IJIST
P ISSN : 2618-1630
E ISSN : 2909-6130
Recognized Journal

Original Article

The Impact of Language Syntax on the Complexity of Programs: A Case Study of Java and Python

Kashif Munawar[1], Muhammad Shumail Naveed[1]
[1]Department of Computer Science & Information Technology, University of Balochistan, Quetta, Pakistan.
* Correspondence: Kashif Munawar, kashifmunawar74@gmail.com.

Citation |Munawar.K, Naveed, S M "The Impact of Language Syntax on the Complexity of Programs: A Case Study of Java and Python". International Journal of Innovations in Science and Technology. Vol 4, Issue 3, 2022, pp: 683-695
Received | May 28, 2022; **Revised |** June 22, 2022; **Accepted |** June 28, 2022; **Published |** June 30, 2022. DOI:https://doi.org/10.33411/IJIST/2022040310

Programming is the cornerstone of computer science, yet it is difficult to learn and program. The syntax of a programming language is particularly challenging to comprehend, which makes learning arduous and affects the program's testability. There is currently no literature that definitively gives quantitative evidence about the effect of programming language complex syntax. The main purpose of this article was to examine the effects of programming syntax on the complexity of their source programs. During the study, 298 algorithms were selected and their implementations in Java and Python were analyzed with the cyclomatic complexity matrix. The results of the study show that Python's syntax is less complex than Java's, and thus coding in Python is more comprehensive and less difficult than Java coding. The Mann-Whitney U test was performed on the results of a statistical analysis that showed a significant difference between Java and Python, indicating that the syntax of a programming language has a major impact on program complexity. The novelty of this article lies in the formulation of new knowledge and study patterns that can be used primarily to compare and analyze other programming languages.

Keywords: Cyclomatic Complexity, Programming Languages, Syntax, Java, Python.

Acknowledgment.
The authors would like to thank the anonymous reviewers for their comments in updating the quality of the article.

Special thanks to Lizard for providing a scalable tool for static analysis **CONFLICT OF INTEREST:**
The authors of this paper declare no conflict of interest.

Author's Contribution.
All authors contributed significantly to the study and all authors agree with the content of the manuscript in IJIST.

Project details. NiL

INTRODUCTION

Information and communication technologies (ICT) have influenced every sector of society, from education to medicine, and from aerospace to industry. ICT enables the design and delivery of digital goods and services, allowing individuals and companies to improve their quality of life, productivity, and standard of living. When compared to traditional learning, computer-aided instruction has a positive impact on students' learning [1]. Big data analysis in polypharmacy benefits the medical industry by revealing connections, patterns, and trends that were previously undetectable using traditional statistical methods [2].

Computers are a fundamental component of ICT, and software is a critical component of any computer system. There is currently no field that does not use computers or computing technologies. Without computers, civilization would be nowhere near where it is now [3]. Computing plays a vital role in responding to global boundaries and building a civilization that can effectively adapt to them [4].

A computer programming language is an artificial language designed to express computations that may be carried out by a computer or machine [5]. Programming is at the heart of computer science [6] and is an extremely rewarding discipline [7].

There have been important achievements in programming linguistics, and a variety of programming languages have been developed, with C, C++, Java, and Python being some of the most popular. The C programming language is appropriate for developing operating systems and a broad array of other application software, from supercomputers to embedded systems [8]. The underlying concept of object-oriented programming is aided by C++. It supports both procedural and object-oriented programming and has adequate support for dynamic memory management [9]. The most promising factors are portability, write-once, and run-anywhere [10]. Java supports object-based programming, and the syntax is quite friendly. Its most notable features are portability, security, and high abstraction. It also supports dynamic features [11]. Java is the most suitable for data science, as it supports a large number of third party libraries [12]. Java is regarded as one of the strongest and most useful languages ever developed. In many fields, it is the programming language that is most extensively used. Java has simple syntax that is simple to write, learn, comprehend, maintain, and the code is simple to debug. Java is also less complex than languages like C and C++ due to the removal of several of these languages' sophisticated features, including the explicit pointer notion, storage classes, operator overloading, and many more. Java lowers the risks and threats to security by avoiding the usage of explicit pointers. A pointer keeps track of another value's memory address, which might lead to unlawful memory access. Eliminating the idea of pointers solves this problem. Additionally, each Java program has a security manager that allows to specify the access guidelines for classes. Java's Java Virtual Machine is in charge of managing Java's automatic memory management. As a multithreaded language, Java allows for the simultaneous operation of several threads.

Python is a powerful, interactive, object-oriented, and interpreted scripting language. High readability is one of Python's design goals. It has fewer syntactical structures than other languages and typically employs English keywords rather than punctuation. The interpreter processes Python while it is running. ABC, Modula-3, C, C++, Algol-68, SmallTalk, the Unix shell, and other scripting languages are the languages that Python is derived from. Python is created under an open source licence that has been accepted by OSI. Because of this, using it is totally free, even for business. Python can be used to create sophisticated scientific and numerical applications on the desktop, the web, and other platforms. Generators and list

comprehensions are two sophisticated programming features found in Python. Python's automatic memory management also eliminates the requirement for manual memory management. Python provides a wide range of libraries and tools for different domains of computation. It supports extensive libraries for theory of automat, numerical computation, statistical computation, machine learning, data science, computer vision, and natural language processing. Py Simple Automata, TensorFlow, Scikit-Learn, Numpy, Keras, PyTorch, and Light GBM are some of the major Python libraries.

So many integrated and effective teaching methods and tools have indeed been developed as a result of the emergence of many programming languages. However, the studies [13]–[16] identified that it is difficult to learn computer programming. Beginners have found it difficult to develop programs to solve problems using programming languages. The main causes of this learning challenge include a lack of mathematical background and problem-solving skills, as well as an odd lexical structure and sophisticated syntax of the programming language. It is normal for students to lose confidence and motivation when confronted with hurdles such as the programming environment, language syntax knowledge, problem understanding, and debugging [17]. Programming languages are unnatural, requiring a high level of abstraction that is difficult to attain for novice programmers.

Programming languages should have a straightforward syntax to make the learning process easier [18]. However, a systematic review of the literature highlighted programming language syntax as a learning barrier [19]. Programming languages should have a straightforward syntax to make the learning process easier. The problems of learning programming languages have been thoroughly researched from several angles. Faculty review, discussion, and consensus were all used to choose a programming language for an introductory programming course. Formal studies on language complexities are frequently conducted by conducting experiments with programming languages on a group of users and observing their issues, behavior, and programming performance. Formal studies on language difficulties are typically conducted by testing programming languages on a set of users and analyzing their issues, behavior, and programming performance. These studies are quite effective, but the results' reliability and usability are largely dependent on the domain and the subjects in the evaluation. Eventually, quantitative studies that comprehensively examine programming languages, their syntax, denotation, and determine their impact on program complexity are required.

Since programmers tend to focus more on a programming language's syntax [20], [21], which is inextricably linked to their comprehension of programming language constructs, it is critical to examine the impact of language syntax on program complexity. Complexity analysis is also essential from the standpoint of software engineering, as it has an impact on software testability and maintenance. Software maintenance is an essential component of software engineering. The complexity of the programming code has a direct link to maintenance. The development and improvement of a programming language would be aided by the identification of a less difficult programming language. Analyzing the programming language would aid in the detection of syntactic structures that may have an impact on a program's inherent complexity. The findings would also be useful in determining which language requires the least amount of testing.

Quantitative software engineering has defined various software metrics for code analysis. Software metrics are used to assess several aspects of a program, including time, complexity, difficulty, maintainability, and dependability. Cyclomatic Complexity (CC)and

Halstead Complexity (HC)measures are two key metrics that are widely used to assess programs.

To our knowledge, no significant study has quantified the impact of language syntax on program complexity. This research tries to close this gap by examining the impact of language syntax on program complexity using quantitative software metrics. The most popular and extensively used programming languages, Java and Python, have been chosen for study.

RELATED WORK

Several studies have been conducted to compare programming languages and examine code complexity. Shoaib et al. [22] analyzed the difficulty, time, and effort involved in implementing fundamental computer algorithms. The study examined 200 algorithms and the 800 programs that went with them. The Halstead complexity metric was utilized in the analysis. According to the study, developing basic-level algorithms in Python is easier than in C, C++, or Java. Its programs are developed designed and comprehended in less time with the fewest possible bugs.

Balogun [23] examined C++ and Java by analyzing the difficulty, effort, time, and delivered defects in the implementation of traditional algorithms. The Halstead complexity metric was used to evaluate the implementation of 200 algorithms in the study. According to the study, Java programming is more difficult than C++ programming. Java needs more time and effort than C++. According to the findings, C++ is better than Java for implementing classical algorithms, making it a good choice for a beginner programming course.

Khan et al. [24] examined the implementation of traditional algorithms in Python, C++, Visual BASIC, C#, and Java. The Halstead Method, LOC, and Cyclomatic Complexity metrics were employed in the investigation. Lines of Code and Cyclomatic Complexity have low complexity values for linear search and somewhat greater values for binary search, according to the study. In selected languages, the Halstead complexity shows a significant difference between binary and linear search. According to the study, C++ has the greatest binary and linear search values, whereas Visual BASIC has the lowest. Visual BASIC has the lowest line of code complexity, whereas Java has the greatest.

Muriana et al. [25] investigated how design patterns affect program size and complexity. The study examined at Java and C++ programs. During the research, all 23 Gang of Four design patterns were studied in pairs of programs developed with and without the use of design patterns. According to the study, using design patterns reduces the average complexity of programs. However, there was a negative impact on the size, Chidamber and Kemerer metrics, and the number of classes.

Costanza et al.[26], investigated the complexity of C++, Java, and Python programming. Chidamber and Kemerer (CK) were utilized, and the results were examined in the languages chosen. Python, Java, and C++ are just a few of the programming languages available. The study's findings revealed that when employing multiple inheritances, Python and C++ provide superior reusability of software, whereas Java has serious flaws that result in the bad object structure.

Simple syntax is well known for reducing syntax errors and improving programming performance. The relationship between Halstead Complexity, Cyclomatic Complexity, and the number of faults detected in software has been studied by Emmerich et al. [27]. Using an open-source project, the study investigated the density of defects (DD) and its association with the HC and CC metrics. Halsted complexity and Cyclomatic Complexity have similar connections to the Defect Density metric, according to the study. Furthermore, their

significant positive linear correlation leads to the conclusion that HC and CC are two software metrics that are consistent when it comes to defect density.

A Multi-paradigm Complexity Metric (MCM) was developed as a way to measure code complexity for multi-paradigm programs. In a study [28], the sample of C++ and Python was examined with MCM. Other complexity metrics, such as effective line of code (eLOC), cyclomatic complexity metric, and Halstead complexity measures, are used to validate the outcomes of the presented metrics. The complexity numbers for sampled Python code were found to be lower than those for C++ code, according to the research.

Most studies comparing programming languages have analyzed the effects of lexicons, structure, and denotation on required effort, time, errors, and program size. Similarly, some studies have examined the reusability and the effective lines of code. However, no significant study has examined the effect of a programming language on program complexity. Complexity analysis is crucial as it affects program testability and maintainability. The present study seeks to fill this gap by examining the effects of language syntax on the complexity of the program.

MATERIAL AND METHODS

The fundamental goal of this research is to determine whether the language syntax affects the program's complexity. Java and Python were chosen for the investigation. Java is a programming language that focuses on objects and classes. Python is a multi-paradigm programming language that can handle structured, imperative, object-oriented, and functional programming. The following research methodology is being used to address the defined objective.

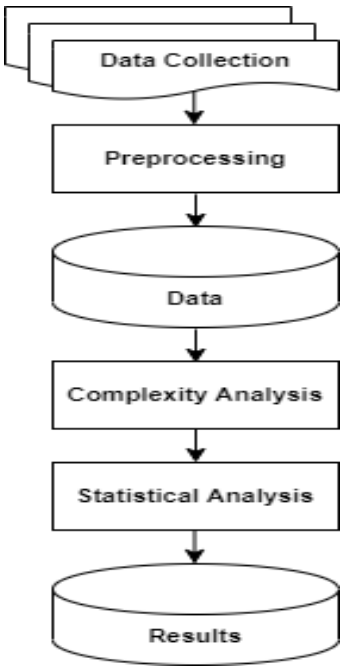

Figure 1. Research Methodology

During the process of data collection, 337 algorithms were initially selected for the study. However, after consultation with three programming experts, 298 algorithms were chosen for the study. The algorithms that are normally covered in programming and data-structures courses are typically chosen for this study.

The equivalent code in Java and Python is generated using the high-level code generator [29] and maintained as a dataset (programming corpus) of the study. The resulting code is then double-checked to ensure that it does not contain any dead code or useless code. The analysis of the dataset is divided into two stages. The basic information about code in the programming corpus is collected in the first phase by identifying the lines of code and tokens. A token is an abstract symbol that represents a specific type of lexical element in the context of language design.

The actual complexity of the dataset is computed in the second stage of the analysis. Most of the existing studies on program complexity have just looked at cyclomatic complexity. However, this work calculates cyclomatic complexity as well as strict cyclomatic complexity and essential cyclomatic complexity.

Cyclomatic complexity is a prominent and widely used software statistic for determining a program's complexity. It's a count of how many decisions there are in the source code. The code becomes increasingly difficult as the count rises. The number of test cases necessary for branch coverage in a structured code module is estimated using cyclomatic complexity. Control flow graphs are commonly used to estimate logical independent paths, which are then used to compute cyclomatic complexity. An independent path is one that contains at least one edge that has never been traversed by any other path.

To estimate the testing time and resources needed to test a software module, the value of CC is combined with past project data. Furthermore, the CC value and control flow graph provide the tester with a new tool for creating white-box test scenarios based on the concept of a path [30]. Many metrics and static analysis methods incorporate cyclomatic complexity. It is commonly used by developers to assess the cognitive difficulty and to focus their refactoring efforts. It is possible to increase the testability and reduce the complexity of code by lowering the cyclomatic complexity.

Cyclomatic complexity is also useful in the evaluation of safety-critical software [31], in developing an application for blood chemistry determination, analyzing Django Web-framework code quality measures [32], and comparing two automation system [12].

Cyclomatic complexity is a dialect of Strict Cyclomatic Complexity (SCC). The number of branches in a code is primarily used to determine CC. The number of conditions within each branch is also taken into account by SCC. Strict Cyclomatic Complexity is an estimate of how many test cases are required to exercise all of the code paths, including all of the numerous ways to trigger each branch.

Essential Cyclomatic Complexity (ECC) is a variation of Cyclomatic Complexity that is used to determine how many unstructured constructs are present in a module. This statistic assesses the code's quality and structure. It aids in the modularization process by predicting maintenance effort.

The present study analyzed the complexity of the dataset by using "Understand," which is a configurable integrated development environment (IDE) that provides useful static code using a variety of visualizations, documentation, and metrics.

To validate the accuracy of the results, Lizard, another effective tool for estimating CC, was used to review the programming code of the sampled algorithms. Similarly, the equivalent code of sampled algorithms was manually checked to ensure that the obtained results were accurate.

The acquired results were statistically evaluated using SPSS (ver. 25), a widely used software application for data analysis. Similarly, R was used to visualize the results, which is a computer language for statistical computing and graphics.

RESULTS AND DISCUSSION

The primary goal of this study was to examine the effect of programming language syntax on program complexity. Table 1 shows the actual results received after the dataset was analyzed.

Table 1. Results of Analysis

Method	Language	Mean	Median	Std. Dev.	Minimum	Maximum
Tokens	Java	250.19	216	139.53	28	818
	Python	165.02	136.5	106.42	18	759
Line of Code (LOC)	Java	37.78	32	19.3	10	111
	Python	23.24	19	15.22	4	133
Cyclomatic Complexity	Java	8.14	7	4.66	1	29
	Python	7.36	7	4.11	1	28
Strict Cyclomatic Complexity	Java	8.71	8	5.2	1	32
	Python	7.86	7	4.5	1	30
Essential Cyclomatic Complexity	Java	4.11	3.5	2.89	1	19
	Python	3.95	3	2.64	1	18

The mean token score in the Java sample was 250.19, while it was 165.02 in the Python dataset. Similarly, Python programs have a minimal level of variance. The percentage difference of 41.03% in mean token scores identifies Python's brevity over Java.

The mean LOC score for the Java corpus was 37.78, whereas the Python corpus had a score of 23.24. Java programming takes more lines of code than Python, based on the percentage difference of 47.66.

The average cyclomatic complexity of the Java corpus was 8.14, whereas the Python corpus was 7.36. Python has a low variance in terms of complexity. The mean cyclomatic complexity of Java and Python was found to differ by 10.06%, indicating that Python's programs are comparatively less complicated than Java's corresponding programs.

The average strict cyclomatic complexity for the Java corpus was 8.71, while Python's was 7.86. Python's corpus was less varied than Java's. Python supports the development of programs that are less complex than Java, as seen by the mean difference of 10.25% in the SCC averages.

Java applications had a mean Essential Cyclomatic Difficulty of 4.11, which was slightly greater than Python's complexity (3.95). Python's support for the development of less difficult programming code was shown by a mean difference of 3.97%. For a better illustration of the results, the line charts are generated and shown in Figure 2.

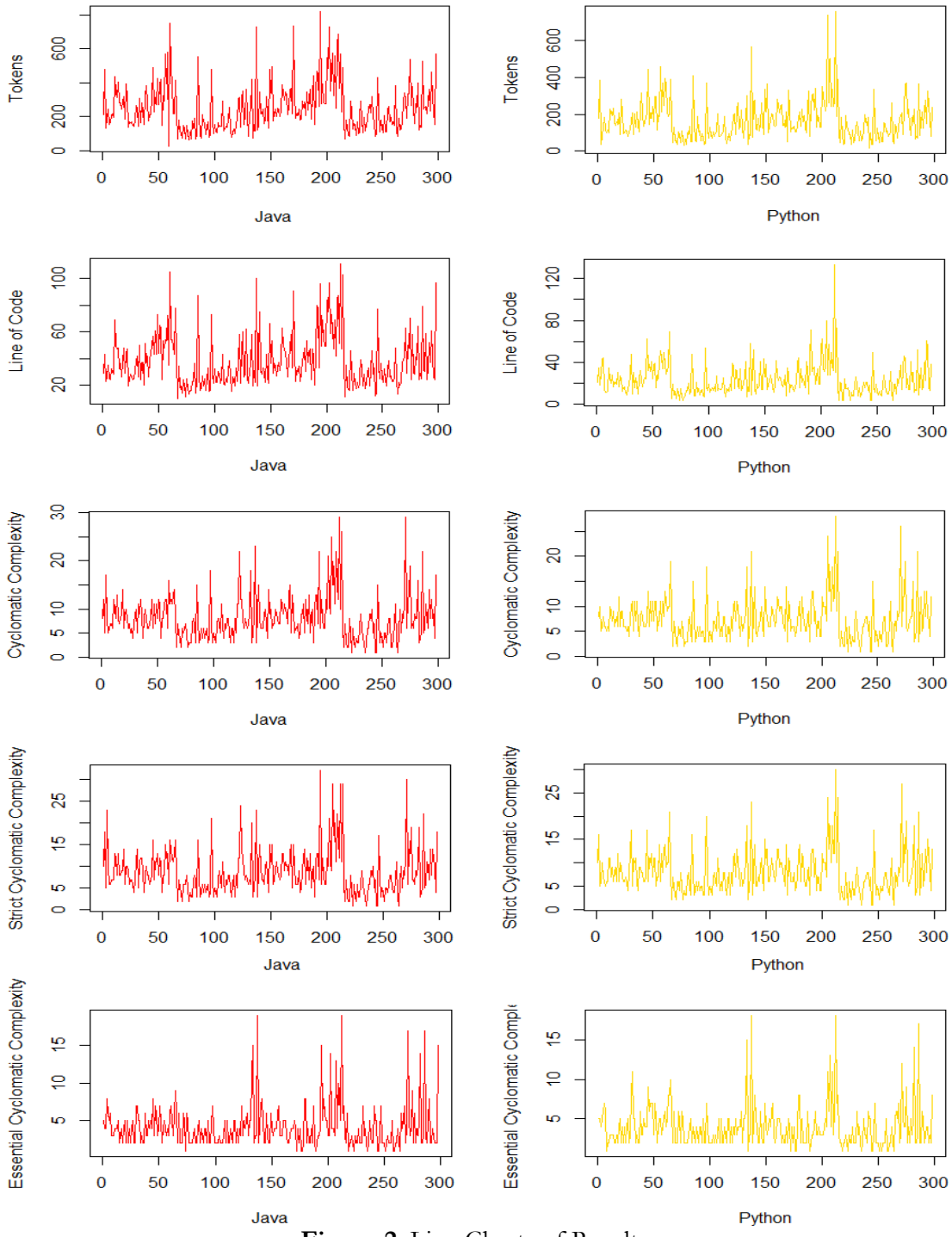

Figure 2. Line Charts of Results

The series of plots shown by line charts demonstrated that Python's support for the sampling algorithms in development programs is better than Java on every scale.

The results of the study are assessed for normality by SPSS for statistical analysis. Table 2 summarizes the details of the Kolmogorov-Smirnov and Shapiro-Wilk tests.

Table 2. Result of Normality Tests

Method	Language	Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Tokens	Java	0.11	298.00	<0.05	0.90	298.00	< 0.05
	Python	0.12	298.00	< 0.05	0.87	298.00	< 0.05
Line of Code	Java	0.13	298.00	< 0.05	0.89	298.00	< 0.05
	Python	0.15	298.00	< 0.05	0.84	298.00	< 0.05
Cyclomatic Complexity	Java	0.15	298.00	< 0.05	0.88	298.00	< 0.05
	Python	0.15	298.00	< 0.05	0.88	298.00	< 0.05
Strict Cyclomatic Complexity	Java	0.15	298.00	< 0.05	0.88	298.00	< 0.05
	Python	0.16	298.00	< 0.05	0.89	298.00	< 0.05
Essential Cyclomatic Complexity	Java	0.19	298.00	< 0.05	0.75	298.00	< 0.05
	Python	0.18	298.00	< 0.05	0.77	298.00	< 0.05

The sampled algorithms under study were functionally quite varied, and as a result, the tokens, lines of code, and their complexities were quite different as well. As a result, non-normality is observed in the Java and Python corpus at all scales. Table 3 shows the ranks of the study, which were further analyzed using the nonparametric Mann-Whitney U Test.

Table 3. Detail of Ranks

Method	Language	Mean Rank	Sum of Ranks
Tokens	Java	359.61	107164.50
	Python	237.39	70741.50
LOC	Java	375.84	112001.50
	Python	221.16	65904.50
Complexity	Java	313.99	93570.50
	Python	283.01	84335.50
Strict	Java	313.06	93293.00
	Python	283.94	84613.00
Essential	Java	302.10	90027.00
	Python	294.90	87879.00

A Mann-Whitney U test revealed significant differences between the programming corpus of Java and Python, for tokens (U =26190.50; Z = -8.664; p<.05), LOC (U = 21353.50; Z = -10.968, p < .05), CC (U = 39784.50; Z =84335.50; p < .05) and SCC (U = 40062, Z = -2.072, p < .05). However, no significant difference was observed for ECC (U =43328, Z =-0.521, p = 0.602).

Boxplots are constructed and shown in Figure 3 for a comparative view of tokens, lines of code, cyclomatic complexity, strict cyclomatic complexity, and essential cyclomatic complexity.

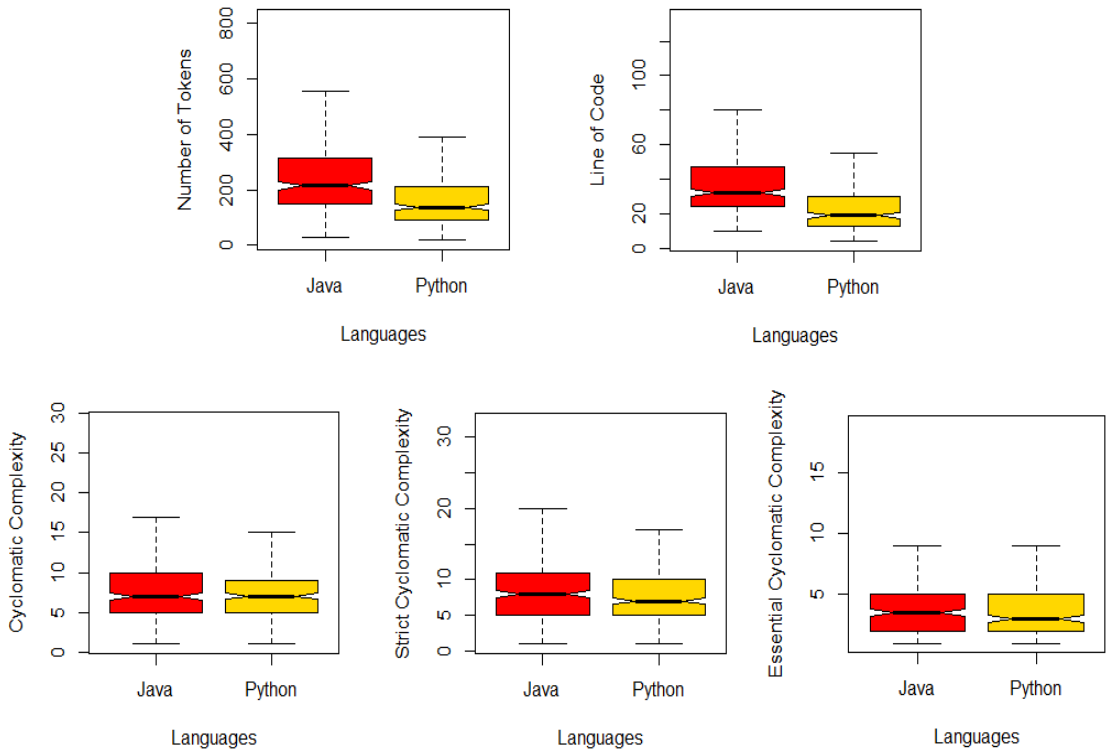

Figure 3. Boxplots of Results

In the whiskers of the boxplots of Java and Python, there is a significant difference. Similarly, the inequalities in the quartile groups of both languages indicate that the Java and Python corpuses for the same class of algorithms differ significantly in terms of tokens, line of code, cyclomatic difficulty, strict cyclomatic complexity, and essential cyclomatic complexity.

The overall study identified that, in comparison to Java, Python requires fewer tokens in the construction of programs based on modern algorithms. A noteworthy difference between Java and Python in respect of lines of code identified that Python strongly supports the development of programs with fewer lines of code than Java. Similarly, while comparing the cyclomatic complexity, strict cyclomatic complexity, and essential cyclomatic complexity of Java and Python programs, it was discovered that Python programs have less complexity than Java programs.

Java is one of the most widely used programming languages in the world today. The language has been used for many years. Many experts consider Java to be one of the most powerful programming languages ever devised. It's the most popular programming language, and it's built for distributed Internet contexts. However, just as every coin has two sides, Java has its own set of limits and benefits. The syntax of Java is simple and less complicated than that of C and C++, because many of these languages' more complicated features, such as explicit pointers, storage classes, and operator overloading, are being eliminated from Java.

Java is an object-oriented programming language that allows developers to improve the code's flexibility and reusability. Due to its platform independence, Java is a portable language. Because Java code can run on any platform, it is portable and can be run on any platform. However, Java code is verbose, which means it contains a lot of words and long, complicated sentences that are difficult to create, read, and maintain. This increases the code's

complexity and reduces its readability. Java focuses on making things easier to handle, but it also needs to deal with unnecessarily complex scripts and lengthy explanations.

Python is one of the most dynamic and versatile programming languages available today. Since its introduction in the 1990s, Python has grown in popularity, and today thousands of people are learning this object-oriented programming language. Python is a dynamic typing language, and duck typing is used extensively in Python.

This study discovered Python applications with fewer tokens and lines of code, as well as less complexity, indicating that a programming language's syntax can have a major impact on program complexity. Python is found to be simpler than Java because its syntax is quite basic, and the learning curve is relatively gentle. Python is incredibly simple to learn and code, and the usage of indentation instead of curly brackets makes Python code very straightforward to understand.

The outcomes of this study will provide a useful dimension for researchers in analyzing other programming languages with the technique and methodology used in the article. The discovery could help instructors choose the correct language for programming and data structure classes. The study, however, has a number of limitations in its current form. The results are based on 298 algorithms implemented in Java and Python. As a result, the conclusions aren't applicable to other programming languages or algorithms.

CONCLUSION

Information and communication technology has become an essential element of human life, allowing people to do more in less time and at a lower cost. Computing is the backbone of ICT, while programming is the core domain of computer science. Thousands of programming languages have been developed since the beginning of the high-level programming paradigm. As a result, a variety of teaching methods and tools have been developed. However, programming is still a difficult task. The difficult and distinctive syntax of a language is a key barrier to learning programming. In this article, the impact of programming language syntax on the complexity of their programs is investigated. Cyclomatic complexity, Strict Cyclomatic Complexity, and Essential Cyclomatic Complexity are used to analyze the programming corpus of Java and Python. The study found that a programming language's syntax has a significant impact on program complexity. Python code is simpler and more concise than Java code. The study concluded that before selecting any language, it is necessary to quantitatively analyze the lexical structure, syntax, and semantics of a programming language, as this will reduce learning challenges and improve the performance of learners and novice programmers.

REFERENCES

[1] S. Gill, "Impacts of Computers on Today ' s Society," vol. 2, no. 1, pp. 173–178, 2015.

[2] M. F. Aqda, F. Hamidi, and M. Rahimi, "The comparative effect of computer-aided instruction and traditional teaching on student's creativity in math classes," *Procedia Comput. Sci.*, vol. 3, pp. 266–270, Jan. 2011, doi: 10.1016/J.PROCS.2010.12.045.

[3] D. Wilfling, A. Hinz, and J. Steinhäuser, "Big data analysis techniques to address polypharmacy in patients - A scoping review," *BMC Fam. Pract.*, vol. 21, no. 1, pp. 1–7, Sep. 2020, doi: 10.1186/S12875-020-01247-1/TABLES/3.

[4] A. K. Bansal, "Introduction to Programming Languages," *Introd. to Program. Lang.*, Dec. 2013, doi: 10.1201/B16258.

[5] B. Teufel, "Organization of Programming Languages," *Organ. Program. Lang.*, 1991,

doi: 10.1007/978-3-7091-9186-6.
[6] B. W. Kernighan and D. M. Ritchie, *The C programming language*, Second edition. Englewood Cliffs N.J.: Prentice Hall, 1988.
[7] W.-H. Steeb and F. Solms, "Applications in Administration," *C++ Program. with Appl. Adm. Financ. Stat.*, pp. 375–452, Feb. 2000, doi: 10.1142/9789812813343_0011.
[8] K. Sierra and B. Bates, "Head first Java," p. 688, 2005.
[9] D. S. McFarland, "JavaScript & jQuery: The Missing Manual," *O'Reilly*, p. 538, 2011.
[10] J. Mueller and L. Massaron, "Python for data science for dummies."
[11] "An Introduction to Programming Paradigms - GC Digital Fellows." https://digitalfellows.commons.gc.cuny.edu/2018/03/12/an-introduction-to-programming-paradigms/ (accessed Jul. 01, 2022).
[12] A. H. Watson, D. R. Wallace, T. J. McCabe, and National Institute of Standards and Technology (U.S.), "Structured testing : a testing methodology using the cyclomatic complexity metric," p. 113, Accessed: Jul. 01, 2022. [Online]. Available: https://books.google.com/books/about/Structured_Testing.html?id=vr2OpwAACAAJ.
[13] M. Lopez, "An Analysis of the Mc Cabe Cyclomatic Complexity Number."
[14] "Practical Software Testing," *Pract. Softw. Test.*, 2003, doi: 10.1007/B97392.
[15] M. D. S. Nesc *et al.*, "Cyclomatic Complexity and Basis Path Testing Study," no. December 2020.
[16] F. Murtadho, D. W. Sudiharto, C. W. Wijiutomo, and E. Ariyanto, "Design and Implementation of Smart Advertisement Display Board Prototype," *Proc. - 2019 Int. Semin. Appl. Technol. Inf. Commun. Ind. 4.0 Retrosp. Prospect. Challenges, iSemantic 2019*, pp. 246–250, Sep. 2019, doi: 10.1109/ISEMANTIC.2019.8884289.
[17] P. Gsellmann, M. Melik-Merkumians, and G. Schitter, "Comparison of Code Measures of IEC 61131-3 and 61499 Standards for Typical Automation Applications," *IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA*, vol. 2018-September, pp. 1047–1050, Oct. 2018, doi: 10.1109/ETFA.2018.8502464.
[18] D. F. Evans, Benjamin J., "Java in a Nutshell, Seventh Edition," 2018, Accessed: Jul. 02, 2022. [Online]. Available: http://repositorio.unan.edu.ni/2986/1/5624.pdf.
[19] "8 Best Popular Projects on Java. Today, according to reliable publicly… | by Andrew Suschevich | Javarevisited | Medium." https://medium.com/javarevisited/8-best-popular-projects-on-java-e1a663ab3cc1 (accessed Jul. 02, 2022).
[20] and J. Z. B. d. Böck, A. Grooff, A. Dhiratara, *Final_Chapter.html*. 2018.
[21] "See top programming languages big companies prefer." https://flyaps.com/blog/top-10-coding-languages-used-by-global-companies/ (accessed Jul. 02, 2022).
[22] "12 Top Python App Examples from Top-notch Companies." https://www.netguru.com/blog/python-app-examples (accessed Jul. 02, 2022).
[23] B. M. O. . S. K. A., "A Comparative Analysis of Complexity of C++ and Python Programming Languages Using Multi-Paradigm Complexity Metric (MCM)," *Int. J. Sci. Res.*, vol. 8, no. 1, pp. 1832–1837, 2019, [Online]. Available: https://www.ijsr.net/archive/v8i1/ART20194760.pdf.
[24] A. A. Khan,et al., "Comparison of Software Complexity Metrics," *Int. J. Comput. Netw. Technol.*, vol. 4, no. 1, pp. 19–26, 2016, doi: 10.12785/ijcnt/040103.

[25] B. Muriana and O. Paul Onuh, "Comparison of Software Complexity of Search Algorithm Using Code Based Complexity Metrics," *Int. J. Eng. Appl. Sci. Technol.*, vol. 6, no. 5, pp. 24–29, 2021, doi: 10.33564/ijeast.2021.v06i05.003.

[26] P. Costanza, C. Herzeel, and W. Verachtert, "Comparing Ease of Programming in C++, Go, and Java for Implementing aNext-Generation Sequencing Tool," *Evol. Bioinform. Online*, vol. 15, Aug. 2019, doi: 10.1177/1176934319869015.

[27] P. Emmerich *et al.*, "The Case for Writing Network Drivers in High-Level Programming Languages," *2019 ACM/IEEE Symp. Archit. Netw. Commun. Syst. ANCS 2019*, Sep. 2019, doi: 10.48550/arxiv.1909.06344.

[28] C. Chen, "An Empirical Investigation of Correlation between Code Complexity and Bugs," Dec. 2019, doi: 10.48550/arxiv.1912.01142.

[29] N. Qamar and A. A. Malik, "Impact of Design Patterns on Software Complexity and Size," *Mehran Univ. Res. J. Eng. Technol.*, vol. 39, no. 2, pp. 342–352, Apr. 2020, doi: 10.22581/MUET1982.2002.10.

[30] E. McDaid and S. McDaid, "Quantifying the Impact on Software Complexity of Composable Inductive Programming using Zoea," *arXiv*, pp. 1–8, 2020.

[31] K. C. Louden and K. A. Lambert, *Programming languages : principles and practice*, 3rd ed. Boston MA: Course Technology/Cengage Learning, 2012.

[32] R. Zakaria, *Big Data and Machine Learning: How Using KNN Algorithms Can Help to Predict Employee Attendance*. 2020.