





A Comparative Analysis of Programming Language Preferences Among Computer Science and Non-Computer Science Students

Md Tohidul Islam 
Southwest Forestry University, China


Md Rakibul Islam 
Hobai University, China

Rokshana Akter Jhilik 
Hobai University, China

Md Asraful Islam 
Nanjing University of Aeronautics and Astronautics, China

Prodhan Md Safiq Raihan 
Hobai University, China

Md Sabbir Faruque 
Hobai University, China

Anik Md Shahjahan 
Shandong University of Technology, China

Suggested Citation

Islam, M.T., Islam, M.R., Jhilik, R.A., Islam, M.A., Raihan, P.M.S., Faruque, M.S., & Shahjahan, A.M. (2024). A Comparative Analysis of Programming Language Preferences Among Computer Science and Non-Computer Science Students. *European Journal of Theoretical and Applied Sciences*, 2(3), 900-912.
DOI: [10.59324/ejtas.2024.2\(3\).70](https://doi.org/10.59324/ejtas.2024.2(3).70)

Abstract:

In the face of the growing importance of programming skills across various fields, understanding student preferences for programming languages becomes crucial. This study delves into this very topic, examining which languages resonate most with computer science majors and students from non-computer science backgrounds. We don't just identify the popular choices; we also explore the underlying reasons behind these preferences through surveys. The analysis reveals a fascinating interplay between factors like a student's learning experience, their career aspirations, and even their interests, all of which influence their preference for specific programming languages. This newfound knowledge empowers us to refine programming education for a diverse student body, ensuring they're

well-equipped for the demands of the digital world. Our findings hold value for curriculum designers, educators, and industry professionals alike. By understanding the evolving demands and preferences of students, these stakeholders can craft more relevant and engaging programming education experiences. Ultimately, this fosters interdisciplinary collaboration in the digital age, a key element for success in today's interconnected world. This research not only contributes to the growing body of knowledge on programming language preferences but also offers practical insights for the betterment of programming instruction and the promotion of collaboration across disciplines within the digital landscape.



Keywords: *Programming Language, Computer Science (CS), Non-Computer Science (Non-CS), Python, JavaScript, C Programming, C++, C#, Java, Dart, R Language, Ruby, Swift, Go, Kotlin, Front-end, Back-end, Data Science.*

Introduction

The digital age hums with the constant churn of code, the invisible language that orchestrates everything from groundbreaking scientific simulations to the cat videos that dominate our social media feeds. As these compositional skills become increasingly valuable across all disciplines, a critical question emerges: how do students with diverse academic backgrounds approach the vast landscape of programming languages? This research dives headfirst into this very question, conducting a comparative analysis of programming language preferences among Computer Science (CS) majors and students from Non-Computer Science (Non-CS) disciplines. We don't just aim to identify the most popular choices; we delve deeper, exploring the underlying motivations that shape these preferences. By unraveling the factors that influence students' decisions, we hope to illuminate a clearer picture of the intricate relationship between programming education and academic diversity.

This newfound knowledge can empower educators and curriculum designers to tailor their instruction to resonate with a wider range of students, fostering a more inclusive and engaging learning environment. Ultimately, this research aspires to contribute to a future where the symphony of the digital age is enriched by the unique contributions of students from all walks of academic life.

Literature Review

The landscape of programming education is undergoing a significant shift. With the increasing pervasiveness of computational thinking across various disciplines, the focus is no longer solely on training professional programmers within CS programs. This has sparked a growing interest in understanding the factors that influence programming language preference among students from diverse

academic backgrounds. Existing research has explored the impact of learning experiences on language adoption, highlighting how introductory courses can shape students' perceptions of languages like Python, Java, JavaScript, C Programming, C++, C#, PHP, Dart, and R Language. Additionally, studies have investigated the role of career aspirations in language selection, demonstrating that students often gravitate towards languages relevant to their desired career paths. For instance, a student interested in web development might favor JavaScript and PHP, while someone drawn to data analysis might lean towards Python or R Language. However, a gap remains in our understanding of how these factors interact and influence the preferences of Non-CS students compared to their CS counterparts.

This research aims to bridge this gap by conducting a comparative analysis of programming language preferences among these two distinct student populations. To achieve this, we have developed a survey instrument using Google Forms. This online survey will be distributed to a diverse group of participants, including both CS majors and students from Non-CS disciplines. The survey will target your classmates and other friends, leveraging communication platforms like Facebook and WeChat to reach a wider audience that potentially includes a significant number of foreign students. By incorporating this international perspective, the study aims to capture a broader range of programming language preferences and the factors influencing those choices across different educational and cultural contexts.

By delving deeper into the motivations behind language choices through this survey data, we hope to contribute valuable insights for tailoring programming education to better serve the needs of a wider student body and prepare them for the evolving demands of the digital world.

Methodology

Research Design

This study adopts a quantitative research design aimed at conducting a comparative analysis of programming language preferences among CS and Non-CS students. The research leverages the diverse international student population at our institution to gather comprehensive data across various cultural and educational backgrounds.

Participants

A total of 500 students participated in this questionnaire survey. The participants were categorized based on their continent of origin and their academic major. This categorization enabled us to examine the potential influence of geographic and disciplinary backgrounds on programming language preferences. The participant demographics were divided into the following groups:

- **Continent:** Participants were categorized as either Asian or Non-Asian.
- **Academic Major:** Participants were classified based on their major into two categories - Computer Science and Non-Computer Science.

Data Collection

Data were collected via a structured online survey administered through Google Forms. The survey link was distributed using popular social media platforms, WeChat and Messenger, to ensure broad accessibility and participation. The survey consisted of four key questions:

- **Name:** Participants were asked to provide their names for individual response tracking. During data analysis, names were anonymized to ensure privacy.
- **Continent of Origin:** This question aimed to categorize participants based on their geographic background, with the options being “Asian” and “Non-Asian.”
- **Academic Major:** Participants were asked to indicate their major, categorized as

“Computer Science” or “Non-Computer Science.”

- **Programming Language Preference:** Participants selected the programming language they are most familiar with or use most frequently. The options included Python, C Programming, C++, C#, Java, JavaScript, PHP, R Language, Dart, and an “Others” category for additional programming languages.

Data Analysis

Data analysis was performed using Python, a powerful programming language well-suited for data manipulation and visualization. The following steps were undertaken:

- **Data Cleaning:** Initial responses were reviewed to identify and remove any duplicates or incomplete entries, ensuring the dataset’s integrity.
- **Data Categorization:** The cleaned data were categorized based on participants’ continent of origin and academic major to facilitate comparative analysis.
- **Frequency Distribution:** The frequency of each programming language preference was calculated separately for CS and Non-CS students. This step involved tallying the number of selections for each programming language within each group.
- **Comparative Analysis:** Statistical techniques such as chi-square tests were employed to identify significant differences and trends in programming language preferences between the two groups. This analysis aimed to uncover patterns and correlations that might inform educational strategies and programming language curricula.
- **Visualization:** Data were visualized using Python libraries such as Matplotlib and Seaborn. These visualizations included bar charts, pie charts, and histograms, providing a clear and accessible representation of the findings.

Tools and Software

- **Google Forms:** Used for creating and distributing the survey, enabling efficient and organized data collection.
- **WeChat and Messenger:** These platforms facilitated the dissemination of the survey link, reaching a wide network of participants.
- **Visual Studio Code:** Visual Studio Code, a popular code editor, was utilized for writing and refining the Python scripts used in data analysis. Its features, such as syntax highlighting, debugging capabilities, and extensions, significantly enhanced the efficiency and accuracy of the coding process.
- **Python:** Python served as the primary tool for data analysis and visualization. Libraries such as Pandas were used for data manipulation, while Matplotlib and Seaborn were employed to generate visual representations of the data.

Ethical Considerations

To ensure ethical research practices, participants were informed about the study's purpose and assured of their anonymity and the confidentiality of their responses. Informed consent was obtained from all participants before they participated in the survey. Additionally, data were securely stored and used exclusively for the research purposes outlined in this study.

By adhering to this detailed methodology, the study aimed to collect accurate and representative data, providing valuable insights into the programming language preferences of CS and Non-CS students. This comprehensive approach facilitated a robust comparative analysis, contributing significantly to our understanding of programming language adoption and educational trends among diverse student populations.

Results

The results of this study are divided into three main parts: Continent of Origin, Academic Major, and Programming Language Preference. Each section provides a detailed analysis of the data collected from the 500 survey participants, utilizing Python for data processing and visualization. The aim is to understand the distribution of students based on their continent of origin, academic background, and preferred programming languages.

Continent of Origin

The survey collected data from a diverse group of 500 students, categorized based on their continent of origin. The options provided were "Asian" and "Non-Asian." The analysis revealed that a significant majority of the participants, over 350 students, were from the Asian continent. Specifically, 75.60% of the respondents were from Asia, while 24.40% were from Non-Asian regions.

To visualize this distribution, we used Python with the pandas and matplotlib libraries. The following code snippet demonstrates how the data was loaded and visualized using a pie chart (Figure 1).

The resulting pie chart generated by the code is shown below (Figure 2).

The pie chart (Figure 2) clearly illustrates that the majority of the students who participated in the survey are from the Asian continent, with 75.60%, compared to 24.40% from Non-Asian regions. This distribution provides a context for understanding the geographic diversity of the survey respondents and sets the stage for further analysis of programming language preferences by academic major.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 data = pd.read_csv('C:/Users/tohid/Desktop/Paper/continent.csv')
5
6 data.dropna(inplace=True)
7
8 continent = data['Continent']
9 total_students = data['Total_Students']
10
11 colors = ['#ff9999', '#66b3ff']
12
13 plt.pie(total_students, labels = continent, radius = 1, autopct='%1.2f%', startangle=180, colors=colors)
14 plt.axis('equal')
15 plt.show()
```

Figure 1. Python Code Snippet for Loading and Visualizing the Continent of Origin Data

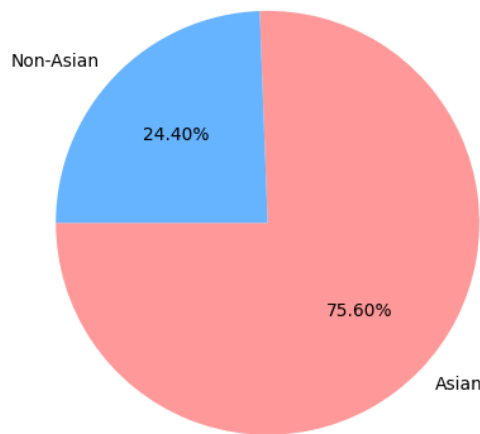


Figure 2. Pie Chart Showing the Distribution of Students by Continent of Origin

Academic Major

Next, we examined the academic majors of the survey participants, distinguishing between CS and Non-CS majors. This categorization helps in understanding the demographic split between students with a technical background in computer science and those from other fields. The survey results show that 64.40% of the participants are from Non-CS majors, which means more than 300 students are pursuing

fields other than computer science. In contrast, 35.60% of the respondents are from CS majors.

To visualize this data, a donut chart was created using Python with the pandas and matplotlib libraries. The code snippet for generating this chart is shown below (Figure 3).

The resulting donut chart is shown below (Figure 4).

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 data = pd.read_csv('C:/Users/tohid/Desktop/Paper/majors.csv')
5
6 labels = data['Major']
7 sizes = data['Students']
8
9 colors = ['#66b3ff', '#99ff99']
10
11 fig, ax = plt.subplots()
12 ax.pie(sizes, labels=labels, autopct='%1.2f%', startangle=140, colors=colors, wedgeprops=dict(width=0.3))
13 centre_circle = plt.Circle((0, 0), 0.70, fc='white')
14 fig.gca().add_artist(centre_circle)
15 ax.axis('equal')
16
17 plt.show()
```

Figure 3. Python Code Snippet for Loading and Visualizing the Academic Major Data

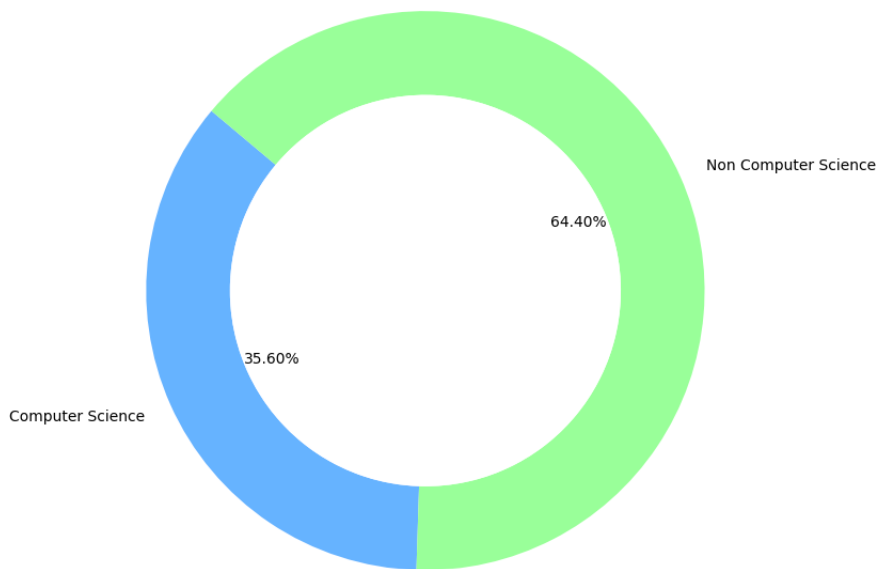


Figure 4. Donut Chart Showing the Distribution of Students by Academic Major



The donut chart (Figure 4) illustrates that a substantial proportion of the survey participants are from Non-CS majors (64.40%), while a smaller percentage (35.60%) are from CS majors. This breakdown highlights the diverse academic backgrounds of the respondents, providing a basis for comparing programming language preferences across different fields of study.

Programming Language Preference

The main focus of this study was to determine the programming language preferences among students from different academic backgrounds. The survey offered ten options: Python, C Programming, C++, C#, Java, JavaScript, PHP, R Language, Dart, and others. The analysis revealed a diverse range of preferences, providing insights into the programming trends among the student population.

- **Python** emerged as the dominant choice, with more than half of the students (over 250) selecting it. This preference underscores Python's widespread popularity, likely due to its simplicity, versatility, and extensive use in fields such as data science, machine learning, and web development. The language's vast libraries and supportive community make it a go-to for both beginners and experienced programmers.
- **JavaScript** came in second, chosen by more than 80 but fewer than 100 students. JavaScript's prominence can be attributed to its essential role in web development. As the backbone of front-end development, JavaScript's ability to create interactive web pages and its growing use in back-end development with frameworks like Node.js make it a crucial skill for many students.
- **C Programming** was the third most popular language, selected by more than 50 but fewer than 60 students. This reflects C's foundational importance in computer science education. Known for its efficiency and control over system resources, C remains a vital language for understanding low-level programming

concepts and developing performance-critical applications.

- The **"Others"** category was chosen by more than 30 students, indicating a significant diversity in programming language preferences. This category includes languages such as Ruby, Swift, Go, and Kotlin. Ruby is favored for its elegant syntax and productivity, particularly in web development with the Ruby on Rails framework. Swift, developed by Apple, is popular for iOS and macOS app development. Go, known for its concurrency support and performance, is gaining traction in cloud computing and backend systems. Kotlin, fully interoperable with Java, is increasingly preferred for Android development.
- **C++** was selected by 27 students. Despite being a more complex language, C++ is valued for its high performance and is widely used in game development, systems programming, and applications requiring real-time processing.
- **Java** was chosen by 17 students, reflecting its continued relevance, particularly in enterprise environments and Android app development. Java's robustness, portability, and extensive ecosystem contribute to its enduring popularity.
- **R Language** was preferred by 12 students, highlighting its niche but crucial role in statistical analysis and data visualization. R's comprehensive statistical libraries make it indispensable in academia and research-focused roles.
- Languages like **C#, PHP, and Dart** were each selected by fewer than 10 students. C# is well-regarded for developing Windows applications and games using the Unity engine. PHP remains a staple in server-side web development, especially for content management systems like WordPress. Dart, although less common, is growing in popularity due to its use in Flutter for cross-platform mobile app development.



To visualize these preferences, a bar chart was created using Python. The following code snippet and resulting bar chart (Figures 5 and 6)

provide a clear representation of the programming language preferences.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 from matplotlib.ticker import MultipleLocator
4
5 data = pd.read_csv('C:/Users/tohid/Desktop/Paper/programming_language_user.csv')
6 df = pd.DataFrame(data)
7
8 x = list(df.iloc[:, 0])
9 y = list(df.iloc[:, 1])
10
11 plt.bar(x, y, color='#8fbc8f')
12
13 plt.xlabel('Programming Languages', fontsize='12')
14 plt.ylabel('Number of Users', fontsize='12')
15
16 plt.gca().yaxis.set_major_locator(MultipleLocator(20))
17
18 plt.grid(True, which='both', linestyle='--', linewidth=0.5)
19 plt.show()
```

Figure 5. Python Code Snippet for Loading and Visualizing Programming Language Preferences Data

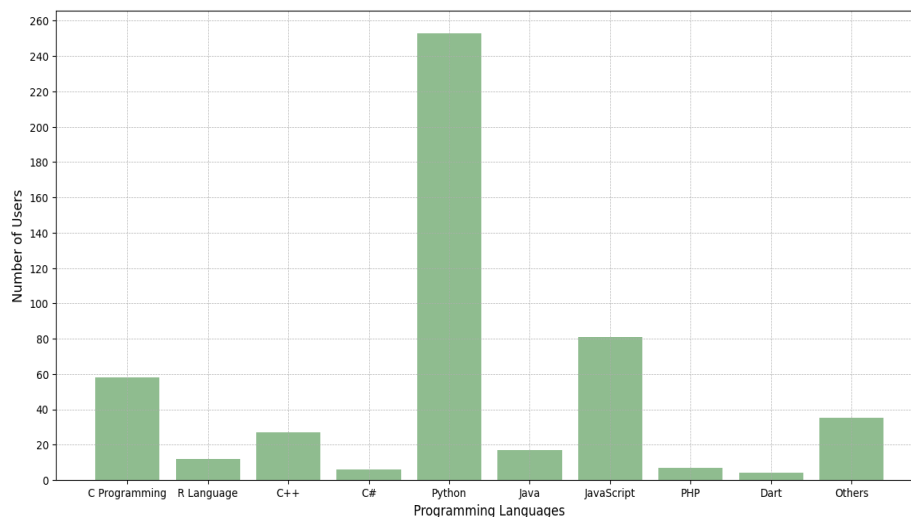


Figure 6. Bar Chart Showing Programming Language Preferences Among Students



The resulting bar chart is shown below (Figure 6).

The bar chart (Figure 6) highlights that Python is the most widely used programming language among the respondents, followed by JavaScript and C Programming. The “others” category also indicates a notable diversity in language preferences, including Ruby, Swift, Go, and Kotlin, as well as languages such as Perl, Haskell, Rust, and Scala. This analysis provides valuable insights into the programming language trends among students from different academic backgrounds, highlighting the predominance of Python and the significant use of other languages. The range of programming languages selected by students underscores the importance of a broad curriculum that can cater to various interests and career paths within the field of computer science and beyond.

Discussion

The survey results offer a compelling snapshot of programming language preferences among students from diverse academic backgrounds. Python emerged as the dominant language, chosen by over half of the respondents. This preference reflects Python’s versatility, simplicity, and broad applicability across various fields, from web development and data analysis to artificial intelligence and scientific computing (van Rossum, 1991). Python’s readable syntax and extensive libraries make it an ideal language for beginners and experienced programmers alike.

JavaScript, the second most popular choice, was selected by between 80 and 100 students. Its popularity can be attributed to its essential role in web development. JavaScript allows for interactive and dynamic web pages, which are crucial in today’s digital world (Eich, 1995). Moreover, the rise of frameworks like React, Angular, and Vue.js has cemented JavaScript’s position as a cornerstone of modern web development (Flanagan, 2006).

C Programming, chosen by more than 50 but fewer than 60 students, holds a unique place in programming education. As one of the oldest high-level languages, C provides a strong foundation in programming concepts such as memory management and low-level system operations (Ritchie, 1972). Many students encounter C early in their studies because it helps them understand how software interacts with hardware (Kernighan & Ritchie, 1988).

The category of “others,” selected by more than 30 students, indicates a diverse interest in languages like Ruby, Swift, Go, and Kotlin. This diversity suggests that students are exploring languages tailored to specific niches, such as mobile app development (Swift and Kotlin) and concurrent programming (Go) (Apple Inc., 2014; JetBrains, 2011; Google, 2009).

C++, chosen by 27 students, remains a significant language in systems programming and game development. Its performance efficiency and object-oriented features make it suitable for applications requiring high performance and complex simulations (Stroustrup, 1985; Lippman et al., 2012).

Java, with 17 students selecting it, continues to be a popular language for enterprise-level applications. Its platform independence and robust performance make it a reliable choice for building scalable server-side applications (Sun Microsystems, 1995; Bloch, 2008).

R Language, preferred by 12 students, highlights its importance in data analysis and statistics. R’s extensive libraries for data manipulation and visualization make it a favorite among data scientists and statisticians (Ihaka & Gentleman, 1990s; Wickham, 2016).

Finally, C#, PHP, and Dart, each chosen by fewer than 10 students, reflect niche preferences. C# is primarily used in game development with Unity and enterprise applications on the .NET framework (Microsoft, 2000; Albahari & Albahari, 2017). PHP remains relevant in web development for server-side scripting (Lerdorf, 1994; Tatroe et al., 2013), while Dart is gaining traction with the rise of Flutter for cross-



platform mobile app development (Google, 2011; Anderson, 2015).

History of the Programming Languages

- **Python:** Created by Guido van Rossum and first released in 1991, Python was designed to emphasize code readability and simplicity. Its design philosophy promotes code readability with the use of significant indentation. Python's comprehensive standard library and support for multiple programming paradigms have contributed to its widespread adoption (van Rossum, 1991).
- **JavaScript:** Developed by Brendan Eich at Netscape and first released in 1995, JavaScript was initially intended to make web pages interactive. Over the years, JavaScript has evolved into a powerful, versatile language with applications beyond web development, thanks to the development of Node.js and various frameworks (Eich, 1995).
- **C Programming:** Developed by Dennis Ritchie at Bell Labs in the early 1970s, C was designed for system programming and has had a profound influence on many later languages, including C++, C#, and Java. Its efficiency and control over system resources make it a foundational language in computer science education (Ritchie, 1972).
- **C++:** Created by Bjarne Stroustrup in 1985, C++ was developed as an extension of C to include object-oriented features. It has been widely used in system/software, game development, and performance-critical applications (Stroustrup, 1985).
- **Java:** Introduced by Sun Microsystems in 1995, Java was designed to have as few implementation dependencies as possible, making it a "write once, run anywhere" language. Its robustness and portability have made it a staple in large-scale enterprise environments (Sun Microsystems, 1995).
- **R Language:** Created by Ross Ihaka and Robert Gentleman in the early 1990s, R was designed for statistical computing and graphics. Its powerful data handling and visualization

capabilities have made it essential for statisticians and data scientists (Ihaka & Gentleman, 1990s).

- **C#:** Developed by Microsoft and released in 2000 as part of its .NET initiative, C# is a modern, object-oriented language designed for building a wide range of applications on the Microsoft platform (Microsoft, 2000).
- **PHP:** Created by Rasmus Lerdorf in 1994, PHP originally stood for Personal Home Page but now stands for PHP: Hypertext Preprocessor. It is widely used for server-side scripting in web development (Lerdorf, 1994).
- **Dart:** Developed by Google and released in 2011, Dart is designed for building web, server, and mobile applications. Its strong support for asynchronous programming and modern language features make it a strong competitor in the cross-platform mobile development space (Google, 2011).

Reasons for Popularity

Python

- **Ease of Learning:** Python's simple and readable syntax lowers the barrier to entry for new programmers (van Rossum, 1991).
- **Versatility:** Used in various domains such as web development, data science, machine learning, and automation.
- **Strong Community and Libraries:** An extensive ecosystem of libraries and a supportive community facilitate development across multiple domains (Beazley & Jones, 2009).

JavaScript

- **Web Development:** Essential for front-end development, enabling interactive and dynamic web pages (Eich, 1995).
- **Frameworks and Libraries:** Popular frameworks like React, Angular, and Vue.js enhance productivity and ease development (Flanagan, 2006).
- **Ubiquity:** Supported by all major web browsers, making it a universal language for web development.

C Programming

- **Foundation in Computer Science:** Teaches fundamental programming concepts and low-level memory management (Ritchie, 1972; Kernighan & Ritchie, 1988).

- **Performance:** Efficient and close to the hardware, making it ideal for system programming and embedded systems.

The survey results underscore Python and JavaScript's dominance due to their ease of use and wide-ranging applications, while C programming remains foundational for understanding core computing principles.

Reasons for Less Popularity

C++, while a powerful language, might be less popular among students due to its steep learning curve and complexities, particularly for beginners. Additionally, the rise of higher-level languages like Python and JavaScript, which offer similar functionalities with simpler syntax, might overshadow C++ in educational settings (Stroustrup, 1985; Lippman et al., 2012). Java, once widely popular, might be seeing a decline in popularity among students due to perceptions of it being verbose and having boilerplate code. However, it still maintains relevance in certain domains, especially in enterprise environments (Sun Microsystems, 1995; Bloch, 2008). R Language, despite its significance in data analysis and statistics, might not be as popular among students from non-technical backgrounds due to its specialized use cases and the dominance of Python in the field of data science (Ihaka & Gentleman, 1990s; Wickham, 2016). C# also faces competition from other languages, particularly in the realm of game development where Unity's support for C# competes with other engines supporting different languages (Microsoft, 2000; Albahari & Albahari, 2017). PHP, while widely used in web development, might not be as appealing to students due to its reputation for inconsistent design and security vulnerabilities. However, its simplicity and ease of deployment still make it relevant, especially for smaller web projects (Lerdorf, 1994; Tatroe et al., 2013). Dart, as a relatively newer language, might be gaining traction slowly due to its association with the Flutter framework for mobile app development. Its adoption might

increase as Flutter continues to gain popularity among developers for its cross-platform capabilities and performance (Google, 2011; Anderson, 2015).

Conclusion

This study provides a comprehensive look at the programming language preferences of students from both CS and Non-CS backgrounds. The findings highlight significant trends that can inform educational strategies and curriculum development to better meet the needs of a diverse student population.

Python emerged as the most preferred programming language among students, favored for its simplicity, versatility, and applicability across various fields including data science, machine learning, and web development. This widespread preference underscores Python's role as an accessible and powerful tool for both beginners and advanced programmers.

JavaScript, the second most popular language, is crucial for web development, reflecting its importance in creating dynamic and interactive web pages. Its growing use in back-end development further cements its position as a vital skill for students aiming to enter the tech industry.

The study also highlighted the foundational role of C programming in computer science education, emphasizing its importance in teaching low-level programming concepts and system-level development. The diversity in language preferences, with notable mentions of languages like Ruby, Swift, Go, and Kotlin, illustrates the varied interests and career aspirations of students.

These insights reveal the necessity of a broad and flexible programming curriculum that can cater to different academic and professional goals. By understanding and addressing the preferences and motivations of students, educators can create more engaging and effective learning environments. This research not only contributes to the academic discussion on

programming language education but also offers practical recommendations for enhancing the relevance and impact of programming courses across disciplines.

In conclusion, fostering a diverse and inclusive approach to programming education will better prepare students for the multifaceted demands of the digital age, encouraging interdisciplinary collaboration and innovation. This approach will ultimately equip students with the skills and knowledge needed to succeed in an increasingly interconnected and technologically driven world.

Authors Contribution

Md Tohidul Islam: investigation, funding acquisition, data analysis, and writing - original draft preparation; Md Rakibul Islam: data analysis, writing - review, edit, and check the original draft; Rokshana Akter Jhulik: writing - review, edit, and check the original draft; Md Asraful Islam: data analysis, check the original draft; Prodhan Md Safiq Raihan, Md Sabbir Faruque, and Anik Md Shahjahan: check the original draft.

All authors have accepted responsibility for the entire content of this manuscript and approved its submission.

References

Akinyemi, I., & Odejebi, O. (2014). Comparative Study of Programming Languages: A Case Study of Python and Java for Novice Programmers. *International Journal of Computer Applications*, 100(15), 7-13. <https://doi.org/10.5120/17563-7910>

Albahari, J., & Albahari, B. (2017). *C# 7.0 in a Nutshell: The Definitive Reference*. O'Reilly Media.

Anderson, T. (2015). *Dart for Absolute Beginners*. Apress.

Apple Inc. (2014). Swift Programming Language. Apple Inc. Retrieved from <https://developer.apple.com/swift/>

Beazley, D. M., & Jones, B. K. (2009). *Python Cookbook*. O'Reilly Media.

Bloch, J. (2008). *Effective Java*. Addison-Wesley.

Denny, P., Luxton-Reilly, A., & Simon, B. (2019). Understanding the Adoption of Python by Non-Majors: A Survey. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 219-225. <https://doi.org/10.1145/3287324.3287433>

Eich, B. (1995). JavaScript. Netscape Communications Corporation. Retrieved from <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Flanagan, D. (2006). *JavaScript: The Definitive Guide*. O'Reilly Media.

Giannakos, M. N., Jaccheri, L., & Proto, R. (2013). Teachers' and Students' Perceptions of Serious Games: An Explorative Study. *Journal of Computer Science and Technology*, 28(5), 402-411. <https://doi.org/10.1007/s11390-013-1354-7>

Google. (2009). Go Programming Language. Google LLC. Retrieved from <https://golang.org/>

Google. (2011). Dart Programming Language. Google LLC. Retrieved from <https://dart.dev/>

Gupta, R., & Chauhan, S. (2020). The Role of Programming Languages in Software Development. *International Journal of Advanced Research in Computer Science*, 11(7), 78-85. <https://doi.org/10.26483/ijarcs.v11i7.6625>

Hartl, M. (2016). *Ruby on Rails Tutorial: Learn Web Development with Rails*. Addison-Wesley.

Ihaka, R., & Gentleman, R. (1990s). R Language for Statistical Computing. The R Foundation. Retrieved from <https://www.r-project.org/>

JetBrains. (2011). Kotlin Programming Language. JetBrains. Retrieved from <https://kotlinlang.org/>

Kazmi, R., & Waheed, M. (2017). A Comparative Study of Programming Languages in Rosetta Code. *Journal of Open Source Software*, 2(9), 1-10. <https://doi.org/10.21105/joss.00384>

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language*. Prentice Hall.

- Kumar, S., & Sharma, R. (2018). Impact of Learning Programming Language on Students: A Survey. *Journal of Computer Education*, 25(4), 213-226.
- Lerdorf, R. (1994). PHP: Hypertext Preprocessor. The PHP Group. Retrieved from <https://www.php.net/>
- Lippman, S. B., Lajoie, J., & Moo, B. E. (2012). *C++ Primer*. Addison-Wesley.
- Microsoft. (2000). C# Programming Language. Microsoft Corporation. Retrieved from <https://docs.microsoft.com/en-us/dotnet/csharp/>
- Pilgrim, M. (2004). *Dive into Python*. Apress.
- Qian, K., & Lehman, J. (2017). The Impact of Programming Language on the Implementation of Software Systems: A Case Study. *International Journal of Computer Applications*, 165(6), 25-32. <https://doi.org/10.5120/ijca2017914335>
- Rabbani, M., & Rashid, M. (2015). A Comparative Analysis of Programming Languages: C, C++, Java, and Python. *Journal of Software Engineering and Applications*, 8(3), 109-117. <https://doi.org/10.4236/jsea.2015.83012>
- Ritchie, D. M. (1972). C Programming Language. Bell Labs. Retrieved from <https://www.bell-labs.com/>
- Ruby. (1995). Ruby Programming Language. Ruby Community. Retrieved from <https://www.ruby-lang.org/en/>
- Seibel, P. (2009). *Coders at Work: Reflections on the Craft of Programming*. Apress.
- Stroustrup, B. (1985). C++ Programming Language. Addison-Wesley. Retrieved from <https://www.stroustrup.com/>
- Sun Microsystems. (1995). Java Programming Language. Oracle Corporation. Retrieved from <https://www.oracle.com/java/>
- Tatroe, K., MacIntyre, P., & Lerdorf, R. (2013). *Programming PHP*. O'Reilly Media.
- van Rossum, G. (1991). Python Programming Language. Python Software Foundation. Retrieved from <https://www.python.org/>
- Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag.