PyBuggy: Testing the Effects of Enhanced Error Messages on Novice Programmers

Rachel D'souza, Angela Zavaleta Bernuy, Brian Harrington

Department of Computer and Mathematical Sciences, Univeristy of Toronto Scarborough

Abstract

Several studies have shown mixed results when presenting enhanced (simplified or extended) error messages to introductory programming students. In this work, we detail a pilot study using a tool specifically designed to capture data about students presented with different error messages. Initial data indicates that the tool is capable of capturing relevant data, and that future studies may show an impact of modifying error messages on novice error recovery.

Problems and Motivation

- Error messages (EMs) can be an important source of guidance for novice programmers.
- Unfortunately, the technical syntax and brevity of most EMs render them inadequate for use as a learning tool, which can serve as a source of discouragement and hinder learning [1].
- Several studies have attempted to determine if varying EMs help students with mixed results. [2, 3, 5].
- *PyBuggy* [4] presents debugging problems to introductory computing students, and provides them with differing suites of EMs, while tracking variables of interest including number of attempts, completion rates, elapsed time and correctness.

Approach and Uniqueness

- Our study was conducted at the University of Toronto Scarborough in an introduction to programming course for non majors, where 123 students volunteered to participate.
- *PyBuggy* consisted of 5 pieces of 'buggy' Python code which students needed to fix in a period of 2 hours.
- Students were assigned to one of the three experimental conditions: default, simplified and enhanced Python EMs.
- Simplifieed EMs were made by removing much of the technical terms from default EMs especially in the Traceback section. The enhanced EMs were designed by a TA, explaining the nature and possible cause of the error to a student.
- In Figures 1, 2 and 3, we show the three different versions of the same EMs that students got while solving one of the questions. For this question, students needed to fix the code to find the sum of digits of an integer.

```
>>> sum_of_digits(123): 6
Traceback (most recent call last):
  File "./script.py", line 30, in <module>
    sum_of_digits(123)
  File "./script.py", line 18, in sum_of_digits
    total = total + number[index]
TypeError: 'int' object is not subscriptable
```

Figure 1: Example of default EM when trying to find the sum of digits.

```
>>> sum_of_digits(123): 6
Line: 30
    sum_of_digits(123)
Line: 18
    total = total + number[index]
TypeError: 'int' object is not subscriptable
```

Figure 2: Example of simplified EM when trying to find the sum of digits.The Traceback section has been simplified to only contain the line number and the code on that line.

```
>>> sum_of_digits(123): 6
Line: 30
    sum_of_digits(123)
Line: 18
    total = total + number[index]
Are you trying to access a certain value from the an
  'int'? 'int' objects are not like lists where we can
  access a certain character by giving its position in
  brackets.
```

Figure 3: Example of enhanced EM when trying to find the sum of digits. Similar format to the simplified EM version with the inclusion of a conversational explanation of the error type.

- At the conclusion of the session, students were asked to assess their own experience with the app and the debugging tasks in a usability survey and a series of Likert scale questions.

Results and Contributions

- Students who received the simplified EMs, had a higher rate of completion for each of the problems [Figure 4].

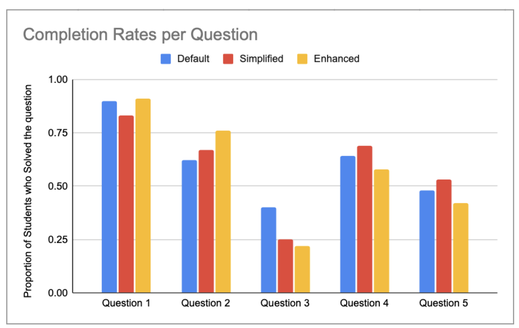

Figure 4: Proportion of students in each treatment group (default EMs, simplified EMs, enhanced EMs) who completed each of the five debugging questions on PyBuggy.

- Of the students who solved the problems, those those who received either of the modified EMs did so in less time [Figure 5]
- Of the students who solved the problems, those who received either of the modified EMs did so in fewer attempts [Figure 6].

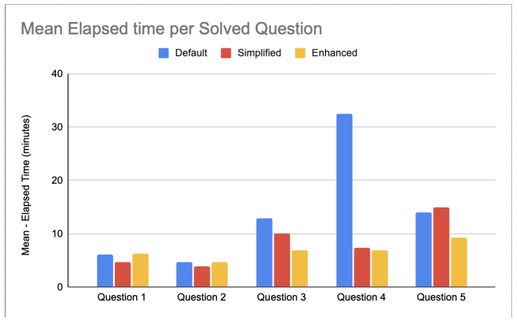

Figure 5: Mean elapsed time, in minutes, that it took students to solve each of the given debugging problems.

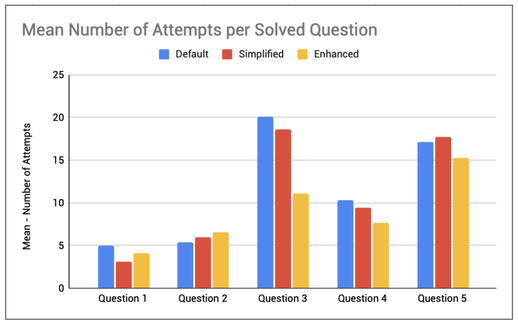

Figure 6: Mean number of attempts that it took students to solve each of the five given debugging problems.

- While these results are not statistically significant, they validate the ability of the tool to monitor completion rate, timing and number of attempts.
- The survey result was inconclusive, with many students finding the *PyBuggy* tool itself awkward, as it is different from the regular text editors, and responding based on the tool rather than the intervention [Figure 7].

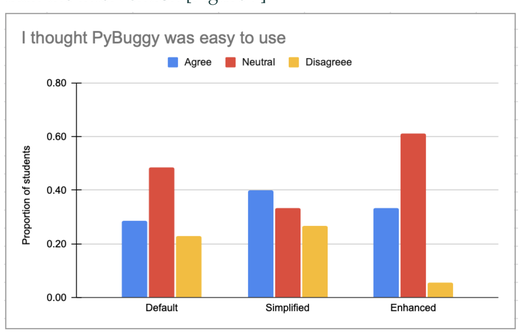

Figure 7: The distribution of student responses to a question on the usability of the PyBuggy app given at the conclusion of the session. Students were asked if they agree or disagree with the statement, "I thought PyBuggy was easy to use" and to what extent.

- We observe mixed results while evaluating students' opinions about the EMs, with some suggestive evidence that students found the enhanced EMs easier to understand than the simplified or default [Figure 8]
- Valuable information was gained through the survey on how to improve *PyBuggy* and the survey tool for the full scale study.

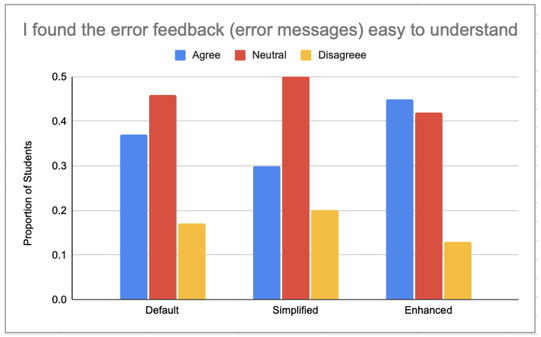

Figure 8: The distribution of student responses to a question on the post-PyBuggy survey regarding the error messages they interacted with during the session. Students were asked if they agree or disagree with the statement, "I found the error feedback (error messages) easy to understand" and to what extent.

Future Work

This study shows that replacing standard technical error messages with natural language enhanced error messages specifically designed for novice programmers is possible. In the future, we hope to incorporate this tool as part of an introduction to programming course to novice programmers on a regular basis. We will also continue to develop and improve our *PyBuggy* tool. We hope to evaluate our tool in a larger setting and more clearly elucidate if these enhanced error messages have an impact on a student's ability to find and fix errors or if they can improve their experience while debugging.

References

[1] Becker, B. A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D. J., Harrington, B., Kamil, A., Karkare, A., McDonald, C., Osera, P.-M., Pearce, J. L., and Prather, J. Compiler error messages considered unhelpful: The landscape of text-based programming error message research. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (2019), ITiCSE-WGR '19, p. 177–210.

[2] Becker, B. A., Goslin, K., and Glanville, G. The effects of enhanced compiler error messages on a syntax error debugging test. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (2018), pp. 640–645.

[3] Denny, P., Luxton-Reilly, A., and Carpenter, D. Enhancing syntax error messages appears ineffectual. In *Proceedings of the 2014 Conference on Innovation amp; Technology in Computer Science Education* (New York, NY, USA, 2014), ITiCSE '14, Association for Computing Machinery, p. 273–278.

[4] D'souza, R., Bhayana, M., Ahmadzadeh, M., and Harrington, B. A mixed-methods study of novice programmer interaction with python error messages. In *Proceedings of the Western Canadian Conference on Computing Education* (2019), WCCCE '19.

[5] Pettit, R. S., Homer, J., and Gee, R. Do enhanced compiler error messages help students? results inconclusive. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (2017), pp. 465–470.