2021 24th International Conference on Computer and Information Technology (ICCIT), 18-20 December, 2021.

Comparative Analysis of Six Programming Languages Based on Readability, Writability, and Reliability

Zahin Ahmed
Department of Computer Science and Engineering
Brac University
Dhaka, Bangladesh
ext.zahin.ahmed@bracu.ac.bd

Farishta Jayas Kinjol
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
farishta.jayas@northsouth.edu

Ishrat Jahan Ananya
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh
ishrat.jahan16@northsouth.edu

***Abstract* - In recent years, the development of programming languages has been centered around making them easily understandable and learnable to users. Hence, the readability, writability of languages is being constantly improved while trying to keep the performance reliable. These factors affect how many new users start to use a particular language and how many experienced programmers continue to use it reliably in real applications. Hence, this research has compared the readability, writability, and reliability of six mainstream programming languages, namely C, C++, Java, JavaScript, Python, and R, based on their theoretical characteristics. Furthermore, we conducted a survey determining the choice of a language among programmers and nonprogrammers, which complemented the results gathered from the study. We found that Python outperforms others in terms of its readability and writability, while Java is proven to be the most reliable of all. We reported our findings, insights, and a discussion on the future development of better evaluation metrics.**

Keywords— programming languages, readability, writability, reliability, efficiency.

I. Introduction

A set of instructions called the 'Syntax,' and the string of code generated from combining these syntaxes, creates a formal language used to communicate with machines such as computers. The first programming language, "Plankalkül," was designed by German scientist Konrad Zuse; however, it was never implemented [1]. Short Code, one of the first pseudocode languages, was developed by John Mauchly in 1949. Soon after, FORTRAN became the first high-level programming language commercially implemented in 1954 by John Backus [1]. There are two major categories of programming languages: Imperative languages and Declarative languages. Imperative languages such as C, C++, and Java are more commonly used by general people and are often called general-purpose languages. Declarative languages such as functional languages (LISP) have specific use cases.

In most programming languages, the difference usually lies in syntax, how the syntax is converted to machine language, efficiency, closeness to machine language (high level vs. low level). Languages such as Python, R, and Julia are much higher level compared to the languages they are built on (mostly C, C++, S, and Scheme) [2], [3]. They were developed to increase abstraction and make the whole process of coding easier and convenient [4]. The readability and writability of code hold great importance in today's digitally advancing world to all programmers and nonprogrammers. With good readability and writability of a programming language, the working of large programs can become easily understandable, reducing the learning curve for novice programmers with simpler and orthogonal syntax. Finally, experienced programmers deal with codebases used in the industry and maintained over long periods, and the programmer(s) who deal with the code change often. Hence, not only are readability and writability crucial here, but reliability also comes into play. The reliability of a programming language ensures that the code behaves the way it is supposed to and carries out its purpose at all times.

Hence, this research aims to compare the readability, writability, and reliability of 5 commonly used imperative programming languages: C (developed in the 1970s), C++ (1983), Java (1991), Python (1994), JavaScript (1995), and one functional language, R (1993). While the underlying functionalities of these languages are similar, it is essential to evaluate their readability, writability, and reliability to detect their edge on specific applications and understand the trade-offs. This study consists of data extracted from a survey and a theoretical comparison of six languages to qualitatively assess the importance of the three key language evaluation criteria and how they impact the decision-making process of choosing a programming language.

II. Literature Review

For this research, we went with a more conventional assortment of algorithms and bits of code that are commonly employed. However, there are far more excruciating ways that accentuate the limits of what programming languages can output due to the way they were designed. In a study, the researchers came up with an evaluation Framework and comparative analysis of the widely used first programming languages. Farooq et al. sought to glorify the importance of computer programming and its pivotal contribution to the computing curricula. The goal of the research was to find a way to evaluate the suitability of a programming language as an FPL (First programming language). Their research concluded that Java and Python were the top choices to be taught as FPL [5].

A similar study conducted by David Gries at the Computer Science Department of Cornell University discussed what should be taught in an introductory programming course and how the complexity of a programming language makes it difficult for students to think critically while solving problems [6].

In another work, Bhattacharya took a different approach in benchmarking the programming languages C and C++.

978-1-6654-9435-9/21/$31.00 ©2021 IEEE

They took in factors like developer competence and the whole development process in general. They conducted statistical analysis on a set of long-lived, widely-used, and open source projects like Firebox, Blender, VLC, and MySQL, which already have a substantial amount of portions of development in C and C++ [7]. They found out that C++ yielded more reliable and better quality outcomes.

Finally, Fourment et al. used standard bioinformatics programs- the Sellers algorithm, the Neighbor-Joining tree construction algorithm, and an algorithm for parsing Blast file outputs. They compared the memory usage and speed of execution implemented in 6 different programming languages. The results yielded C and C++ to be faster and use the least memory; however, Perl and Python were much more flexible than all the other languages. Whilst, C# appeared to be a bit of a compromise between the flexibility of Perl and Python and the efficient performance of C and C++ [3].

III. Readability, Writability, and Reliability

While assessing the usability and impact of programming languages, we must consider the key factors that dictate the choice of language for new users, such as the simplicity of the syntax, the learning curve and conventions employed, etc. Decidedly, these features are more formally termed Readability, Writability, and Reliability. A detailed definition that illustrates the significance of these features is as follows.

A. Readability

As the term suggests, the "readability" of a language is defined by the ease of understanding a program written in that language [1]. When earlier programming languages were developed, the main factor to be considered was the efficiency of the language. However, as the use of these languages increased, it became apparent that if users were taking a long time understanding an existing piece of code, it slowed down the process and, therefore, reduced efficiency before the computation even began. Hence, eventually, the focus shifted from machine-oriented efficiency to human efficiency.

B. Writability

The writability of a programming language is defined by the ease of creating a new program for a specific problem domain [1]. Similar to readability, the writability of a language is also heavily influenced by the problem domain it is being used in. A straightforward and typical example would be the difference in writability for a program with a Graphical User Interface (GUI). A language such as Visual BASIC or Java would have high writability for such programs as they are designed for such applications. In contrast, C would have very low writability as it is simply not intended for programs that require a GUI.

C. Reliability

The performance of a program depends significantly on the language it has been written in. The language determines whether the program will be reliable for further extension and modification in the future if required. Of course, reliability is a more abstract term, and not all languages need to have the same standard. Moreover, it depends on the purpose of the program as well.

In Table I, factors that affect readability, writability, and reliability are shown. In Table II, the metrics that define the aspects and what their expected values should or should not be, have been discussed.

TABLE I. Factors affecting Readability, Writability, and Reliability

Readability	Writability	Reliability
Simplicity	Simplicity	Simplicity
Orthogonality	Orthogonality	Orthogonality
Data types	Data types	Data types
Syntax design	Syntax design	Syntax design
Comment style	Variable naming conventions	Support for abstraction
Indentation/White spacing	Support for abstraction	Expressivity
Variable naming conventions	Expressivity	Type checking
		Exception handling
		Restricted aliasing

TABLE II. Metrics to Define Readability, Writability, and Reliability Of Languages

Factors	Metrics (expected values)
Simplicity	• Number of constructs (lesser the better) [1] • Feature multiplicity (should not be supported) [1] • Operator overloading (should not be supported) [1]
Orthogonality	• Context-dependent syntax? (should not be) [8] • Return types (should be consistent) [8] • Exceptions in rules [return types, parameter passing] (should be consistent) [1]
Data type	• Support available for all data necessities? (should be available) [1]
Syntax design	• Form and meaning (should be related to the meaning) • Compound statements (should have definitive ways to signify hierarchy)
Comment style	• Allows single line? (should allow) [5] • Allows multi-line? (should not allow) [5]
Whitespace /indentation	• Indented? (should be) • White spacing disregarded? (should not be)
Variable naming conventions	• Enforced by language? (should be)
Support of abstraction	• Level of support (should be high-level support) [1]
Expressivity	• Average number of lines needed to write a program (should be fewer) [1]
Type checking	• Done in compile-time or run-time? (should be compile-time/ statically) • Strongly typed? (should be)
Exception handling	• Available? (should be available)
Restricted aliasing	• Provided? (should be provided)

IV. Methodology

For the purpose of this paper, a unique procedure was developed that compared the six languages based on the three criteria (readability, writability, reliability) discussed previously. The methodology involved devising a metric evaluation system categorized as 'Bad,' 'Moderate,' and 'Good.' This worked as a tool to assess each of the languages for every indicator presented in Table II. Furthermore, to reaffirm the evaluation of this abstract metric system, a survey was conducted on a group of people. Their opinions were recorded for a set of questions, as discussed in section V.

A. Theoretical Comparison

To visually explain the secondary research findings, the following tables were produced that concisely mention the performance of each of the languages for the given metrics. Judgment criteria used:

BAD	MODERATE	GOOD

In Table III, the metrics that affect readability have been compared between the six languages. In Table IV, the additional metrics that affect writability have been discussed, and finally, in Table V, the remaining factors that affect reliability only have been discussed.

TABLE III. Comparison between Metrics That Affect Readability

	Simplicity	Orthogonality	Data types	Syntax design	Comment style	Whitespace/ indentation	Variable naming conventions
C	Moderate number of constructs	Some constructs are context-dependent [5]	Multiple data types are available just for integer type, but no primitive type for string	Form and meaning reflects the purpose	Allows single line comment	Not indented	Enforced by language
	Supports feature multiplicity[5]	Return types not consistent [5]		Compound statements signified by curly braces	Allows multi-line comment	Whitespace disregarded	
	Does not support operator overloading	Has exceptions in rules [5]					
C++	Large number of constructs	Some constructs are context-dependent [5]	Sufficient number of data types available	Some syntax do not reflect their meaning (such cout, cin)	Allows single line comment	Not indented	Enforced by language
	Supports feature multiplicity [5]	Return types not consistent [5]		Compound statements signified by curly braces	Allows multi-line comment	Whitespace disregarded	
	Supports operator overloading	Has exceptions in rules [5]					
Java	Large number of constructs	constructs are not context-dependent [5]	No support for complex numbers	Form and meaning reflects the purpose	Allows single line comment	Not indented	Enforced by language
	High feature multiplicity[1]	Return types consistent[5]		Compound statements signified by curly braces	Allows multi-line comment	Whitespace disregarded	
	Supports operator overloading	No exceptions in rules[5]					
JavaScript	Large number of constructs	Commonly used constructs are not context-dependent	No support for complex numbers	Some syntax is a bit vague	Allows single line comment	Not indented	Enforced by language
	High feature multiplicity	Return types consistent		Compound statements signified by curly braces	Allows multi-line comment	Whitespace disregarded	
	Does not support operator overloading	Few exceptions in rules					
Python	Fewer number of basic constructs	constructs are not context-dependent [5]	Everything is an object; proper support available for data types	Form and meaning reflects the purpose, except list and tuples	Allows single line comment	Indented	Enforced by language
	Mostly no feature multiplicity [5]	Return types consistent[5]		Compound statements signified by indented blocks	Does not allow multi-line comment	Whitespace not disregarded	
	Supports operator overloading	No exceptions in rules[5]					
R	Fewer number of basic constructs	constructs are not context-dependent	Everything is an object; proper support available for data types	Form and meaning reflects the purpose, but some are quite different than common languages	Allows single line comment	Indented	Enforced by language
	No feature multiplicity	Return types consistent		Compound statements signified by indented blocks	Does not allow multi-line comment	Whitespace not disregarded	
	Supports operator overloading	No exceptions in rules					

TABLE IV. Comparison between Metrics That Affect Writability

	Support for abstraction	Expressivity
C	No support	About five lines needed to print hello world
C++	High-level abstraction	About six lines needed to print hello world
Java	High-level abstraction	About five lines needed to print hello world
JavaScript	No built-in support	1 line to print hello world
Python	High-level abstraction	1 line to print hello world
R	High-level abstraction	1 line to print hello world

Based on our findings from Table III and IV, Python and R seem to be the most readable and writable. These languages have become very prominent in recent years, which might be attributed to their high-level readability and writability. C has very low readability and writability, which is logical considering it is the oldest and lowest level language amongst these six languages.

TABLE V. Comparison between Metrics That Affect Reliability

	Type checking	Exception handling	Restricted aliasing
C	Static type checking Not very strongly typed[5]	Does not provide support	Support provided
C++	Static type checking Mostly strongly typed[5]	Moderate support [5]	Support provided
Java	Static type checking Mostly strongly typed[5]	Strong support using catch and throws	Handles aliasing in run-time, so moderate support
JavaScript	Dynamic type checking Not strongly typed	Moderate support using try-catch throw	Aliasing not possible
Python	Dynamic type checking Mostly strongly typed[5]	Strong support using try-except [5]	Support not provided
R	Dynamic type checking Strongly typed	Moderate support	Aliasing not possible

However, in terms of reliability, Java seems to be the reigning champion. Java has been continuously developed for many years and is one of the main languages used in industry-level development and software. Hence, it is an affirmative indication that Java has the most reliable characteristics. JavaScript and R have low reliability, which is consistent with the fact that JavaScript is used only in web and server applications, and R is not used for software development.

B. Survey

The primary purpose of this survey was to check whether people's opinions reflected the findings found in the theoretical comparison. Alongside, the survey questionnaire was designed to see the correlation between the programming experience of the participants and how they perceive the readability and writability of different languages based on code snippets. The bubble sort algorithm was implemented from scratch in all six languages without using any advanced library functions. The same pseudocode was implemented in all snippets, and the codes were kept as similar as possible so that the comparison was fair.

Code listing 1: Bubble sort pseudocode

```
Arr<- read integers from a text file

N<- length of Arr

for i= 0 to N-1
        for j=0 to N-i-1
                if Arr[j] > Arr[j+1]
                        temp = Arr[j-1]
                        Arr[j-1]= Arr[j]
                        Arr[j]= temp

Print sorted array
```

The first question asked the participant about their experience in programming on a scale of 1 to 5, 1 being "Never used a programming language or seen code snippets in my life" and 5 being "Programming whiz in many languages". In the next segment, brief definitions of readability and writability and their importance were given to aid the participant in understanding the survey's goal. They were asked how important readability and writability seem to them based on the offered explanations. The rating was between 1 to 5, where 1 stood for "Not important, Efficiency/other factors are more important," and 5 stood for "Most important, don't care about efficiency". Finally, for each of the six languages, the code snippet was given which was followed by the questions to rate the readability of the snippets from 1 to 5, 1 being "Not readable, too complicated" and 5 being "Very readable, easily understandable", and for writability, where 1 stood for "Too complicated for me to write" and 5 stood for "Very easily writable, wouldn't take me more than 5 minutes".

V. Results and Analysis

A total of 58 participants filled out the survey, which was carried out using Google forms. Rather than targeting programmers only, nonprogrammers were asked for their opinions as well. Most participants were undergraduate students between the ages of 18 to 24. One important point to note is that the survey was done over a short duration of 4 days only, and most of the participants were students of the same university. Hence, more rigorous surveys or methodologies are needed to claim any of the findings discussed in this section strongly.

In Fig. 1 and 2, the boxplots of each language for the readability and writability scores, respectively, have been compared.

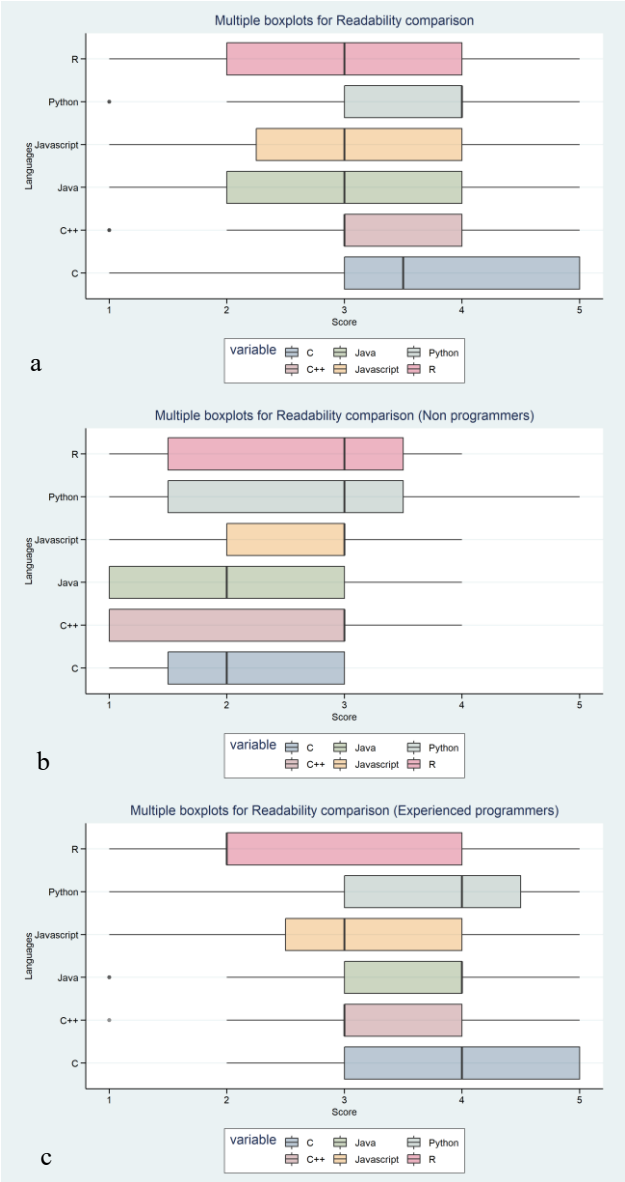

Fig. 1 Multiple boxplots showing the readability scores of each language for a) all participants, b) non/new users, and c) experienced users.

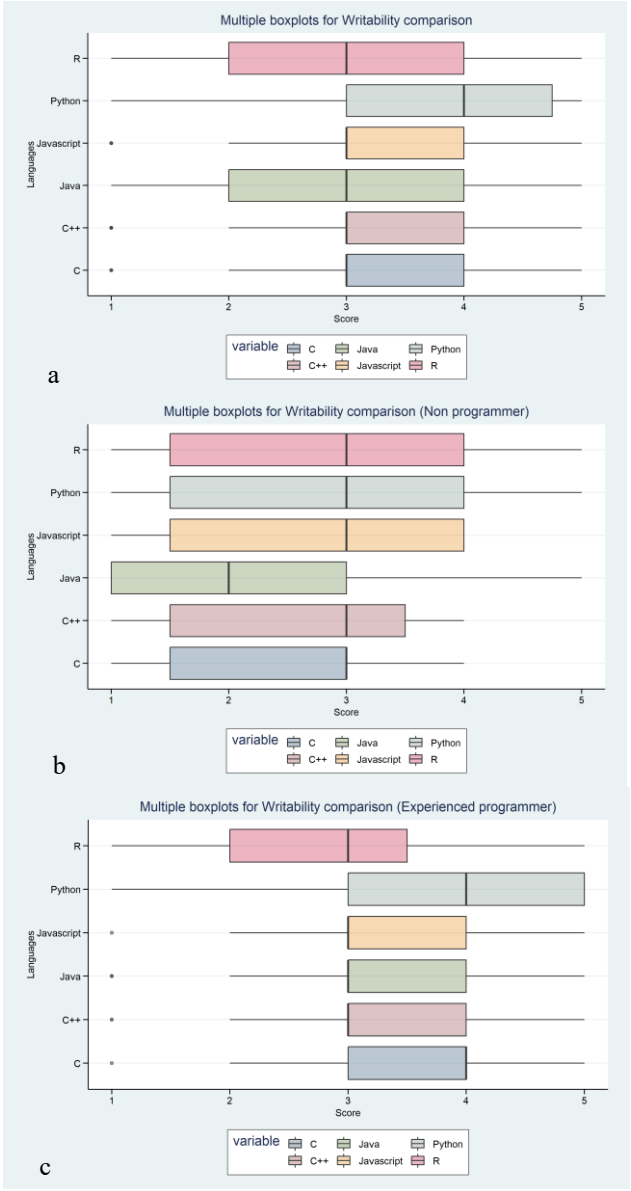

Fig. 2 Multiple boxplots showing the writability scores of each language for a) all participants, b) non/new users, and c) experienced users.

From Fig. 1, the first finding is that the majority of the participants found C to be most readable as the third quartile is much bigger than the first quartile and stretches over scores 4 and 5, which is demonstrated by the preference of experienced programmers as well. This is a possible bias in the dataset as most of the participants of this survey are students majoring in Computer Science at North South University, where the first programming language taught is C. Therefore, it is a safe assumption that most of the participants in the experienced programmers category are familiar with C, even if they are not familiar with the other languages, therefore making C the most readable language to them. The bias is further proved as the same is not seen from the perspective of new programmers, where both C and C++ have lower readability scores. Another noticeable finding is that the mean score of the importance of readability (and writability, as shown in Fig. 2) is slightly higher for experienced programmers. This finding can be attributed to the logic that non/new programmers are unaware of poor readability and writability consequences. Scores from 1 to 3 seem to be more prominent in the new programmers' boxplots, which can be attributed to the fact that nonprogrammers will find most code less readable due to their lack of experience. However, this inexperience also signifies the absence of bias, as they are judging purely based on visual characteristics. To summarize all three boxplots, new programmers seem to find Python and R more readable, consistent with our theoretical findings. Experienced programmers are geared more towards C and Python, but this cannot be strongly claimed due to the possible bias.

The boxplots in Fig. 2 suggest that most languages have similar distributions except R and Java, which show a lower range of scores. Non/new programmers score Java the lowest, which might be because Java's I/O syntax is the most complicated amongst all the languages. Other than that, no particular trend can be seen as the scores are across a wide range of values. Experienced programmers show a strong liking towards Python and find R to be the least writable. Java, JavaScript, C++, and C have very similar distributions.

In Fig.3, we have plotted the correlation between the scores of each language and compared the matrices between the new/nonprogrammer and experienced programmers. It can be seen that C and C++ are quite highly correlated, which is what we expected as C++ has been developed based on C. However, the correlation is not as strongly seen in experienced users, confirming our deduction that there was bias in their opinions. It is safe to assume that programmers who have worked with C but not C++, or vice versa, do not find the two languages similar in terms of readability. New programmers also find R to be quite similar to Python and JavaScript, which is probably because the code snippets of these languages had similar lengths and some syntax.

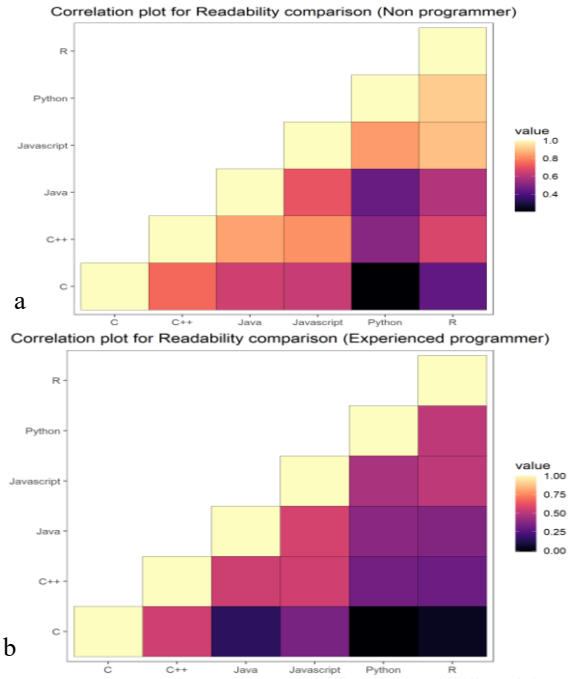

Fig. 3. Correlation plots comparing the readability of the languages between a) new users and b) experienced users

VI. Discussion

The scope of the research was to demonstrate a comparison of six popular languages in terms of their readability, writability, and reliability, and highlight characteristics that make a language popular and easy to use. The paper highlighted the significance of the three language criteria in determining the choice of language among programmers. It also detailed two methods that concluded Python and Java as the more user-friendly and reliable languages respectively.

Due to the qualitative nature of the study, verifying the reliability of Java experimentally was out of the research scope. There was no method to convey the reliability of a programming language using code snippets in a survey. As the reliability of a programming language can be detected by the performance of a program under stress conditions and unprecedented corner cases, a potential method of comparing the reliabilities of different languages could be by conducting an experiment where various programs or algorithms of different domains (such as bioinformatics algorithms, server-side programs, numerical calculations) are implemented in the various languages and their performance, efficiency, memory management etc. are monitored in numerous conditions.

This analysis can be a point of reference when developing other languages in the future or iterating improvements of existing ones. Being an abstract approach to this comparative study, the lack of resources and available research materials limited the extent of this experiment. However, the research area still holds great potential given a more extensive survey that extrapolates the developer community as a whole. Further steps could be to develop a more mathematically sound evaluation metric that does not entirely rely on conceptual features.

VII. Conclusion

In this research, six popular, mainstream languages have been analyzed based on their readability, writability, and reliability. According to our theoretical findings, Python and R were the most readable and writable, whereas Java has an advantage on the factors that only affected reliability. In our survey, not much could be concluded from the general results, however, dividing the results into the non/new-programmer and experienced programmers categories showed some interesting information. It could be seen that new/nonprogrammers chose Python and R as the more readable languages while ranking Java, C++, and C as the least readable language, which supports our theoretical findings. We also ascertained a potential bias among expert programmers to the language they're most familiar or comfortable with, which is very prevalent yet an interesting anecdote to consider. Our analysis shows that the Python programming language is the most popular choice as it delivers the best performance in terms of readability and writability. It is highly user-friendly and easy to learn and develop programs with, making it a more obvious choice for new/nonprogrammers breaking into the computer science field. On the other hand, despite being rated poorly by users, Java, theoretically, outperforms all other languages in the reliability criterion. The survey outcome complements the theoretical analysis quite well, where a similar conclusion was drawn.

References

[1] R. Sebesta, Concepts of programming languages, 11th ed. Pearson.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] R. Ihaka and R. Gentleman, "R: A Language for Data Analysis and Graphics", Journal of Computational and Graphical Statistics, vol. 5, no. 3, pp. 299-314, 1996.

[3] M. Fourment and M. Gillings, "A comparison of common programming languages used in bioinformatics", BMC Bioinformatics, vol. 9, no. 1, 2008.

[4] G. Van Rossum, F. Drake and A. Kuchling, Python tutorial. [Lincoln, Neb.]: Open Docs Library, 1999.

[5] M. Farooq, S. Khan, F. Ahmad, S. Islam and A. Abid, "An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages", PLoS ONE, vol. 9, no. 2, p. e88941, 2014.

[6] D. Gries, "What should we teach in an introductory programming course?", Proceedings of the fourth SIGCSE technical symposium on Computer science education - SIGCSE '74, 1974.

[7] P. Bhattacharya and I. Neamtiu, "Assessing Programming Language Impact on Development and Maintenance: A Study on C and C++", in International Conference on Software Engineering, 2011.

[8] E. Raymond, The art of Unix programming. Boston: Addison-Wesley, 2004.