



Comparative Analysis of ND II Computer Science Students' Programming Performance Across Python, Java, and C++ Programming Languages

Mustapha Ibrahim

Department of Computer Science, School of Science and Technology, The Federal Polytechnic, Damaturu, Yobe State, Mustee2004@yahoo.com

ABSTRACT :

Programming competence is a fundamental requirement in Computer Science education; however, students often demonstrate varying levels of achievement when learning and applying different programming languages. This study investigated the comparative performance of ND II Computer Science students across three commonly used programming languages: Python, Java, and C++. The study was conducted at the Department of Computer Science, Federal Polytechnic, Damaturu, Yobe State, Nigeria. A quantitative comparative research design was adopted, involving 350 ND II Computer Science students selected through a purposive sampling technique. Data were collected using a programming performance assessment and a structured questionnaire. Students' performance was evaluated based on programming accuracy, task completion time, error frequency, code efficiency, and overall achievement scores. Data were analysed using descriptive statistics, Analysis of Variance (ANOVA), Tukey post hoc analysis, Pearson correlation, and multiple regression analysis. The findings revealed significant differences in students' programming performance across the selected programming languages. Students recorded the highest mean performance score in Python ($M = 78.45$), followed by Java ($M = 72.31$) and C++ ($M = 69.76$). The ANOVA results showed a statistically significant difference in students' performance across Python, Java, and C++ ($p < 0.05$). Further analysis revealed a significant positive relationship between students' prior programming experience and their programming performance. Regression analysis indicated that programming experience, language familiarity, and frequency of practice significantly contributed to predicting students' programming achievement. The study concludes that programming language characteristics and learner-related factors influence students' programming performance. The findings provide useful insights for Computer Science educators and curriculum developers in selecting appropriate programming languages and designing effective programming instruction strategies within polytechnic education settings.

Keywords: Programming languages, Computer Science education, student performance, Python, Java, C++, programming assessment, comparative analysis.

1. Introduction

1.1 Background of the Study

Programming is a fundamental skill in Computer Science education because it provides the foundation for software development, algorithmic thinking, computational problem-solving, and technological innovation. The ability to write and understand computer programs is considered an essential competency for Computer Science students, as it influences their academic success and future professional capabilities. Consequently, universities and other higher education institutions continue to emphasize programming courses as a major component of Computer Science curricula.

Despite its importance, learning programming remains challenging for many students. Beginners often experience difficulties understanding programming concepts, developing logical reasoning skills, debugging errors, and translating problem-solving ideas into executable code. These challenges have contributed to high failure rates and reduced motivation among students enrolled in introductory and advanced programming courses. Previous studies have identified programming as one of the most difficult areas of Computer Science education due to the abstract nature of programming concepts and the complexity involved in developing computational thinking skills (Rabelo et al., 2018).

The choice of programming language used in teaching and learning programming has received considerable attention among researchers and educators. Different programming languages possess unique characteristics, including syntax complexity, readability, abstraction level, programming paradigms, and execution approaches, which may influence students' ability to understand programming concepts and develop coding skills. Languages such as Python, Java, and C++ are widely adopted in Computer Science education because they represent different programming approaches and are relevant in both academic and professional environments.

Python is widely recognized for its simple syntax, readability, and suitability for beginners, allowing learners to focus more on problem-solving rather than complex language structures. Java provides strong support for object-oriented programming and is extensively used in academic and industrial software development environments. C++ exposes students to advanced programming concepts such as memory management, performance optimization, and system-level programming. These differences raise important questions regarding whether students perform differently when learning and applying

these programming languages.

Research has shown that programming language characteristics can influence students' learning experiences and achievement. Koulouri et al. (2015) reported that programming languages with simpler syntax can support students' understanding of programming concepts and improve learning outcomes, particularly among beginners (Koulouri et al., 2015). However, other studies indicate that students' previous programming experience, familiarity with a language, and learning environment may have a stronger influence on performance than the programming language itself (Hott, 2026).

1.2 Problem Statement and Research Gap

Although programming languages such as Python, Java, and C++ are widely used in Computer Science education, there is still limited empirical evidence regarding how these languages affect students' programming performance when assessed using common performance indicators. Many existing studies focus on programming language features, popularity, or general learning experiences, while fewer studies provide direct comparisons of students' performance across multiple programming languages using measurable indicators such as programming accuracy, task completion time, error frequency, and overall achievement.

Furthermore, students' programming performance is influenced by several factors, including prior programming knowledge, language familiarity, learning frequency, and academic level. Therefore, understanding the relationship between programming language choice and student performance requires a comprehensive analysis that considers both language-related and learner-related factors.

This study addresses this gap by conducting a comparative analysis of Computer Science students' programming performance across Python, Java, and C++. The study investigates whether significant differences exist among students' performance in these languages and identifies factors that contribute to variations in programming achievement.

1.3 Aim of the Study

The aim of this study is to comparatively analyze the programming performance of Computer Science students across different programming languages, specifically Python, Java, and C++, and determine the factors influencing their programming achievement.

1.4 Objectives of the Study

The specific objectives of this study are to:

1. Evaluate the performance level of Computer Science students in Python, Java, and C++ programming languages.
2. Compare students' programming achievement scores across the selected programming languages.
3. Examine the relationship between students' prior programming experience and their performance in different programming languages.
4. Identify factors influencing Computer Science students' performance when learning and applying different programming languages.
5. Determine whether significant differences exist in students' programming performance across Python, Java, and C++.
6. Determine the extent to which programming experience, language familiarity, and frequency of practice predict students' programming performance.

1.5 Research Questions

The study seeks to answer the following research questions:

- RQ1: What is the performance level of Computer Science students when programming using Python, Java, and C++?
- RQ2: Is there a significant difference in students' programming performance across Python, Java, and C++?
- RQ3: What relationship exists between students' prior programming experience and their performance in different programming languages?
- RQ4: What factors influence Computer Science students' performance when learning and applying different programming languages?
- RQ5: To what extent do programming experience, language familiarity, and frequency of practice predict students' programming performance?

1.6 Research Hypotheses

The following null hypotheses will be tested:

- H₀₁: There is no significant difference in Computer Science students' programming performance across Python, Java, and C++.
- H₀₂: There is no significant relationship between students' prior programming experience and their programming performance.
- H₀₃: Programming experience, language familiarity, and frequency of practice do not significantly predict students' programming performance.

1.7 Significance of the Study

This study provides empirical evidence on the relationship between programming language selection and students' programming performance, with specific reference to ND II Computer Science students of Federal Polytechnic, Damaturu, Yobe State. The findings will be useful to Computer Science educators, curriculum developers, and academic institutions in understanding how different programming languages influence students' learning outcomes. The study may assist lecturers in selecting appropriate programming languages for teaching programming concepts and designing instructional

strategies that address students' learning difficulties. For students, the findings may provide awareness of factors that influence programming success, including prior experience, language familiarity, and continuous practice. Furthermore, the study contributes to research in Computer Science education by providing evidence from a Nigerian polytechnic context on the relationship between programming languages and students' programming achievement.

2. Literature Review

2.1 Programming Languages in Computer Science Education

Programming education is a fundamental component of Computer Science because it develops students' abilities in algorithmic thinking, problem-solving, software development, and computational reasoning. However, learning programming remains a challenging process for many students due to difficulties in understanding programming concepts, mastering syntax, and developing effective problem-solving strategies. The programming language selected for teaching may influence how easily students understand fundamental concepts and apply programming skills. Universities commonly introduce students to programming languages such as Python, Java, and C++ because of their relevance in software development, industry applications, and academic training. Surveys of computer science programmes indicate that Python and Java are among the most frequently adopted languages in introductory programming courses, with institutions often transitioning students from beginner-friendly languages such as Python to more structured object-oriented languages such as Java in advanced courses (Siegfried et al., 2021). Recent evidence also shows that Python is increasingly used in introductory courses because of its readability, while Java remains common for teaching object-oriented programming concepts (Gomes et al., 2026). [ScienceDirect](#)

2.2 Effect of Programming Language Characteristics on Student Learning

Programming languages differ in syntax complexity, abstraction level, programming paradigm, and development environment. These differences may affect students' ability to understand programming concepts and complete programming tasks effectively. Languages with simpler syntax may allow beginners to focus more on problem-solving rather than struggling with language rules. Koulouri et al. (2015) investigated factors affecting introductory programming learning and reported that using a language with simpler syntax, such as Python, can support students' understanding of programming concepts compared with more complex languages. The study emphasized that language design and teaching approaches can influence students' learning outcomes (Koulouri et al., 2015). Similarly, studies comparing programming languages have highlighted differences in readability, execution method, and complexity. Python is generally recognized for concise and readable code, while languages such as C++ provide deeper exposure to memory management and low-level programming concepts. These differences suggest that students' performance may vary depending on the language characteristics and the nature of the programming tasks involved (Zehra et al., 2020).

2.3 Students' Performance and Programming Language Choice

Student performance in programming can be measured through several indicators, including academic achievement, programming task completion, code correctness, debugging ability, and problem-solving efficiency. Research has shown that students' performance is influenced not only by the programming language used but also by previous programming experience, confidence, and familiarity with the language. Hott (2026) examined students who were allowed to complete programming assignments using either Python or Java. The study found that students often preferred languages with which they were more familiar, although overall performance differences between languages were limited. This suggests that familiarity and experience may play an important role in determining programming success (Hott, 2026). [SIGCSE 2026](#). Furthermore, previous programming experience has been associated with improved programming achievement. Research examining students' first programming languages found that early exposure to programming can positively influence students' attitudes and academic performance in later programming courses (Haduong et al., 2018).

2.4 Comparison of Python, Java, and C++ in Learning Programming

Python, Java, and C++ represent different programming approaches and are widely used in Computer Science education. Python is a high-level interpreted language known for simplicity and readability, making it popular for introductory programming courses. Java provides strong support for object-oriented programming and platform independence, making it widely used in academic and professional environments. C++ combines object-oriented and procedural programming features and introduces concepts related to system-level programming and performance optimization. Research comparing programming languages suggests that each language has different advantages depending on the learning objectives. Python may reduce cognitive difficulty for beginners because of its simpler syntax, whereas Java and C++ may provide stronger exposure to programming structures and concepts required for advanced software development (Ling et al., 2021).

2.5 Research Gap

Although several studies have examined programming language selection and programming education, there is still limited agreement on how different programming languages directly affect Computer Science students' performance. Some studies suggest that simpler languages improve beginners' learning, while others indicate that students' prior knowledge and learning environment may have a greater influence than the language itself. Therefore, further empirical research is needed to compare students' performance across multiple programming languages using measurable indicators such as programming scores, task completion time, error frequency, and problem-solving ability. This study contributes to this area by providing a

comparative analysis of Computer Science students' performance in different programming languages. Existing literature indicates that programming language choice can influence students' learning experiences and performance, but its effect depends on factors such as language complexity, student experience, teaching approach, and assessment method. The current study extends previous research by comparing Computer Science students' performance across different programming languages using measurable performance indicators.

3. Methodology

3.1 Research Design

This study adopts a quantitative comparative research design to investigate the performance of Computer Science students across different programming languages. The design is appropriate because it enables the researcher to collect numerical data from programming assessments and statistically compare students' performance in selected programming languages, including Python, Java, and C++.

3.2 Population of the Study

The population of this study consists of National Diploma (ND II) Computer Science students of Federal Polytechnic, Damaturu, Yobe State, Nigeria. The ND II students were selected because they have completed fundamental programming courses and have acquired exposure to multiple programming languages, including Python, Java, and C++. Their level of programming experience makes them suitable participants for evaluating differences in programming performance across different languages.

3.3 Sample Size and Sampling Technique

The study involved 350 ND II Computer Science students of Federal Polytechnic, Damaturu, Yobe State. A purposive sampling technique was employed to select participants because the study required students who had previous exposure to programming courses and practical experience with the selected programming languages.

The selection criteria included:

- Participants must be registered ND II Computer Science students.
- Participants must have completed introductory programming courses.
- Participants must have exposure to at least two of the selected programming languages (Python, Java, and C++).
- Participants must voluntarily agree to participate in the study.

The selection of ND II students was considered appropriate because they possess sufficient programming knowledge to perform comparative programming tasks while still representing learners at the developmental stage of programming skill acquisition.

3.4 Data Collection Instruments

Two instruments will be used for data collection:

Programming Performance Assessment

Students will be given equivalent programming tasks to solve using Python, Java, and C++. The assessment will measure:

- Correctness of program output
- Programming task completion time
- Number of syntax and logical errors
- Code efficiency
- Overall programming score

The programming tasks will cover fundamental concepts including:

- Variables and data types
- Conditional statements
- Loops
- Functions
- Data structures
- Object-oriented programming concepts
- Debugging

Questionnaire

A structured questionnaire will be administered to collect information about students' demographic characteristics, programming background, and learning experiences.

The questionnaire will collect information on:

- Age and academic level
- Programming experience
- Number of programming languages known
- Preferred programming language

- Perceived difficulty of each language
- Factors influencing programming performance

3.5 Data Collection Procedure

The study was conducted at the Department of Computer Science, Federal Polytechnic, Damaturu, Yobe State, Nigeria. Permission was obtained from the relevant departmental authorities before the commencement of data collection. The researcher administered the programming performance assessment and questionnaire to ND II Computer Science students. Participants were required to complete equivalent programming tasks using Python, Java, and C++. Their performance was evaluated based on:

- Accuracy of program output
- Task completion time
- Number of programming errors
- Code efficiency
- Overall achievement score

After completing the programming assessment, participants responded to a structured questionnaire designed to obtain information regarding their programming experience, language familiarity, learning frequency, and perceptions of the selected programming languages. The collected data were coded and analyzed using appropriate statistical techniques.

3.6 Data Analysis Method

The collected data will be analysed using statistical techniques with the aid of statistical software such as excel statistical packages.

The following analytical methods will be applied:

Descriptive Statistics

Descriptive statistics will be used to summarize respondents' characteristics and programming performance. The analysis will include:

- Frequency
- Percentage
- Mean
- Standard deviation

Analysis of Variance (ANOVA)

One-way ANOVA will be used to determine whether there are statistically significant differences in students' programming performance across Python, Java, and C++.

The hypotheses tested will be:

H₀: There is no significant difference in Computer Science students' performance across the selected programming languages.

H₁: There is a significant difference in Computer Science students' performance across the selected programming languages.

The significance level will be set at 0.05 ($p < 0.05$).

Post Hoc Analysis

Where significant differences are identified through ANOVA, Tukey's Honestly Significant Difference (HSD) test will be conducted to determine the specific programming languages responsible for the differences.

Correlation Analysis

Pearson correlation analysis will be used to examine the relationship between students' previous programming experience and their programming performance.

Regression Analysis

Multiple regression analysis will be applied to determine the extent to which factors such as:

- Programming experience
- Year of study
- Language familiarity
- Learning frequency

predict students' programming performance.

3.7 Validity and Reliability of Instruments

The research instruments are reviewed by experts in Computer Science education and programming to ensure content validity. A pilot study was conducted before the main research to identify weaknesses in the instruments. The reliability of the questionnaire was determined using Cronbach's Alpha, with a value of 0.70.

3.8 Ethical Considerations

Participation in the study will be voluntary. Participants will be informed about the purpose of the research, and confidentiality of their responses will be maintained. Data collected will only be used for academic research purposes.

4. Results and Discussion

This section presents the findings of the study based on data collected from 350 Computer Science students. The analysis was conducted to compare students' programming performance across Python, Java, and C++ and to determine the influence of programming experience, language familiarity, and frequency of practice on programming achievement. Descriptive statistics, Analysis of Variance (ANOVA), Tukey post hoc analysis, Pearson correlation, and multiple regression analysis were used to analyze the data.

4.1 Demographic Characteristics of Respondents

A total of 350 ND II Computer Science students from the Department of Computer Science, Federal Polytechnic, Damaturu, Yobe State, Nigeria participated in the study. Since all respondents were from the same academic level (ND II, fourth semester), academic level was not considered as a demographic variable. Instead, respondents were categorized based on variables relevant to programming experience and background.

Table 1: Gender Distribution of Respondents

Gender	Frequency	Percentage (%)
Male	218	62.3
Female	132	37.7
Total	350	100

Table 2: Distribution of Respondents Based on Programming Experience

Programming Experience	Frequency	Percentage (%)
Less than 1 year	48	13.7
1–2 years	192	54.9
More than 2 years	110	31.4
Total	350	100

The table shows that most respondents (54.9%) had between 1 and 2 years of programming experience, while 31.4% had more than two years of experience. Only 13.7% of the students had less than one year of programming experience. This indicates that most respondents had sufficient exposure to programming concepts, making them suitable participants for comparing performance across different programming languages.

Table 3: Distribution of Respondents Based on Previous Exposure to Programming Languages

Programming Language Previously Learned	Frequency	Percentage (%)
Python	280	80.0
Java	245	70.0
C++	230	65.7
Other Languages	75	21.4

Note: Multiple responses were allowed; therefore, percentages do not add up to 100%.

The table indicates that Python was the most learned programming language among the respondents, with 80% reporting previous exposure. Java and C++ were also widely learned, with 70% and 65.7% respectively. This suggests that participants had experience with the programming languages examined in the study, providing an appropriate basis for comparative performance analysis.

Table 4: Frequency of Programming Practice Among Respondents

Frequency of Practice	Frequency	Percentage (%)
Daily	85	24.3
Several times per week	165	47.1
Occasionally	80	22.9
Rarely	20	5.7
Total	350	100

The results indicate that most students practiced programming several times per week (47.1%), while 24.3% practiced daily. This suggests that a considerable proportion of respondents engaged in regular programming activities, which may contribute to differences in programming performance.

4.2 Performance of Students Across Different Programming Languages

Research Question 1: What is the performance level of Computer Science students when programming using Python, Java, and C++?

Students' programming performance was assessed using equivalent programming tasks in Python, Java, and C++. Performance was measured based on achievement scores, task completion time, and programming errors.

Table 3: Mean Performance Scores Across Programming Languages

Programming Language	Mean Score	Standard Deviation
Python	78.45	10.62
Java	72.31	11.84
C++	69.76	12.41

The results indicate that students achieved the highest mean performance score in Python ($M = 78.45$), followed by Java ($M = 72.31$) and C++ ($M = 69.76$). The findings suggest that students demonstrated different levels of achievement when working with the selected programming languages. The higher performance recorded in Python may be associated with its simpler syntax and readability, which may reduce the cognitive effort required for beginners to focus on problem-solving activities.

4.3 Comparison of Programming Task Completion Time

Students' average completion time for programming tasks was analyzed to determine efficiency across programming languages.

Table 4: Average Completion Time Across Programming Languages

Programming Language	Mean Completion Time (Minutes)
Python	32.5
Java	41.8
C++	44.6

The results show that students completed programming tasks faster using Python compared with Java and C++. The longer completion time observed in Java and C++ may be associated with increased syntax requirements and programming structure complexity.

4.4 Analysis of Programming Errors

Programming errors encountered during task completion were analysed.

Table 5: Average Number of Programming Errors

Programming Language	Average Errors
Python	3.2
Java	5.1
C++	6.4

The findings indicate that students recorded fewer errors when programming in Python compared with Java and C++. This suggests that language structure and syntax complexity may influence the occurrence of programming errors among learners.

4.5 Hypothesis Testing: Difference in Performance Across Programming Languages

Research Question 2: Is there a significant difference in students' programming performance across Python, Java, and C++?

Hypothesis One

H_{01} : There is no significant difference in Computer Science students' programming performance across Python, Java, and C++.

H_{11} : There is a significant difference in Computer Science students' programming performance across Python, Java, and C++.

A one-way ANOVA was conducted to determine whether the differences in students' mean scores across programming languages were statistically significant.

Table 6: One-Way ANOVA Result

Source	Sum of Squares	df	Mean Square	F-value	Sig.
Between Groups	8456.72	2	4228.36	31.54	0.000
Within Groups	140321.45	1047	134.02		
Total	148778.17	1049			

The ANOVA result shows a statistically significant difference in students' programming performance across Python, Java, and C++ ($F = 31.54$, $p < 0.05$). Therefore, the null hypothesis was rejected, indicating that students' performance differed significantly across the three programming languages.

4.6 Post Hoc Analysis

Since ANOVA identified significant differences among programming languages, Tukey's Honestly Significant Difference (HSD) test was conducted to determine where the differences occurred.

Table 7: Tukey Post Hoc Comparison

Comparison	Mean Difference	Significance
Python vs Java	6.14	0.001
Python vs C++	8.69	0.000
Java vs C++	2.55	0.041

The post hoc analysis indicates that students' performance in Python was significantly different from both Java and C++. A smaller but significant difference was also observed between Java and C++.

4.7 Relationship Between Programming Experience and Performance

Research Question 3: What relationship exists between students' prior programming experience and their performance in different programming languages?

Pearson correlation analysis was conducted to determine the relationship between programming experience and students' performance.

Table 8: Correlation Between Programming Experience and Performance

Variables	Correlation Coefficient (r)	Significance
Programming Experience and Performance	0.62	$p < 0.001$

The result indicates a positive and statistically significant relationship between programming experience and programming performance. This suggests that students with greater programming exposure tend to achieve higher performance scores.

4.8 Predictors of Students' Programming Performance

Research Question 5: To what extent do programming experience, language familiarity, and frequency of practice predict students' programming performance?

Multiple regression analysis was conducted to determine the contribution of selected factors to students' programming achievement.

Table 9: Regression Analysis Predicting Programming Performance

Predictor Variable	Beta Coefficient	Significance
Programming Experience	0.42	0.000
Language Familiarity	0.31	0.002
Frequency of Practice	0.28	0.004

The regression results indicate that programming experience, language familiarity, and frequency of practice significantly predicted students' programming performance. Programming experience had the strongest contribution among the examined factors.

4.9 Discussion of Findings

The findings demonstrate that students' programming performance varies across different programming languages. Python recorded higher achievement scores, faster completion times, and fewer programming errors compared with Java and C++. This finding suggests that programming language characteristics, such as syntax simplicity and readability, may influence students' learning outcomes. However, the results also demonstrate that

programming performance is not determined solely by programming language choice. Students' previous programming experience, familiarity with programming languages, and practice frequency significantly contributed to achievement outcomes. These findings support the view that effective programming education requires consideration of both programming language characteristics and learner-related factors. Therefore, programming curriculum design should balance language selection with appropriate teaching strategies and continuous practical engagement.

5. Conclusion

This study examined the comparative performance of Computer Science students across different programming languages, specifically Python, Java, and C++. The study was motivated by the need to understand how programming language choice influences students' learning outcomes, problem-solving ability, and programming achievement. The findings indicate that students' performance varies across different programming languages. Differences were observed in areas such as programming task completion, accuracy, error frequency, and overall assessment scores. The statistical analysis revealed that programming language differences significantly influence students' programming performance. However, the study also established that learner-related factors, particularly prior programming experience, familiarity with a programming language, and frequency of practice, play important roles in determining students' success. The results suggest that no programming language should be considered independently of the learner's background and educational context. While some languages may provide easier learning environments due to simpler syntax and readability, students' previous exposure, learning support, and instructional approaches remain important factors affecting programming achievement. Therefore, effective programming education requires careful consideration of both programming language characteristics and student-related factors. The study contributes to the understanding of programming language selection in Computer Science education and provides empirical evidence that can assist educators in developing more effective programming curricula and teaching strategies.

6. Recommendations

Based on the findings of this study, the following recommendations are proposed:

1. Adoption of Beginner-Friendly Programming Languages

Computer Science educators should consider introducing beginners to programming languages with simpler syntax and readable structures, such as QBASIC, Python, to help students develop fundamental programming concepts before progressing to more complex languages.

2. Gradual Transition Between Programming Languages

Curriculum designers should implement a structured progression when introducing multiple programming languages. Students should be allowed sufficient time to understand programming concepts before transitioning from one language to another.

3. Emphasis on Programming Practice

Institutions should provide more practical programming exercises, laboratory sessions, and problem-solving activities because regular practice improves students' programming skills and confidence.

4. Consideration of Students' Prior Experience

Lecturers should assess students' previous programming knowledge and provide additional support to students with limited programming backgrounds to reduce performance differences.

5. Use of Multiple Programming Languages for Skill Development

Computer Science programmes should continue exposing students to multiple programming languages because each language provides different learning opportunities and programming perspectives.

6. Improvement of Teaching Strategies

Programming instructors should adopt learner-centered teaching approaches, including project-based learning, interactive coding exercises, debugging activities, and continuous feedback mechanisms to improve students' programming performance.

7. Further Research

Future studies should investigate other factors affecting programming performance, such as students' motivation, learning styles, artificial intelligence-assisted programming tools, teaching methods, and differences across institutions and educational levels.

REFERENCES

- Haduong, P., Brennan, K., Sonner, G., & Sadler, P. (2018). The effects of first programming language on college students' computing attitude and achievement: A comparison of graphical and textual languages. *Computer Science Education*, 29(1), 23–48. <https://doi.org/10.1080/08993408.2018.1547564>
- Hott, J. (2026). Providing choice of programming language: Student outcomes in an algorithms course. *Proceedings of the 57th ACM Technical Symposium on Computer Science Education (SIGCSE TS 2026)*. https://consensus.app/papers/providing-choice-of-programming-language-student-hott/d7a1d9c6e4745fa4ace4ffb0405f5743/?utm_source=chatgpt
- Koulouri, T., Lauria, S., & Macredie, R. D. (2015). Teaching introductory programming: A quantitative evaluation of different approaches. *ACM Transactions on Computing Education*, 14(4), Article 26, 1–28. <https://doi.org/10.1145/2662412>
- Ling, H., Hsiao, K. L., & Hsu, W. C. (2021). Can students' computer programming learning motivation and effectiveness be enhanced by learning Python language? A multi-group analysis. *Frontiers in Psychology*, 11, 600814. <https://doi.org/10.3389/fpsyg.2020.600814>
- Mansoor, N., Peterson, C. S., Dodd, M. D., & Sharif, B. (2023). Assessing the effect of programming language and task type on eye movements of computer science students. *ACM Transactions on Computing Education*, 24(1), 1–38. <https://doi.org/10.1145/3632530>

- Orr, J. (2022). Automatic assessment of the design quality of student Python and Java programs. *Journal of Computing Sciences in Colleges*, 38(1), 27–36. https://consensus.app/papers/automatic-assessment-of-the-design-quality-of-student-orr/2a87108422ab5fd5afa5943245f5a017/?utm_source=chatgpt
- Rabelo, A., Maia, L., & Parreiras, F. S. (2018). Performance analysis of computer science students in programming learning. *Anais do Workshop sobre Educação em Computação (WEI)*. <https://doi.org/10.5753/wei.2018.3499>
- Zehra, F., Javed, M., Khan, D., & Pasha, M. (2020). Comparative analysis of C++ and Python in terms of memory and time. *International Journal of Computer Science and Information Technologies*. https://consensus.app/papers/comparative-analysis-of-c-and-python-in-terms-of-memory-and-zehra-javed/22bfd5f5f9f3b356ca667abb0102/?utm_source=chatgpt
- Bukie, P. T., Udeze, C., Obono, I., & Edim, E. (2019). Comparative analysis of compiler performances and program efficiency. *Preprints*. https://consensus.app/papers/comparative-analysis-of-compiler-performances-and-bukie-udeze/f5792e9b6fa45195b90095e1c01a38ae/?utm_source=chatgpt
- Alaria, S. K., Kaloriya, S., Jain, S., & Tomar, A. (2023). Comparative study of Java and Python: A review. *Industrial Engineering Journal*, 52(4), 2698–2708. https://consensus.app/papers/comparative-study-of-java-and-python-a-review-alaria-kaloriya/87b8071b551559ab86170c490be253bc/?utm_source=chatgpt