

Comparative analysis of 6 programming languages based on readability, writability and reliability

Zahin Ahmed
Department of Electrical and Computer
Engineering
North South University
Dhaka, Bangladesh
zahin.ahmed@northsouth.edu

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—Since their inception, the development of programming languages had mostly been geared towards improving efficiency. However, in recent years, a lot of the attention has shifted towards making the languages readable and writable, while also improving reliability. 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, in this research, we have compared the readability, writability and reliability of six mainstream programming languages based on their characteristics. To verify the actuality of our findings, a survey was done to see how programmers and non-programmers judge the languages based on code snippets.

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

I. INTRODUCTION

A programming language is a formal language used to communicate with machines such as computers. Programming languages consist of a set of instructions, usually called the “syntax”, and rules on how the syntax can be used in combination with each other. 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 that was commercially implemented in 1954 by John Backus[1]. Consequently, many languages have been developed over the years for many different purposes. Some languages have been designed to be suitable for complex computations, while others cater to other domains such as Artificial Intelligence. 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 as well. Declarative languages such as functional languages (LISP) have specific use cases.

While most imperative programming languages have some basic similarities, the difference usually lies in syntax, how the syntax is converted to machine language, efficiency, closeness to machine language (high level vs low level) etc. In recent years, a lot of emphasis has been given on making languages more readable and writable for programmers. Languages such as Python, R, Julia etc. are much higher level compared to the languages they are built on (mostly C, C++, S and Scheme) [2], [3], and they were developed with the aim of higher abstraction and making the whole process of coding easier and more convenient[4]. Indeed, the readability and writability of code holds great importance to all types of users,

starting from non-programmers, novice programmers or even experienced programmers. In today’s world, almost everyone has to deal with some type of programming or code. People with no experience in coding will not be comfortable in reading or understanding code that is complicated or unable to express its purpose clearly. Similarly, novice programmers will face a huge learning curve if the syntax and rules of a language are not orthogonal and simple[5]. In this case both readability and writability are important, as they are trying to learn the syntax while also implement it themselves. Finally, experienced programmers usually deal with code that is implemented and used in industry. Such codebases are maintained over long periods of time, and the programmer(s) dealing with the code changes often. Hence, not only are readability and writability extremely important here (for the programmers to be able to understand and extend/modify the code as needed), but also reliability comes into play. Reliability of a programming language ensures that the code behaves the way it is supposed to, and carries out its purpose, at all times. This is very important in industry-level programming where the smallest of unforeseen circumstances can cause huge losses.

Hence, the aim of this research is to compare the readability, writability and reliability of 5 commonly used imperative programming languages, which are C, C++, Java, JavaScript, Python, and 1 functional language, which is R. All of these programming languages can, and is currently being, used for similar applications hence it is important to compare the differences in readability, writability and reliability of the languages, and the trade-offs that accompany the differences. The study consists of a theoretical comparison between the constructs and design of the programming languages to judge readability, writability and reliability, a survey conducted using code snippets to judge just readability and writability.

II. LITERATURE REVIEW

For the purpose of 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, 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 compared to 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].

In another work, Bhattacharya took a different approach in benchmarking the programming languages C and C++. They took in factors like developer competence and the whole development process in general. They conducted studies on 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++ [6]. They found out that C++ yielded more reliable and better quality outcomes.

In another literary article 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) [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 tied the readability, writability, and reliability of different programming languages [7].

III. READABILITY, WRITABILITY AND RELIABILITY

Before embarking on the journey of comparing these three metrics for each language, it is important to understand how each is defined and what they signify.

A. Readability

As the term suggests, “readability” of a language is essentially how easy it is to read a piece of code and understand what it is doing [1]. When earlier programming languages were developed, the main factor to be considered was 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, then it was slowing down the process and hence reducing 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, writability of a language is also heavily influenced by the problem domain it is being used in. An easy and common example for this 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, whereas C would have very low writability as it is simply not designed for programs that require a GUI.

C. Reliability

The performance of a program depends considerably on the language it has been written on. Its affects the decision of 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 and it depends on the purpose of the program it's written with.

D. Factors and properties of languages to consider

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

In the table above, factors that affect readability, writability and reliability are shown.

IV. PROGRAMMING LANGUAGES

As mentioned earlier, the six programming languages used in this research are C, C++, Java, JavaScript, Python and R. In this section, some basic information about the languages are given, to provide some context about certain features they may or may not have.

A. C

C is one of the earliest and most impactful programming languages in history. It was developed in the early 1970s at the Bell Laboratory by American computer scientist Dennis M. Ritchie and Ken Thompson.

B. C++

C++ was created by Bjarne Stroustrup as an extension of C programming language that introduced class functionality, which is why it was initially called C with classes. It was renamed C++ in 1983.

C. Java

A small team of engineers called Green Team in Sun microsystems initiated the JAVA language project in 1991.

D. JavaScript

JavaScript, was created by Brendan Eich at Netscape. JavaScript is one of the core technology of World Wide Web.

E. Python

Guido van Rossum started working on Python in 1989 as a successor to ABC programming language. It was first released in 1991 although, version 1.0 of Python was released in 1994.

V. METHODOLOGY

For the purpose of our paper, we defined a unique procedure to compare the six languages based on readability, writability and reliability. First of all, we created an evaluation metric of three standards: ‘Bad’, ‘Moderate’, ‘Good’ to judge each of the languages for every criterion in table 1. This is done to form a theoretical comparison based on the characteristics of each language. However, since the evaluation metrics are quite abstract, and very subjective, we chose to conduct a survey to check whether our theoretical comparison is reflected in the preference and judgement of actual users. Hence, we detailed a set of questions in the form of a survey which is further discussed in section 7b.

A. Theoretical comparison

To visually explain the findings of our secondary research, we produced the following tables that concisely mentions the performance of each of the languages.

Judgement criteria used:

BAD	MODERATE	GOOD
-----	----------	------

In Table II, the factors that affect readability have been discussed. In Table III, the extra factors that affect writability have been discussed and finally in Table IV, the remaining factors that affect reliability only have been discussed.

TABLE II. COMPARISON BETWEEN FACTORS 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 available just for integer type, but no primitive type for string	Form and meaning reflects 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 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 are 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 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 purpose, but some are quite different than common languages	Allows single line comment	indented	Enforced by language
	No feature multiplicity	Return types consistent				Whitespace not disregarded	

	Supports operator overloading	No exceptions in rules		Compound statements signified by indented blocks	Does not allow multi line comment		
--	-------------------------------	------------------------	--	--	-----------------------------------	--	--

Based on our findings so far, Python and R seem to be the most readable and writable. Both of these languages have become very prominent languages in recent years, which might be attributed to their high level readability and

writability. C has very low readability and writability, which makes sense considering it is the oldest and lowest level language amongst these 6 languages

TABLE III. COMPARISON BETWEEN FACTORS THAT AFFECT WRITABILITY

	Support for abstraction	Expressivity
C	No support	About 5 lines needed to print hello world
C++	High level abstraction	About 6 lines needed to print hello world
Java	High level abstraction	About 5 lines needed to print hello world
Python	High level abstraction	1 line to print hello world
JavaScript	No built in support	1 line to print hello world
R	High level abstraction	1 line to print hello world

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 does make sense that it has the most reliable characteristics. JavaScript and R

both have low reliability, which is consistent with the fact that JavaScript is mostly used in web and server applications, but not other types of software and R is not used for software development at all.

TABLE IV. COMPARISON BETWEEN FACTORS THAT AFFECT RELIABILITY

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

B. Survey

The main purpose of this survey was to check whether people's opinions reflected the findings found in the theoretical comparison. However, we also wanted to see if there was a difference in preference or judgement of readability and writability between non/new programmers and experienced programmers. The questions asked the participants the following questions: how familiar they were with programming, how important readability and writability was to them, and then a code snippet of a bubble sort algorithm written in each language was shown and they were asked to score the languages based on readability and writability, in separate questions.

VI. RESULTS AND ANALYSIS

In this section, the results of the survey and efficiency experiments have been discussed. The results of each section is followed by its analysis, however farther research is needed to strongly establish our reasoning.

A. Survey results

A total of 58 participants filled out the survey which was carried out using Google forms. Rather than targeting programmers only, non-programmers were asked for their opinions as well. 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, farther more rigorous surveys or methodologies are needed to strongly claim any of the findings discussed in this section.

In Fig.1, the boxplots of each language for the readability scores are shown for each category.

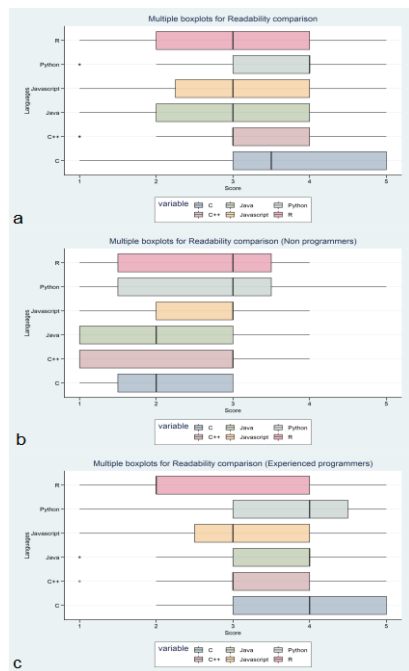


Fig. 1. multiple boxplots showing the readability scores of each language for a) all participants, b) non/new users, c) old users

From Fig. 1, the first finding is that majority of the participants found C to be most readable as the third quartile is much bigger than the first quartile and stretches over score 4 and 5, and this is explained by the preference of experienced programmers as well. This is a possible bias in the dataset as most of the participant of this survey are students majoring in Computer Science in North South University, where the first

programming language taught is C. The bias is farther proved as the same is not seen from the perspective of new programmers, where both C and C++ have lower readability scores. Scores from 1 to 3 seem to be more prominent in the new programmers' boxplots, which can be attributed to the fact that non-programmers will find most code less readable due to their lack of experience. However, this inexperience also signifies the lack of bias, as they are judging purely based on the visual characteristics. To summarize all 3 boxplots, new programmers seem to find Python and R more readable, which is consistent with our theoretical findings. Experienced programmers are geared more towards C and Python, but this cannot be solidly claimed due to the possible bias.

In Fig.2, the boxplots of each language for the writability scores are shown for each category.

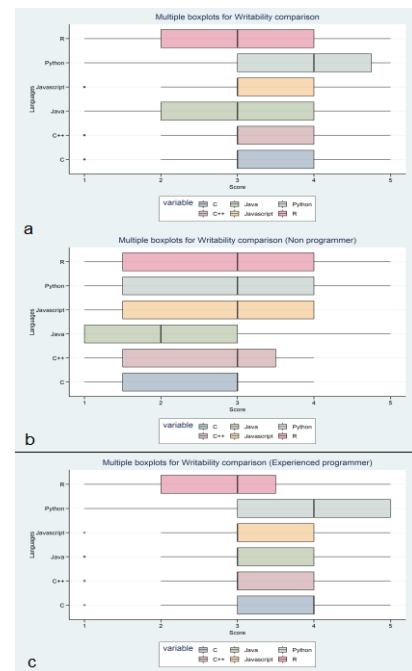


Fig. 2. multiple boxplots showing the writability scores of each language for a) all participants, b) non/new users, c) old users

Judging from the boxplots showing writability in Fig.2, most languages seem to have similar distributions except R and Java, which show lower range of scores. Non/new programmers score Java the lowest, which might be because Java's I/O syntax is most different 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 strong liking towards Python and find R to least writable. Java, JavaScript, C++ and C have very similar distributions.

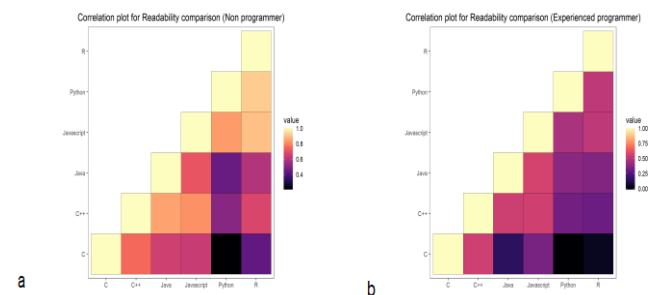


Fig. 3. Correlation plots comparing the readability of the languages between a) new users and b) old users

In Fig.3, we have plotted the correlation between the scores of each language and compared the matrices between the new/non-programmer and experienced programmer categories. 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, which further confirms 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 due to the fact that the code snippets of these languages had similar lengths and some syntax as well. No significant correlation can be seen in the scores obtained from experienced programmers.

VII. DISCUSSION

While this research was intended to compare six languages based on their readability, writability and reliability, it is in no way a complete study. It is important to define a mathematically sound methodology to properly assess these metrics, as they are quite subjective. Issues that were faced during this research included lack of resources, as not much work has been done on this topic. Hence, this is a field with potential that should be explored. We also faced hardware issues as much of the calculation is RAM dependent, which is why repeating the computationally intensive algorithms for bigger numbers was not possible. Finally, the scope of the survey was not wide enough to make a solid claim.

VIII. 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/non-programmers 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 found that there is a possible bias in experienced programmers to the language they're most familiar or comfortable with, which is predictable yet interesting. Hence, based on readability and writability, Python is the winner in this research.

However, based on reliability, the only language in which we did not face any issue was Java. Java was able to complete all the experiments smoothly, and had no unpredicted issues that hampered our coding. This supports our theoretical findings well, hence according to this research, Java is the most reliable language amongst this six programming languages.

REFERENCES

- [1] R. Sebesta, Concepts of programming languages.
- [2] R. Ihaka and R. Gentleman, "R: A Language for Data Analysis and Graphics," *J. Comput. Graph. Stat.*, vol. 5, no. 3, pp. 299–314, 1996, doi: 10.1080/10618600.1996.10474713.
- [3] M. Fourment and M. R. Gillings, "A comparison of common programming languages used in bioinformatics," *BMC Bioinformatics*, vol. 9, pp. 1–9, 2008, doi: 10.1186/1471-2105-9-82.
- [4] G. van Rossum, "Python tutorial, May 1995," *CWI Rep. CS-R9526*, no. CS-R9526, pp. 1–65, 1995, [Online]. Available: <http://oai.cwi.nl/oai/asset/5007/05007D.pdf>.
- [5] M. S. Farooq, S. A. 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, pp. 1–25, 2014, doi: 10.1371/journal.pone.0088941.
- [6] P. Bhattacharya, "Assessing Programming Language Impact on Development and Maintenance : A Study on C and C++ Categories and Subject Descriptors," *Methodology*, no. 2, pp. 171–180.
- [7] D. Gries, "What should we teach in an introductory programming course?," *Proc. 4th SIGCSE Tech. Symp. Comput. Sci. Educ. SIGCSE 1974*, pp. 81–89, 1974, doi: 10.1145/800183.810447.
- [8] J. D. Gannon and J. J. Horning, "Language Design for Programming Reliability," *IEEE Trans. Softw. Eng.*, vol. SE-1, no. 2, pp. 179–191, 1975, doi: 10.1109/TSE.1975.6312838.
- [9] E. S. Raymond, "UNIX Philosophy," *art UNIX Program.*, p. 560, 2003, [Online]. Available: <http://portal.acm.org/citation.cfm?id=829549>.