# An Online Tool for Easy-to-Set-up and Auto-gradable Full Tracing Exercises

Wei Jin
Department of Information Technology
Georgia Gwinnett College
Lawrenceville, GA 30043

## I. Problem Description

Understanding how a programming construct executes is a prerequisite for coding. Tracing exercises are often used to help students develop accurate mental models of how different constructs execute [1, 2]. With an incorrect or non-existent mental model, it is impossible for a student to take the next step: write programs to solve problems.

Full tracing exercises are often paper-based, which is difficult to grade. Auto-gradable line-by-line full tracing exercises have also been used, in either the fill-in-the-blanks or the multiple-choice quiz format. They ask students to identify which line of code is executed next and what that line of code does in memory or output. Such exercises, even though auto-gradable, are very time-consuming and tedious to prepare. From the PI's personal experience, only one or two such full tracing exercises were made for each topic and were used for several semesters.

Due to the time-consuming nature of setting up quizzes that ask students line-by-line tracing questions, instructors do not use this type of full-tracing exercises. Instead, they simply ask students to predict the final result of a segment of code. The drawback of this simpler approach is the lack of feedback. For a wrong prediction, the system simply marks it as such without telling the student exactly where in the tracing process they made a mistake.

## II. Project Summary

We proposed to develop a web-based system that allows instructors to set up auto-gradable full tracing exercises easily. This report describes the current status of the project and our planned actions in the near future.

We augmented pythontutor.com [3], a popular open-source program visualization system to deliver auto-gradable tracing exercises. Pythontutor.com helps students understand the semantics of each programming construct (e.g. an if statement, a loop, or an array) in that it not only shows the order of instructions to be executed, but also shows what each instruction does to memory/output graphically (e.g. a variable is represented as a small box and a stack frame as a big rectangle).

The new system requires a student to determine which line is executed next and what happens in memory/output before the system demonstrates the step. The system also assigns a grade for a tracing activity by subtracting points for mistakes. These features transform a program visualization activity into an auto-gradable tracing exercise. We hope that this visualizer-based system will help make full tracing activities to be used more often as part of the teaching/learning process.

## III. Project Status

Before the project started, we had a prototype system, which could do the following:

- It allowed students to click a line which they thought would be executed next.
- The system then presented a multiple-choice question as to what happened next. The choices were new output, new variable allocated, variable updating, new stack frame, or return.

- After that, the system presented a follow-up question, such as what the new output is, the name of the new/changed variable and its value, or the name of the new stack frame, etc.
- Each step of execution had assigned point value and the initial grade for an exercise was 100. Each incorrect answer resulted in points deducted.

In addition to full testing of the functions described above and improving the appearance of the interface, we have implemented the following new features:

- Added follow-up questions about return statements.
- Added questions for array tracing.
- Added questions for objects and class variables.
- Added capacity to deduct points when a wrong line is selected as the next line to be executed.
- Resolved miscellaneous issues, such as dividing a for-loop head into three lines to allow proper tracing of each component of the header and addressing issues related to the closing right bracket for a block.

This system can be accessed at http://programtracing.altervista.org/visualize.html. Students can practice full tracing exercises on programs that can be seen in a typical introductory programming course. This is the barebone system that can be accessed by anyone without a login. Therefore, the grade a student receives is not automatically shared with an instructor. Even so, the system could still be used for graded tracing assignments by asking students to share the screenshot of a completed exercise with their instructor.

To facilitate recording students' grades, we developed an online registration system that allows an instructor to compose tracing quizzes. Each quiz can consist of one or more programs to be traced. As a proof of concept and also due to the limited scale of the project, the system supports only a single instructor and one class. The system was deployed at https://tracingquiz.xyz. We have started working on extending the system into a multi-instructor and multi-class system. Please see Section V for more details.

In summer 2020 and the early part of fall 2020, we completed major planned development tasks described above. In the rest of fall 2020, we deployed the prototype system in the PI's two sections of *Programming Fundamentals* to facilitate testing of the system. Several issues were discovered, debugged and resolved during the semester. The students were using the traditional quiz-based full tracing exercises for the first part of the semester. They then used the new visualizer-based system for the remainder of the semester. Therefore, they were able to compare the two formats. Due to the convenience of creating tracing exercises, there were much more visualizer-based tracing exercises available than the quiz-based tracing exercises.

## IV. Project Outcome

We collected students' survey evaluation of the system at the end of fall 2020. Since the system was introduced in the middle of the semester, the exercises were for extra credit. There were altogether 16 students who consented and participated in the survey at the end of fall 2020. We asked them to self-identify themselves into the following three groups:

- Group 1: Did not do majority of the new visualized tracing exercises.
- Group 2: Completed majority of the visualized tracing exercises.
- Group 3: Completed all the visualized tracing exercises.

Even though it is a small sample size, we could still see clear patterns along the group divisions. Figure 1 shows how the three groups evaluated the usefulness of the two formats of tracing exercises: the quiz format that they used for the first half of the semester and the new visualization format for the second half of the semester. We can see that the more a student used the new system, the more useful they rated it. Group 3 gave the new system an average rating of 6.875 out of 7 (i.e. 98%), while group 2 and group 1 both gave

an average rating of 5 out of 7 (i.e. 71%). The difference among the three groups, despite the small sample size, is marginally significant [F = 3.166, p = .076].

The blue bars in Figure 1 are the average group ratings for the traditional quiz format. While it is not surprising that group 1 rated this format highest among all groups, it is interesting that group 3 rated the format higher than group 2. It is possibly due to students' different attitudes towards tracing exercises in general.

Figure 2 presents the information contained in Figure 1 in a different format. We calculated the rating increase from the traditional quiz format to the visualizer-based format for each student and then calculated/plotted the average of this value for each group. We can see a clear increasing pattern from group 1 to group 3.

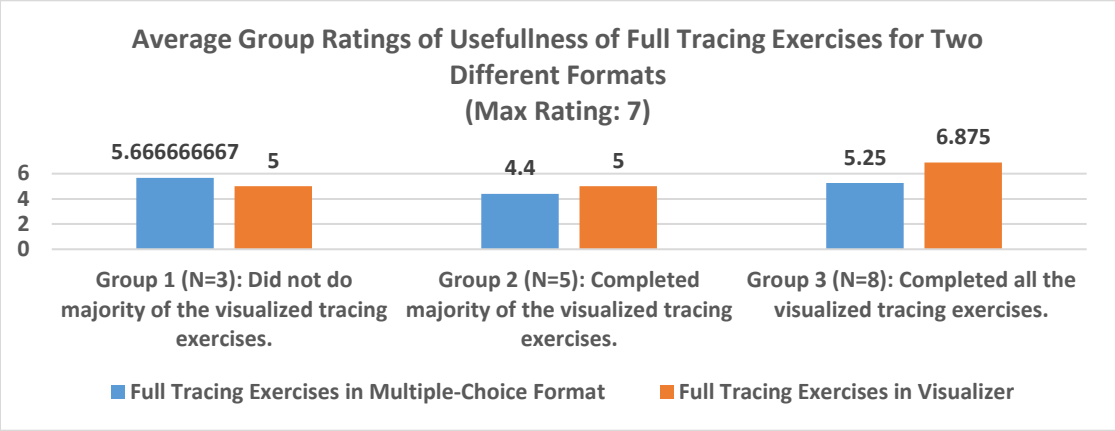


*Figure 1. Evaluation of the usefulness of the tracing exercises in two formats: the old multiple-choice format and the new visualization format*

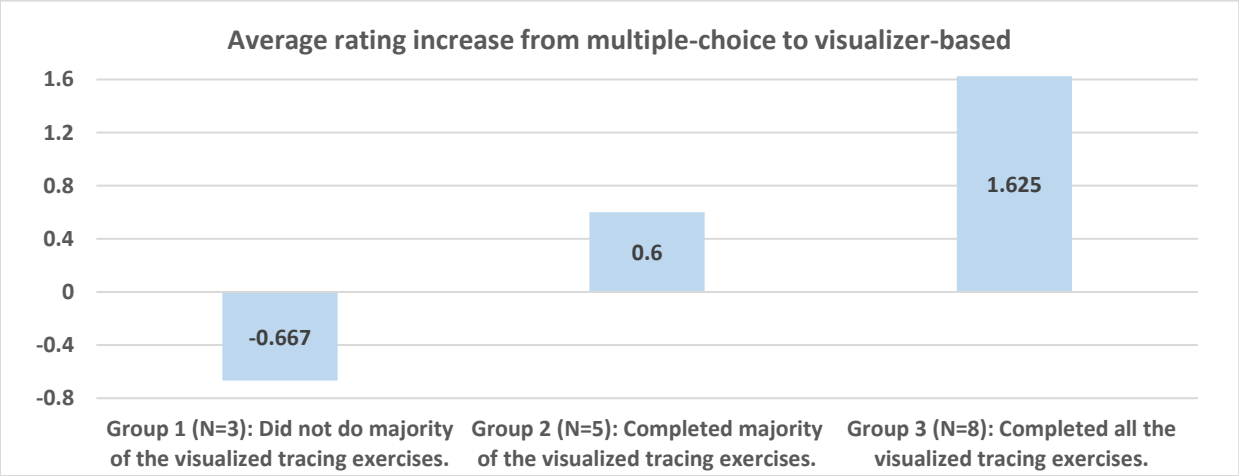


*Figure 2. Average rating increases for three groups.*

Figure 3 shows what the three groups think in general of the usefulness of tracing exercises for learning to program. It is a five-scale answer, with 1 for definitely no, 3 for neutral, and 5 for definitely yes. We can see a clear increasing pattern from group 1 to group 3. Group 3's average is 4.87 out of 5 (i.e. 97.5%), an almost definitely yes, while group 1's is close to neutral. The difference among the three groups is statistically significant [F = 7.725, p = 0.006]. It is not clear whether group 1's neutral attitude is due to that group 1 did not have enough experience with the new format. We will be interested to see whether students will rate tracing exercises better when only the new format is used in the future.

We also asked students to select which format is easier to use, more intuitive, and more helpful. Each student gave all three measures exactly the same choices. From Figure 4, we can see that for students who used the new format more often, it was rated higher. The difference among the three groups is statistically significant [F = 4.510, p = .033].

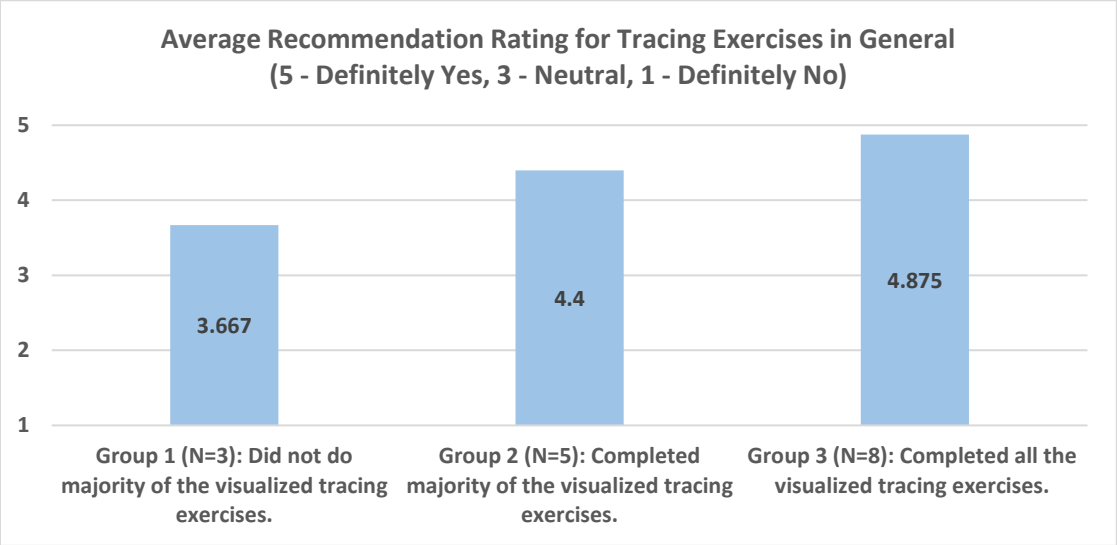


*Figure 3. How different groups recommended use of tracing exercises in the learning process.*

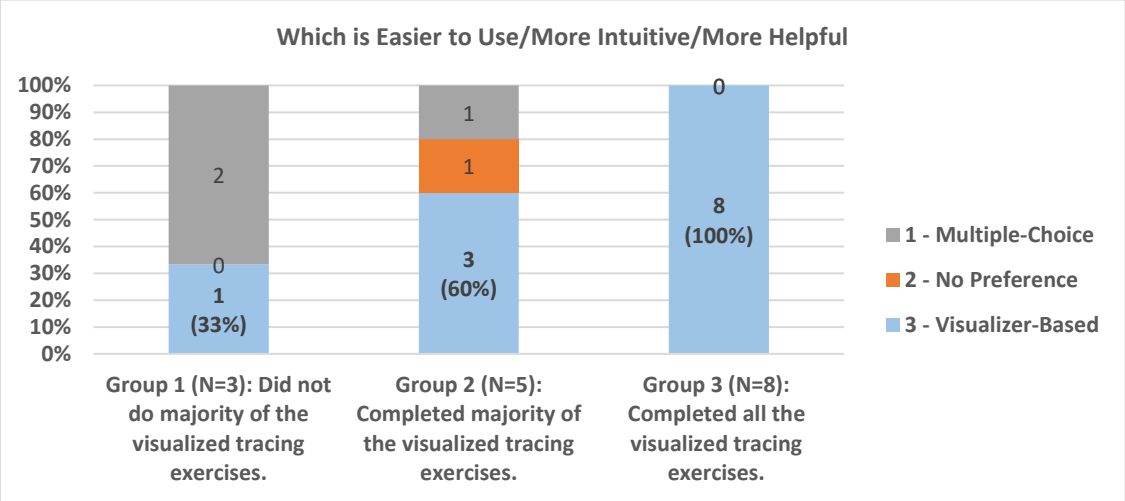


*Figure 4. Groups ratings for which format is more helpful.*

## V. Budget and Justification

The budgeted $5000 has been fully spent by the end of Fall 2020.

- We originally budgeted $750 for an industry consultant. However, the industry person offered to volunteer to help us.
- We moved the consultant fee to student summer/fall pay. We hired two undergraduate students at GGC and they received a total of $3400 for their work in the summer and fall, but mostly in the summer.
- In Summer 2020, the PI worked with students closely in weekly meetings and received the payment of $1600.

## VI. Plan for Extending the Project and Dissemination

The sections above document what we have accomplished through the SIGSE grant. We plan to extend the system to make it useable by the SIGCSE community. We are in the process of improving the online registration system to allow multiple instructors to use the system. We are also addressing a couple of issues critical for a larger scale deployment. We hope to be able to start a larger-scale study toward the middle or end of February 2021. The extended system will be deployed at https://tracing-quiz.herokuapp.com/. The old system https://tracingquiz.xyz will not be supported.

We plan to advertise the new system at SIGCSE'2021 in a SIGCSE'2020 Rewind session and in the SIGCSE'2022 conference through a paper presentation and/or a demo session. We will also actively seek instructors who are willing to adopt the tool in their classrooms before and after the conference presentations.

## References


[1] Benjamin Xie, Greg L. Nelson, and Andrew J. Ko. 2018. An Explicit Strategy to Scaffold Novice Program Tracing. In Proceedings of The 49th ACM Technical Symposium on Computing Science Education, Baltimore, MD, USA, February 21–24, 2018 (SIGCSE'18), 6 pages.

[2] Matthew Hertz and Maria Jump. 2013. Trace-based teaching in early programming courses. In Proceedings of the 44th ACM technical symposium on Computer science education, Denver, Colorado, USA, March 06 - 09, 2013 (SIGCSE'13), 6 pages.

[3] P. J. Guo. 2013. Online Python Tutor: Embeddable web-based program visualization for CS education. In Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13), Pages 579-584. ACM.