignment, the students wrote a Python code that asked themself questions about the project being proposed. Students submitted their code and their responses as the code ran on the terminal.

Project proposal: The assignment asks students to write a Python code with the following prompts:

1. Ask the user to input the goal of their ME 021 project in 1 line. Instruct the user that this goal must be succinct and should describe the big-picture problem that the project is addressing.
2. Ask the user to write down inputs and outputs of the project. That is, what are the particular objectives that this project will achieve.
3. Ask the user to input the data that this project would need.
4. Ask the user whether they met with their course TA to discuss the project or not (Yes or No).
5. Check whether the user answered all four questions. If yes, then output...

The other milestone assignments addressed each of the constraint in the exam instructions (adding branching, loops, functions, files, and visualization).

1.4 Categories of proposed projects

A total of 207 projects were proposed by the students in the ME021 class. The projects can be classified into the following categories: game design (tetris, tic-tac-toe, rock paper scissor, adventure games), computational apps (computing math formula, finance calculators), data analysis (COVID-19 data analysis, stock market analysis), and simulations (UC Merced bus tracker, diet simulator, class registration system). A full list of all projects is available online on GitHub [4].

The second challenge in this exam design is the grading. I designed a grading rubric that was shared with the students before the exam. The rubric was designed to be holistic and included the following criteria: whether the code runs successfully given the desired inputs, significant contribution (subjectively assessed by graders), optimal use of data structures, and correct use of branching, loops, and files/visualizations. A demonstration of the project to a TA was a required grading component for the project. The oral exam ensured the authenticity of the exam and challenged students to present their approach. The TAs were provided with a list of generic question starters (see GitHub [4]). To summarize, the grading of the exam consisted of: 5 milestone assignments, 1 demonstration oral exam, and 1 final project submission (graded on a rubric). The time commitment for grading was high and that is the main limitation of this open-ended exam design, which calls for more research on automatic graders.

Table 1: Open-ended project description

Project Assignments	Description
Milestone 1	Set up computer
Milestone 2	Propose project idea (or choose from provided list)

Milestone 3	Create the program framework and I/O structure
Milestone 4	Integrate loops and required logic
Milestone 5	External data and personalize
Demonstration 1	Oral exam
Demonstration 2	Class/lab presentation
Demonstration 3	Skill quiz
Final Project Submission	All files and cover sheets

The open-ended projects can be designed to be creative and exploratory in nature. We gave a set of constraints to the students for their Python projects but the project goals, specific objectives, and the technical approach were assigned as tasks for the students. The project constraints were:

All students must work on an independent Python project. This project must demonstrate the following five key elements of Python that you learn in ME 021. An ideal project that grades 100 will

1. *Control the program flow with branching and loops*
2. *Uses correct data structures for optimal computations*
3. *Uses functions for modular code*
4. *Loads data from files (real-world data is preferable) and writes outputs to files (if needed)*
5. *Documents the flow of logic with comments, docstrings, and user-friendly messages*

Beyond these constraints, the students were given the independence to propose a project and develop it for their exam. Two main challenges with this style of summative assessment are: lack of clarity for the students on "what do I do?", and the design of a fair grading strategy and rubrics. To address the first challenge, we provided a list of Final Dataset project ideas to the students.[1] The students were also encouraged to propose their own project ideas. Scaffolds were designed carefully for the project so that students could make consistent progress and achieve the best results.

1.5 Milestone assignments as project scaffolds

A total of 5 milestone assignments were designed as scaffolds for this project-based summative assessment shown in Table 1. The second scaffold was a "project proposal" assignment where the students proposed the project. The other milestone assignments addressed each of the constraints in the exam instructions (adding branching, loops, functions, files, and visualization).

1.6 Categories of proposed projects

A total of 183 projects were proposed by the students. The projects can be classified into the following categories: game design (tetris, tic-tac-toe, rock paper scissor, adventure games etc.),

[1] List of project ideas are available on GitHub.

computational apps (computing math formula, finance calculators etc.), data analysis (COVID-19 data analysis, stock market analysis etc.), and simulations (bus tracker, diet simulator, class registration system etc.). A full list of all projects is available online on GitHub [4].

1.7 Grading and rubric

The second challenge in this assessment design is the grading. A detailed grading rubric was shared with the students before the exam, and a short summary of the rubric is shown in Table 3. Due to space constraints, the details of each rubric item are not shown. In summary, we designed open-ended project-based summative assessment exams for a CS1 course taught in Fall 2023 in a course with total enrollment of 207. Note: not all students who were enrolled submitted a project.

Table 3: Rubric for open-ended assignment

Criteria	Ratings	Points
Execution: Does the code run?	20 if yes, 15 for logical errors, 10 if ∼50% runs, 0 if not.	20
Branching: Does the code use if-else and loops as needed?	15 if criteria satisfied, 10 if only one, 0 if none.	15
Modularity: Does the code use functions for modularity?	10 if modular, 5 if improper, 0 if none.	10
Structures: Does the code use data structures as needed?	5 if appropriate DS, 0 if not.	5
Files/Advanced Features: Does the code use files/advanced features?	10 if criteria satisfied, 5 if attempted, 0 if not.	10
Documentation: Is the code well-documented?	5 if comments present, 5 if attempted, 0 if not.	10
Demo: Did the code demo the proposal?	25 if yes, 20 if one missing, 15 if ∼50%, 10 if one feature only, 0 if none.	25
UI: Effective UI?	10 if yes, 0 if not.	10

References

[1] T. J. Feldman and J. D. Zelenski, "The quest for excellence in designing cs1/cs2 assignments," *ACM SIGCSE Bulletin*, vol. 28, no. 1, pp. 319–323, 1996.

[2] J. Garner, P. Denny, and A. Luxton-Reilly, "Mastery learning in computer science education," in *Proceedings of the Twenty-First Australasian Computing Education Conference*, 2019, pp. 37–46.

[3] J. DeNero, S. Sridhara, M. Pérez-Quiñones, A. Nayak, and B. Leong, "Beyond autograding: Advances in student feedback platforms," in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 2017, pp. 651–652.

[4] A. Frias and A. Pandey, *Github for open-ended assessments*, 2024. [Online]. Available: https://github.com/pyEdTools/flexigrader.