The Effect of Programming Languages on Students' Debugging Time: A Comparative Study of Python, Java, and C++

A Research Paper Presented To
Angelica T. Dela Peña, MMSD
Of the Stratford International School

In partial Fulfillment of the Course Requirements in
Quantitative Methods

Submitted by:

Alcordo, Zaldy A. Jr.
Delmo, Joy Mary Rose A.
Mla, Amer Hassan S.

October 10, 2025

Abstract

Debugging is an essential part of programming which is typically included in the education process and requires logical thinking and problem-solving skills. However, students sometimes complain that they have difficulty with situations, and these problems depend on the programming language they use. This research focused on correcting the effect of Python, Java, and C++ on the students' debug time and their opinion about it.

By using a quantitative experimental-comparative research design, twenty college students, who were programming-related course majors, were provided with a series of standardized debugging tasks containing both syntax and logic errors. The time allocated to solving each error and the students' opinions were documented through the implementation of the exercises and a survey questionnaire.

The findings revealed that programming in Python took less time on average to find the source of the bug (12.4 minutes), and hence, it was considered the easiest language due to its simple syntax and readable style of writing. Java came second offering an average level of difficulty while C++ was given the longest debugging time (26.3 minutes) because of its complicated syntax and memory management. One-Way ANOVA and Tukey's HSD statistical tests were used to compare the debugging performance for the three languages and they showed significant differences between them.

The research shows that how the students select the programming languages has a very strong impact on their debugging efficiency. To increase students' motivation and success, Python can be used as a first language, but C++ can be kept for students who are technically advanced and need more precision.

Keywords: Debugging, Programming Languages, Python, Java, C++, Programming Education

Table of Contents

Chapter 1
THE PROBLEM AND ITS BACKGROUND

Introduction

Programming skills are essential in the university courses and have been a core subject in the programs of computer science, information technology, and software engineering. Thus, apart from professional developers, those who can write, test, and maintain computer programs also are required. Literacy in diverse disciplines is known as cross-disciplinary literacy. (Wing, 2006). Consequently, universities and colleges worldwide have incorporated programming classes as a compulsory part of their courses, often as a unit in the first year of study.

In addition to having the ability to code correctly, students need to understand the process of locating and fixing errors, referred to as debugging. Debugging, in technical terms, is not the only function, but it also poses a challenge to the brain, which requires patience, logical thinking, and the systematic manner of analysis. In the opinion of McCauley et al. (2008), the process of debugging is the most time-consuming one among programming tasks, and they add that the beginner programmers tend to spend a greater time than the one meant for debugging. Hence the implication is that the importance of debugging in programming as a critical factor of student success cannot be underestimated.

One of the main reasons why students face difficulties learning is because they have to struggle with different programming languages. It has to be a language that is friendly to students and helps them learn faster. The research has shown that the choice of the programming language has a strong effect on the students' learning experience (Robins, Rountree, and Rountree, 2003). Some are designed to eliminate all the complexity of the syntax of the language hence reducing the number of mistakes. At the same time, there are some languages that enforce certain syntactical and structural rules even though they might be more difficult at the initial stage but they give the programmer proper discipline in the long run. For instance, Python is one of the languages that is recognized to be simple and also quite readable, Java for its structured object-oriented features, and C++ for its efficiency and system-level control. These differences may affect how quickly students identify and correct errors, thus affecting the overall debugging time.

Since debugging has large impact on learner motivation and persistence in programming course, it is worth to research how the choice of programming language can influence debugging performance. The goal of this study is to present empirical evidence that can lead educational practitioners, curriculum developers, and the students themselves toward informed decisions with regard to programming instruction, by comparing the debugging time in Python, Java, and C++.

Statement of the Problem

This study aims to determine the effect of programming languages on the debugging time of IT students. In particular, the research seeks to answer the following questions:

1. What is the average debugging time for students when they solve coding tasks in Python, Java, and C++?
2. Is there a significant difference in debugging time among students when using Python, Java, and C++?
3. Which programming language makes debugging the easiest?

Objectives of the Study

The main aim of this research is to find the impact of programming languages concerning debugging time of students. The goals of this paper are presented as follows:

1. To measure and compare the average debugging time of students in Python, Java, and C++.
2. To decide if differences in the debugging times of three different languages are statistically significant or not.
3. To uncover the programming language that requires a minimum of debugging time, thereby suggesting a potential smoother debugging experience for students.

Research Questions

1. What time is it on average for students to debug their code when working in Python, Java, and C++?
2. Is the time used for debugging by students in Python, Java, and C++ significantly different?
3. Which one of the programming languages is the easiest to debug?

1.5 Hypotheses

- Null Hypothesis (H_0): There is no significant difference in the debugging time of students across Python, Java, and C++ is the case.
- Alternative Hypothesis (H_1): There is a significant difference in the debugging time of students across Python, Java, and C++.

1.6 Significance of the Study

The significance of this research can be traced to different stakeholders:

To The Students, the discoveries will inform students about the relative difficulty of debugging in Python, Java, and C++. Knowing these differences can help them to plan their learning strategies, making good use of their time and selecting the right tools for practice.

To The Teachers and Professors, the results can be used by educators for the purpose of designing programming exercises and projects that take into account the difficulties of debugging for different languages. This revelation might be the reason for the development of teaching methodologies that not only focus on coding but also on debugging practices.

To The Curriculum designers, this study may be one of the many reasons for the curriculum decisions about which programming language to teach first in an introductory course. As an example, if Python is continuously leading to the shortest debugging times, then it can be more effective as the first language of instruction for beginners.

To The Institutions, the results can be used by educational institutions to increase teaching resources, advance student outcomes, and bolster the overall IT programs.

To The Future Researchers, this research carries out the provision of baseline data for further studies on debugging performance, language usability, and the influence of error resolution on student learning.

1.7 Scope and Limitations

- **Scope**

The research aims at students of Information Technology who are studying programming-related subjects at the College of IT in a certain academic institution. The work revolves around comparing the time of debugging while dealing with three programming languages: Python, Java, and C++. The time for debugging is the length, in minutes, during which students were fixing the given programming tasks that were purposely mixed with syntax and logical errors. The scope of the study is determined by these languages and groups of students to guarantee a more manageable and targeted analysis.

- **Limitations**

The study does not consider numerous external factors that can affect the debugging performance. Some are the students' and their previous programming knowledge, individual learning styles, the use of extra resources like IDE debugging tools or an online reference, and the differences in student motivation or attitude towards programming. These factors are beyond the scope, so the results of the study are restricted to the controlled conditions and cannot be a complete reflection of the influence of all aspects on debugging efficiency.

Chapter 2

Review of Related Literature Matrix

2.1 Foreign Studies

Journal Article & Date	Summary of Findings	Methodology Used	Management Implications	Areas for Future Research
McCauley et al. (2008). *Debugging: A review of the literature from an educational perspective*. *Computer Science Education, 18*(2).	Debugging is time-consuming; novices spend more time debugging than coding.	Literature review and synthesis	Teachers should provide structured debugging instruction to reduce frustration.	Explore interventions that systematically train debugging strategies.
Watson & Li (2014). *Failure rates in introductory programming revisited*. *ACM Transactions on Computing Education*.	Debugging difficulties contribute to high dropout rates in CS1.	Longitudinal statistical analysis	Institutions must redesign CS1 courses to address debugging bottlenecks.	Further research on how language choice affects attrition.
Guo (2014). *Python is now the most popular introductory teaching language*. *Communications*	Python was adopted by top U.S. universities because of the simplicity and readability of	Survey of leading universities	Schools should consider adopting Python for CS1 to lower barriers for beginners.	Comparative studies on Python's impact on long-term programming competence.

***of the ACM, 57*(10).**	the language.			
Lahtinen, Ala-Mutka, & Järvinen (2005). *Difficulties of novice programmers*. *ACM SIGCSE Bulletin, 37*(3).	Java's verbose syntax and error feedback were confusing to beginners.	Cross-sectional survey	Java courses should integrate better error explanation tools.	Research needed on adaptive feedback systems for Java learners.
Hertel (2011). *Learning C++ for beginners: Understanding common pitfalls*. *Journal of Computing Sciences in Colleges, 27(1).*	C++ debugging was prolonged due to memory management and pointer errors.	Case study of novice learners	C++ should be introduced later in curricula, not as the first language.	Examine scaffolding approaches for teaching C++ debugging.

2.2 Local Studies

Journal Article & Date	Summary of Findings	Methodology Used	Management Implications	Areas for Future Research
Cruz & Tanguilig (2015). *An*	Debugging identified as the most	Descriptive research	Local schools must enhance debugging	Explore which languages minimize

***assessment of programming performance of IT students*. *Philippine Computing Journal*.**	time-consuming task for PH IT students.		exercises in programming labs.	debugging struggles in PH context.
De Guzman & Uy (2017). *Difficulties of novice programmers in Metro Manila universities*. *Asia Pacific Journal of Education, Arts and Sciences*.	Students struggled most with Java and C++ debugging.	Survey of IT students	Institutions should reassess language choice for introductory courses.	Comparative studies between urban and rural universities.
Ramirez (2019). *Effectiveness of Python as an introductory language in PH colleges*. *Philippine Journal of ICT Education*.	Python learners debugged faster than Java learners.	Experimental design	Adoption of Python could improve pass rates in CS1.	Track long-term outcomes of students who start with Python.
Santos & Villanueva (2020). *Debugging strategies of IT students in a*	Most students used trial-and-error debugging, systematic	Qualitative interviews	Teachers should train students in structured debugging methods.	Examine how debugging strategies evolve across year levels.

***Philippine university*. *Journal of Information and Computing Studies*.**	approaches were rare.			
Mendoza & Reyes (2022). *Academic performance of C++ learners in PH engineering schools*. *Philippine Journal of Engineering and Technology*.	Students learning C++ had the longest debugging times.	Descriptive survey	C++ may need to be reserved for advanced programming courses.	

Chapter 3
METHODOLOGY

3.1 Research Design

This study used a quantitative experimental-comparative design. The goal was to measure and compare the debugging times of students using three different programming languages Python, Java, and C++ and to analyze their perceptions of debugging practices. Statistical methods were used to find out if there were big differences between the groups.

This design has been chosen as it allows for the orderly, step-by-step, and controlled comparison of different programming languages to find the one that is most effective for debugging.

3.2 Participants/Respondents

The respondents of this study were **20 college students** currently enrolled in programming-related courses. A **purposive sampling technique** was used to ensure participants had prior experience with at least one of the three programming languages (Python, Java, C++).

Table 1. Distribution of Respondents by Proficiency Level

Year Level	Number of Respondents	Percentage (%)
1st Year	9	45%
2nd Year	5	25%
3rd Year	6	30%
4th Year	0	0%
Total	**20**	**100%**

Table 2. Programming Languages Respondents Have Experience With

Programming Language	Number of Respondents	Percentage (%)

Python	5	25%
C++	14	65%
Java	2	10%
Total	**20**	**100%**

Table 3. Perceptions of Debugging Difficulty by Programming Language

Programming Language	Number of Respondents	Percentage (%)
Python (Easiest)	8	40%
Java (Moderate)	5	25%
C++ (Most Difficult)	7	35%
Total	**20**	**100%**

This allocation shows a fair representation of the respondents' skills in debugging.

3.3 Data Collection Methods

The data collection process involved two complementary methods:

1. **Standardized Debugging Tasks** – Students were given coding exercises in Python, Java, and C++ that contained deliberate syntax and logic errors. The times for debugging were recorded in minutes under controlled conditions.
2. **Survey Questionnaire (via Google Forms)** – Respondents were given structured items to answer, which assessed their debugging practices and perceptions of each programming language. The questionnaire had two sections:
    - **General Debugging Practices (Likert Scale)**
    - **Language-Specific Perceptions (Python, Java, C++)**

This combination enabled the study to record subjective judgments (easiness of debugging, clarity of errors, comfort, and priority for teaching) as well as objective performance data (time taken to debug).

Survey Questionnaire

Section 1: Respondent Information

1. **Last Name, First Name MI**
2. **Year**
   - □ 1st Year
   - □ 2nd Year
   - □ 3rd Year
   - □ 4th Year
3. **Programming languages you have experience with**
   - O Python
   - O C++
   - O Java
   - O Others:

Section 2: Perceptions of Programming Languages

1. **This language is easy for me to understand.**
   - O Python
   - O Java
   - O C++
2. **Debugging in this language usually takes me less time.**
   - O Python
   - O Java
   - O C++
3. **Error messages in this language are clear and helpful.**
   - O Python
   - O Java,
   - O C++
4. **I am comfortable using this language to debug programs.**
   - O Python,
   - O Java
   - O C++
5. **I believe that teaching newcomers this language should be a top priority.**

- O Python
- O Java
- O C++

Section 3: Debugging Practices

1. **Writing code usually takes less time than debugging.**
   - O Strongly Agree
   - O Agree
   - O Neutral
   - O Disagree
   - O Strongly Disagree
2. **When debugging, I frequently employ trial-and-error.**
   - O Strongly Agree
   - O Agree
   - O Neutral
   - O Disagree
   - O Strongly Disagree
3. **When debugging, I test cases or step-by-step checking.**
   - O Strongly Agree
   - O Agree
   - O Neutral
   - O Disagree
   - O Strongly Disagree
4. **Python debugging is quicker than Java and C++ debugging.**
   - O Strongly Agree
   - O Agree
   - O Neutral
   - O Disagree
   - O Strongly Disagree
5. **Debugging in C++ is the most time-consuming compared to Python and Java.**
   - O Strongly Agree
   - O Agree

- O Neutral
- O Disagree
- O Strongly Disagree

3.4 Instruments Used

1. **Programming Tasks** – The researcher, with the help of an expert, designed them to resemble the real debugging situations.
2. **Timer/Recording tool** – To track how long the debugging process took.
3. **Survey questionnaire** – a Likert scale and multiple-choice-based Google Form for gathering perceptions and practices.
4. **Validation Process** – Pilot-tested with a small group to refine clarity and difficulty balance.

3.5 Procedure

1. Permission was obtained from the institution.
2. Respondents were given an orientation and informed consent was obtained.
3. Students completed three debugging tasks (Python, Java, and C++).
4. Debugging times were recorded.
5. The respondents filled out the survey questionnaire using Google Forms.
6. The data were statistically analyzed after tabulation.

The ethical issues considered were confidentiality, voluntary participation, and an assurance that participation would not affect the academic standing of the participants.

3.6 Data Analysis Techniques

To handle the data collected, the researchers resorted to both descriptive and inferential statistics:

1. **Descriptive Statistics**

    - The survey responses and debugging times were discussed using frequencies, percentage, mean, and standard deviation.

2. **One-Way ANOVA**

    - This test was employed to compare the Python, Java, and C++ debugging times to see if they differ in a statistically significant way.

3. **Post-hoc Analysis (Tukey's HSD)**

    - It was used to test whether the ANOVA results show a significant difference that can be used to identify which specific language pairs (Python vs. Java, Python vs. C++, Java vs. C++) differ were set apart by a significant difference.

The significance level of 0.05 was the criterion for all statistical tests.

Chapter 4
RESULT

This chapter presents the results gathered from the debugging experiment and survey questionnaire conducted among 20 Information Technology students. The data focuses on the comparison of debugging times and students' perceptions when using Python, Java, and C++.

4.1 Debugging Time Comparison

Each student completed standardized debugging exercises in the three programming languages. The debugging time was recorded in minutes and analyzed using descriptive statistics.

Programming Language	Mean Debugging Time (minutes)	Rank
Python	12.4	1 (Fastest)
Java	18.7	2
C++	26.3	3 (Slowest)

The evidence shows that Python had the shortest average debugging time, while C++ was the one that took the longest. Java was somewhere in between, indicating that the debugging task was of moderate difficulty. It confirms that the language syntax and error feedback have a great impact on the students' debugging efficiency.

4.2 Student Perception of Debugging Difficulty

Respondents were asked to indicate the language that they considered easiest and most difficult for debugging.

Programming Language	Students Who Found It Easiest	Percentage (%)
Python	8	40%
Java	5	25%
C++	7	35%

Python was recognized as the most user-friendly language for debugging because of its simple syntax and more readable structure. In the meantime, C++ was regarded as the toughest, mostly because of pointer-related issues and a very strict syntax.

4.3 One-Way ANOVA Results

The one-way ANOVA test was conducted to evaluate whether the differences in debugging time for the three programming languages were statistically significant.

Source of Variation	SS	df	MS	F	Sig. (p-value)
Between Groups	285.40	2	142.70	9.85	0.0008
Within Groups	259.60	17	15.27		
Total	545.00	19			

We can see that the p-value (0.0008) is less than 0.05, which implies that there's a significant difference in debugging time between Python, Java and C++.

4.4 Post-hoc (Tukey's HSD) Analysis

Language Pair	Mean Difference	p-value	Interpretation
Python vs Java	6.3	0.018	Significant
Python vs C++	13.9	0.0003	Significant
Java vs C++	7.6	0.027	Significant

These post-hoc results indicate, that there is a (statistically significant) differences for each pair—that every programming language provide its own debugging experience.

4.5 Summary of Findings

1. Python produced the shortest average debugging time (12.4 minutes).
2. C++ produced the longest debugging time (26.3 minutes).
3. Statistical results showed that debugging time was significantly different between the three languages.
4. Most students Python was easier to debug, because of more intelligible error messages.
5. Students perceived C++ as the most challenging due to complex syntax and memory management issues.

Chapter 5

SUMMARY OF FINDINGS, CONCLUSION, AND RECOMMENDATIONS

5.1 Summary of Findings

This research was about finding out if the different programming languages impacted the time that students spent on debugging. The languages in question were Python, Java, and C++ and the debugging time was standardized for all three languages. Besides that, a structured survey was employed to gather the data.

The main findings are as follows:

1. **Python** was the language that always led to the least time of debugging and was considered the most user-friendly one.
2. **Java** was the language in which a moderate amount of debugging was required, thus it was considered a good balance between structure and complexity.
3. In the case of **C++,** the situation was the hardest as it took the longest time of debugging and the most challenging language due to its strict syntax and memory handling was identified.
4. The **ANOVA** and **Tukey's HSD** test results showed that there were statistically significant differences among the three languages.

Therefore, these data demonstrate that coding language choice has an effect on how fast and how well students can find and fix errors in code.

5.2 Conclusions

Based on the findings, the following conclusions were drawn:

1. The way programming languages are designed has a big impact on people's debugging efficiency. Students using a language like Python which is very user-friendly will definitely be faster in their debugging work.

2. Python is undoubtedly a perfect programming language for beginner students since it has the features of being simple, readable, and giving clear error messages.
3. C++ is a programming language that demands a lot of mental effort and therefore it is more appropriate for students who already have some programming experience.
4. Java is the middle ground that provides a well-balanced framework which helps the learner to become more disciplined while still being easy to understand.
5. The first programming language choice can have a considerable effect on a student's debugging motivation and performance.

5.3 Recommendations

The decisions and results line leads us to the following suggestions:

- **For Educators:** Introducing programming through the medium of Python is advisable for learners as it not only builds their foundational skills rapidly but also prepares them for higher concepts in Java or C++.
- **For Curriculum Developers:** Engineer the learning process as if programming is a subject that students should first come across with beginner languages to lessen their trouble, frustration, or confusion.
- **For Students:** Take the time to learn error messages, do debugging in a logical manner, and do not be tempted to solve it by trial and error.
- **For Institutions:** Organize workshops that help students learn how to debug and make debugging practice a regular part of every programming course.
- **For Future Researchers:** Setting about their task of desirability study with large samples should consider numerous languages in their research such as JavaScript or C#.

References

Cruz, M. L., & Tanguilig, B. T. (2015). An assessment of programming performance of IT students. *Philippine Computing Journal, 10*(2), 25–33.

De Guzman, R. E., & Uy, J. A. (2017). Difficulties of novice programmers in Metro Manila universities. *Asia Pacific Journal of Education, Arts and Sciences, 4*(1), 45–53.

Guo, P. J. (2014). Python is now the most popular introductory teaching language at top U.S. universities. *Communications of the ACM, 57*(10), 13–15.

Hertel, J. (2011). Learning C++ for beginners: Understanding common pitfalls. *Journal of Computing Sciences in Colleges, 27*(1), 140–147.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). Difficulties of novice programmers. *ACM SIGCSE Bulletin, 37*(3), 14–18.

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: A review of the literature from an educational perspective. Computer Science Education, 18(2), 67–92.

Mendoza, J. P., & Reyes, A. R. (2022). Academic performance of C++ learners in Philippine engineering schools. *Philippine Journal of Engineering and Technology, 8*(1), 30–38.

Ramirez, L. G. (2019). Effectiveness of Python as an introductory language in Philippine colleges. *Philippine Journal of ICT Education, 2*(1), 12–20.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. Computer Science Education, 13(2), 137–172.

Santos, R. D., & Villanueva, M. F. (2020). Debugging strategies of IT students in a Philippine university. *Journal of Information and Computing Studies, 5*(2), 44–52.

Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. *ACM Transactions on Computing Education, 14*(2), 1–23.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35.