

Zahin- R,C

Kinjol-Java

Ishrat- Python

Sabbir- Javascript

Bitan- C++

1. Research on your language based on Readability, Writability, Reliability
2. Write your findings in this doc. Each language has a dedicated page to it
3. Include the references of each finding. Preferred that you used <https://www.citethisforme.com/> in IEEE format so that we can just copy paste the references in the paper. If that’s a hassle no worries we can do this later
4. It’s preferred to write the findings in your own words so that we can just copy paste them in the report. However, if you just copy and paste it here, remember to paraphrase it in the actual report!

We will be researching till 22nd December (1st update deadline). We will show this doc to sir as our progress. From 22nd Dec to 5th Jan (2nd update deadline), we will make the first draft of the paper and show it to sir.

# General findings (Zahin):

“There are 4 key items in keeping code readable- good names, commenting, formatting, and good design”[1]

“A beginner can be easily overwhelmed by code that is unintuitive or hard to read.

Consequently, languages with unusual syntax and a counter-intuitive control flow are

difficult to learn.” [2]

“Regularity means that the language should maintain consistency, both syntactically

and semantically. This may include invocations with unexpected side-effects or

violation of expected semantics. “ [2]

[1]"Introduction to C programming/Lectures/Readability - Wikiversity", *En.wikiversity.org*, 2020. [Online]. Available: https://en.wikiversity.org/wiki/Introduction\_to\_C\_programming/Lectures/Readability. [Accessed: 21- Dec- 2020].

[2]D. Gupta, "What is a Good First Programming Language?", *Crossroads- The ACM Student Magazine*.

# General findings (Kinjol):

Readability in software programming can be defined by the ease with which the software is read and understood.

Source code is read by both the compiler, which doesn't care about your writing style and people, who care enormously about your writing style.

ALL professional programmers insist that certain readability rules must be followed. Some of the most important are:

* **Indentation and spacing.** Statements that are contained within something else (a class, a method, a loop, etc) must be indented to the right. There are two common styles of indentation: Allman and K&R; use one of them.  
  **Whitespace.** Just as using correct paragraph and word spacing in natural languages makes them much easier to read, appropriate use of blank lines and internal spacing in a statement can make them much more readable. Some IDEs, such as NetBeans, will indent your program in either style.
* **Meaningful names.** Variables, methods, classes, etc must have names that mean something to the reader. Only use names like *a*, *b*, and *c* when there is no inherent meaning in a variable -- it's just an abstract number. See [Names](https://perso.ensta-paris.fr/~diam/java/online/notes-java/principles_and_practices/style/names.html) for a more detailed discussion.
* **Simplicity.** Always try to make the program as simple as possible. Resist the temptation to use be clever. For example, which of these is the clearest way to exchange two integer values a and b? The exclusive-or approach may even be faster in some circumstances, but it certainly isn't clearer to most programmers.
* **Conventions** are important to follow. There are standard naming, formatting, documentation, and usage conventions for Java. the organization you work for may have additional conventions. The use of these conventions makes program source code much easier for others to read.] => Combine these with Java

Orthogonality is one of the most important properties that can help make even complex designs compact. In a purely orthogonal design, operations do not have side effects; each action (whether it's an API call, a macro invocation, or a language operation) changes just one thing without affecting others. There is one and only one way to change each property of whatever system you are controlling.

# C (Zahin)

## Findings:

Readability:

Commenting- C allows single line AND multi line commenting

“Readable Syntax. The syntax of the language should be readable and consistent . The regular cases of errors are discovered in programs only because the programmer does not

understand code written by others due to its poor readability. Both beginner and experienced programmers, take advantage of good readability. In order to evaluate the readability of a language we use the following three parameters: i) Identifier’s name should neither be

length dependent, nor declared implicitly; ii) Consistent compound statement; iii) Meaning of constructs is not context dependent.” [3]

C supports the identifier length condition, but does not support the other 2.

“Another consideration for the evaluation of readability is that the forms and meaning of construct should not be appearance or context dependent C, C++, and

Fortran do not conform to this requirement. For example, in C static keyword has different meaning if declared inside and outside function.” [3]

Writability:

“The most common complaint against use of the C language is pointers. The fact that

pointers are required even for very basic things (such as implementing a linked list)

coupled with the fact that pointers are not easy to teach or to understand adds to the

level of confusion. C also faces a lot of criticism for poor support for data encapsulation

and information hiding” [2]

“Enforceability of Good Habits. A good FPL should enforce programmers to write clean and consistent code. Good program writing style is based on clarity and readability, and these habits

should be encouraged from the beginning. A good language should not allow:

1. Coercion with demotion (narrowing).

2. Expression side effects

3. Intermixing of arithmetic, logical and relational operators in Boolean expressions

4. Unconventional operator usage and overloading

5. Scope overriding-meaning of some constructs are context dependent such as static” [3]

C allows all these conditions. [3]

“Quality Coding Standards. The main objective of the coding standard is maintainability. Other important things that relate to the strength of the quality standards include simplicity,

consistency, portability, extensibility, clarity, safety, and correctness. Stroustrup states that production of quality code should be elevated to a central role in software development.

When in doubt, the programmer should endeavor for clarity rather than efficiency .The style of writing directly impacts the readability and understandability of the end product . By enforcing languages to implement the coding standards we can save cost of code review and minimize the human dependency and obviously minimize the possibility of bad coding practices and explicitly improve the readability of our source code. Indentation, comments, braces, naming conventions, and parentheses are most commonly considered as quality coding standard attributes” [3]

“In our evaluation we rate the languages higher if they facilitate the programmers with end-of-line, documentation, and mega comments, while we denounce the usage of block comments in a language.

Based on the above discussion and supported code listings we conclude that the languages C, Modula-2, and Pascal partially support the comments. The reason is that C supports mega

comment, but also has notorious block comment. “ [3]

Reliability:

if(x = 1) {

// do something;

}

“The above condition will always evaluate to true, irrespective of the value of x! C has

the uncanny ability to typecast almost any data type into an integer, without notifying

the programmer in most cases, and so the above error goes unnoticed. Strongly typed

languages like JAVA enforce that only a Boolean value is provided, thus avoiding this

problem. Equally notorious is the default prototype for undeclared functions, which

assumes the data type of parameters and return value to be integer, unless specified

otherwise. This has been corrected in ANSI C.

Another aspect of consistency comes with regard to data types. Taking another

example from C, on some platforms, a C int will take 4 bytes, while on some it will

take two bytes. On the other hand, JAVA guarantees that data types behave exactly

identically on all platforms” [2]

“Variable Data Types

Beginners must navigate numerous (often redundant) data types. In fact, many

experienced programmers also have difficulty identifying the specific data type for a

particular application domain, and usually end up using the most general data type.

This problem is most common with the design and implementation of basic numerical

and character string data types. An example is the different floating point types in C:

float and double. The float type is notorious because of its single precision overflow

behavior, so most people end up using double even when double precision is not

required. As another example, consider that C/C++ have 32 distinct numerical data

types, and the size of these depends on the underlying hardware as well! The standard

int type, for instance, varies from 16 to 32 bits depending on the machine and

runtime implementation. Clearly, when using numerical data types, it is often required

that the user be aware of the internal representation used and its limitations. For

instance, the following C code never terminates:

int main()

{

for(double i = 0;i != 10;i += .1)

cout << i << endl;

return 0;

}

The binary representation of 0.1 is non-terminating, so the increment is not in precise

steps of 0.1 so the loop condition is never satisfied.” [2]

“Strongly Typed. Strongly typed means all type checking issues are resolved either at compile time, or at run time. It ensures that no unexpected results occur at runtime due to type mismatching. Thus, it must be checked by the compiler, or by the runtime system, and no automatic conversions should be allowed. The only possible way for type conversion is

explicit type casting by the programmer. Strongly typed languages are more reliable, and are easy to program and debug by novices. The concept of strongly type is usually implemented in two forms:dynamic strongly typed, and static strongly typed.

In static strongly typed languages the variables are explicitly declared and the type binding takes place at compile time. Similarly, all errors related to type are detected at compile time.

The type of a variable cannot be changed after its declaration. Fortran, Ada, C/C++, Pascal, Modula-2, Java, and C# belong to static strongly typed category.“ [3]

## References:

[2]D. Gupta, "What is a Good First Programming Language?", *Crossroads- The ACM Student Magazine*.

[3]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. Available: 10.1371/journal.pone.0088941.

# Javascript (Sabbir)

## Findings:

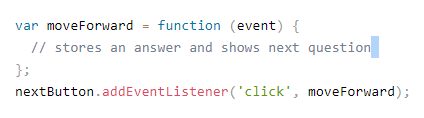
**Reliability:**

JavaScript is one the most famous programming languages. It is simple and abundance makes it easy for beginners. It’s large tools and libraries make it more reliable. The code is not complex, it's very simple. It has a maintainable code base which makes it reliable also. From a browser implementation standpoint, it is not so reliable because different browsers behave differently, and many users turn it off. JavaScript creates an enhancement layer and not base of any core functionality to make it usable.

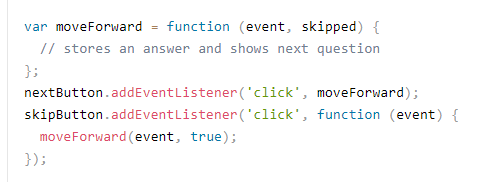
**Readability:**

JavaScript has been evolving into a more readable language. Because of non-strict nature we can write the same thing in different ways. The same thing could be achieved with creating simple helpers that literally explain what the code is doing. We can see some examples.

The user presses the next button, the application gets the input, sends it to the server and shows another question. The code is also pretty simple [3].



Here is another example we can see we have a function for moving forward with a skipped flag. The next button is calling the function above. There is also a skip button that has a handler attached [3]. It’s easily readable that 1st skipButton and then it will jump to moveForward.



JavaScript is very readable and we can understand what code is doing. It is very helpful for beginners to understand the code's purpose.

**Writability:**

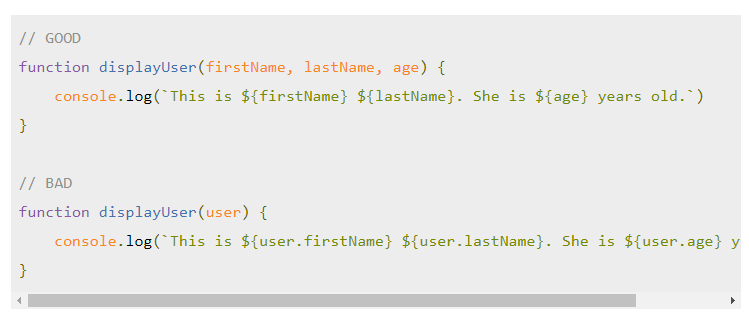
JavaScript has its origins in the early web. Starting out as a scripting language, it has now evolved into a fully-fledged programming language with support for server-side execution. [4]

There are some steps that can solve the problem errors.

1. **Isolate your code:** Always keep your Codebase clean and readable. Function should have one purpose.

### 2. **Modularization:** You use multiple functions in one module if those functions do in a similar way.

3. **Prefer multiple parameters over single object parameters:** Use multiple parameters when you declare a function.

****

### 4. **Destructuring:** Destructuring is a nice tool that was introduced with ES6. It lets you grab specific fields from an object and assign it to a variable immediately. You can use this for any kind of object or module. [4]

### 

1. **Use default values:** Default values for destructuring or even basic function parameters are very useful. Firstly, they give you an example of what value you can pass to the function. Secondly, you can indicate which values are required and which are not. [4]

## References:

[1]<https://medium.com/better-programming/js-reliable-fdea261012ee>

[2] <https://www.tek-tips.com/viewthread.cfm?qid=494170>

[3]<https://krasimirtsonev.com/blog/article/rethinking-javaScript-readability>

[4] [12 tips for writing clean and scalable JavaScript - LogRocket Blog](https://blog.logrocket.com/12-tips-for-writing-clean-and-scalable-javascript-3ffe30abfe20/?fbclid=IwAR2UZMpPFgF3V9qqXX24Jb5Rcbc9_XxoywH1LA8TIiHTsTxCkhAo1gRUQZk)

# Java (Kinjol)

## Findings:

## Readability

Note from a guy in Stack overflow: In Java you'd have to evaluate for example if there is a combination of keywords/constructs that could affect each other when used simultaneously on an identifier. For example, when applying public and static to a method, they do not interfere with each other, so these two are orthogonal (no side effects besides what the keyword is intended to do)

You'd have to do that to all its features to prove the orthogonality. That is one way to go about it. I do not think there exists a clear cut *is* or *is not* orthogonal in this matter either.

## Writability

Java is more readable, easier to write, and far more reliable than C. Java’s use of object-oriented design brings the language more closely in tune with adjective-noun syntax in spoken language. Java and C both have similar data types and structures. However, C has “two kinds of structure data types, arrays and records … records can be returned from functions but arrays cannot.” (11) This is an orthogonal discrepancy, and diminishes the readability. Java and C both enjoy control statements such as the “for” and “while” loop, which increase readability. Java and C are similar in how they read, but Java is a bit more orthogonal and the object-orientation makes it far more readable.

Java is a little easier to write than C. Both share common writability features. For example, both allow abstraction through functions. One need only write a function once and then call it multiple times. Both allow expressivity by conveniently allowing multiple ways to do things. “The notation count ++ is more convenient and shorter than count = count +1.” (17). Sometimes “for” loops are more convenient than “while” loops even though both can be written to do the same thing. Java does not have the confusing pointer syntax that C has. Also in C, one must allocate and deallocate memory with \_alloc() functions that really make no sense. C++ remedied this with the “new” and “delete” keywords, but Java is easier to write than either C or C++ due to garbage collection.

Java is far more reliable than C. First, Java has strong type checking, thereby identifying any mistypes before it creates logical errors. Failure to type check "has led to countless program errors ... in the original C language." (17) Java has excellent exception handling, which “take(s) corrective measures, and then continue(s).” (17) While C++ has exception handling, C does not, which greatly reduces reliability. Part of C’s power is to directly access memory with pointers. However, the same memory location can be accessed with different pointers, thereby creating aliasing, which is another detriment to reliability.

It is easy to see that Java has many advantages over C. First, Java is more readable. It has all of the control statements, data types and simplicity of C without much the confusing syntax, making it more readable. Both Java and C share similar grammar, although C’s memory allocation and pointers make it less writable than Java. Java is far superior to C in reliability because of its exception handling, type checking and lack of pointer-aliasing. So why would anyone use C over Java? See the next question

## References:

[1]"Readability - an overview | ScienceDirect Topics", *Sciencedirect.com*, 2020. [Online]. Available: https://www.sciencedirect.com/topics/engineering/readability. [Accessed: 22- Dec- 2020]

Link: [Click Me](https://www.sciencedirect.com/topics/engineering/readability#:~:text=Readability%20in%20software%20programming%20can,software%20is%20read%20and%20understood.&text=Simplicity%20in%20logic%2C%20conditional%20statements,code%20all%20help%20with%20readability.)

[2]F. (www.fredswartz.com), "Java: Readability", *Perso.ensta-paris.fr*, 2020. [Online]. Available: https://perso.ensta-paris.fr/~diam/java/online/notes-java/principles\_and\_practices/style/readability.html. [Accessed: 22- Dec- 2020]

Link: [Click Me](https://perso.ensta-paris.fr/~diam/java/online/notes-java/principles_and_practices/style/readability.html)

[3]"Brian Buckley", *B.web.umkc.edu*, 2020. [Online]. Available: http://b.web.umkc.edu/buckleyb/cs\_441.htm. [Accessed: 22- Dec- 2020]

Link: [Click Me](http://b.web.umkc.edu/buckleyb/cs_441.htm)

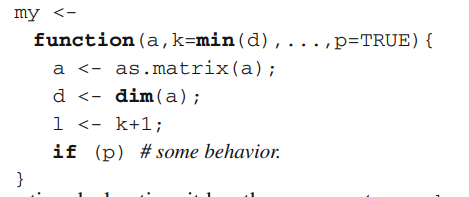
# R (Zahin)

## Findings:

Readability:

Comments- R only supports in line comments

Writability:



“. R gives programmers much freedom in how functions are defined and called. Function declarations can specify default values and a variable number of parameters. Function calls can pass parameters by position or by name, and omit parameters with default values. Consider the my function declaration, it has three parameters, a, k, and p. The ellipsis specifies a variable number of parameters which can be accessed by the array notation or passed on to another function in bulk. This function can be called in different ways, for instance, my(x); my(x, y); my(x, y, z); my(k=y, x, z, p=FALSE) A valid call must have at least one, positional, parameter. Any argument with a default value may be omitted. Any argument may be passed by name, in which case the order in which it appears is irrelevant. Arguments occurring after an ellipsis must be passed by name. The default values of arguments can be expressions; they are boxed into promises and evaluated within the same environment as the body of the function. This means that they can use internal variables in the function’s body. So min(d) above refers to d, which is only created during evaluation of the function. As k is always forced after d has been defined, the function will work as intended. But this shows that code can be sensitive to the order of evaluation, which can lead to subtle bugs.” [4]

Reliability:

“Platform Independent

R is a platform-independent language or cross-platform programming language which means its code can run on all operating systems. R enables programmers to develop software for several competing platforms by writing a program only once. R can run quite easily on Windows, Linux, and Mac.” [5]

“R lacks basic security. It is an essential part of most programming languages such as Python. Because of this, there are many restrictions with R as it cannot be embedded in a web-application.”[5]

## References:

[4]F. Morandat, B. Hill, L. Osvald and J. Vitek, "Evaluating the Design of the R Language", *ECOOP 2012 – Object-Oriented Programming*, pp. 104-131, 2012. Available: 10.1007/978-3-642-31057-7\_6 [Accessed 22 December 2020].

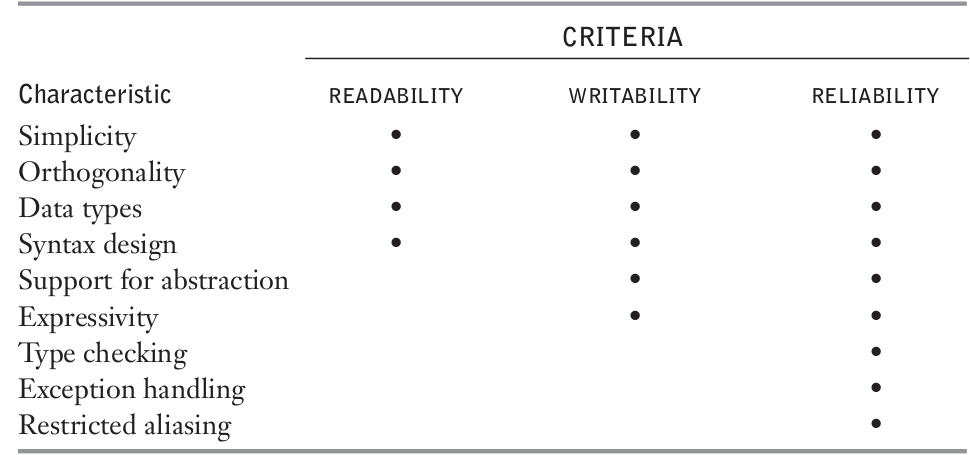
[5]"R Advantages and Disadvantages - javatpoint", *www.javatpoint.com*, 2020. [Online]. Available: https://www.javatpoint.com/r-advantages-and-disadvantages. [Accessed: 22- Dec- 2020].

# Python (Ishrat)

## Findings:

Python’s design principle emphasizes on elegance, readability and writability first and foremost.

### **Language Evaluation Criteria to be discussed [2]**



### **Criteria: Readability & Writability**

1. Simplicity

1. Small number of basic components

2. One syntax :: one meaning

2. Orthogonality

1. Any composition of basic primitives is allowed.

2. Need small set of primitives and ways to combine them

3. Violation: Out parameters for functions

4. Violation: Return types for functions

5. Violation: Functions as first-class objects

6. Non-orthogonal often to simplify implementation

### **Criteria: Readability & Writability (Cont.)**

1. Control Statements/Constructs

Which control mechanisms to include?

2. Data Types

\* Primitive (e.g. boolean, char, unsigned)

\* Structured

\* Provide abstraction.

3. Syntax

\* Identifier length, reserved words, layout, etc.

4. Abstraction

\* Hide details to manage complexity!

\* Process abstraction

\* Data abstraction

### **Criteria: Reliability**

1. Performs to specifications under all conditions

2. Impacted by:

1. Type checking

1. compile-time

2. runtime

3. none

2. Exception handling

3. Aliasing (i.e. Two or more names for same memory cell), for example:

p,q: access Integer;

p := new Integer;

q := p;

**Readability:**

“ “Python’s simplicity is particularly helpful in reading code—yours or someone else’s. Because Python code has fewer lines and mimics English, reviewing it takes a lot less time. This is a major benefit.

Reducing the time you need to spend on code review is invaluable, since the productivity of your developers should be your top priority. “ “

Using indentation in block comments seems natural, making everything smooth and pretty. It is, in fact, very easy to read and understand, unless you have to nest a bunch of loops and ifs. Even then, however, simple, pragmatic line-by-line scrutiny makes everything clear.[1]

Fewer syntax rules reduces confusion making it understandable in plain English.

Multi-paradigm programming language so programmers can use different programming styles when writing (functional, object-oriented, imperative)

Use of explicit code, one line per statement, limits in line length [3]

Python is known by forcing the programmers to use indentation, which makes the pieces of code conventionally more readable. Of course, other factors would affect this gained readability advantage of indentation. A code without indentation is known to be hard to follow. In other words, the reader would have to do more work to look at the code very closely in order to spot the start and the end of the statements, where an if statement ends, or what statements are inside a for loop. The off-side rule to a language like Python makes the reader not be concerned with spotting out the blocks since they are represented graphically. The brackets used by curly bracket programming languages are useful to define the beginning and ending of blocks, but the nature of these languages does not impose the use of white spaces for a well structured code. [6]

“” For Python, the entry point is famously low, which is why it’s perfect for newbies and junior developers. The language is extremely user-friendly.

Conversely, Java has a high entry point with a clear learning curve. Learning how to write in Java—not to mention mastering it—is a significant time investment. “” THis affects both readability and writability. The learning curve is much lower for python.

**Writability:**

Python is an orthogonal language meaning basic building blocks can be combined in minimal ways to build the language’s data structures. Everything in python, including functions and classes, is an object. This allows higher order functions that take other methods and classes as arguments. It enables library designers to provide elegant composable APIs. For eg: in scikit learn library many ML tasks are expressible as sequences or combinations of transformations to data. These methods or arguments are programmable from existing building blocks. [4]

Abstraction means hiding the complexity and only showing the essential features of the object. Abstraction in Python is achieved by using abstract classes and interfaces. Python allows high level abstraction which means more flexibility and robustness. However this comes at a cost of speed[5]

**Reliability:**

For Python and Perl, the relative variability in memory consumption tends to be much smaller than for C and, in particular, C++. • The typical script program consumes about twice as much memory as does a C or C++ program.[7]

One of the biggest criticisms of Python is the runtime, relatively slow when compared to other languages. However, there’s a workaround to this specific challenge.

When performance takes priority, Python gives you the ability to integrate other, higher-performing languages into your code. Cython is a good example of such a solution. It optimizes your speed without forcing you to rewrite your entire code base from scratch. [8]

“”

Python differs dramatically to the traditional paradigms of C, Java, and other traditional languages. The Python way to handle processes with exceptions is to just go for it and handle the exceptions if they come up.

In C, you might write lines and lines of code to check for all preconditions before writing to a file — such as, does the file exist, does the process have write permissions to the file, etc. It’d be bad practice in C to disregard all these cases and just write directly to the file.

In Python, it’s quite the opposite. In fact, it’s considerably more efficient in some cases to try first and handle exceptions later. In the above example, while it’s entirely possible the file permissions are wrong (or some other issue), it’s an edge case. A majority of the time, you’ll be writing to a file that exists and you have permission to access, so why waste valuable processing power checking for file permissions?

The Pythonic way to handle this is to wrap the write in a try-except and catch only the specific exceptions/edge cases that are likely to come up — and to handle them appropriately. In most cases, the write will succeed, and any logic in the catch block will be missed entirely. In the case where something does go wrong, we still handle it. “” [9]

Specific error message displayed along with line number where the exception was thrown. Not available in java or c.

## References:

## [1]<https://ininet.org/csc-415-programming-languages-jonathan-huelman.html>

## [2] https://www.radford.edu/~nokie/classes/380/intro/intro3.html

## [3]<https://stackabuse.com/introduction-to-the-python-coding-style/>

## [4] <https://www.robosoftin.com/blog/why-python-for-machine-learning>

[5] <https://prateekvjoshi.com/2014/11/23/the-cost-of-abstraction-in-python/>

[6] <https://sci-hub.se/10.1109/CSIT.2016.7549476>

[7] <https://ieeexplore.ieee.org/abstract/document/876288>

[8] <https://www.stxnext.com/python-vs-other-programming-languages/>

[9]https://medium.com/better-programming/a-comprehensive-guide-to-handling-exceptions-in-python-7175f0ce81f7

## 

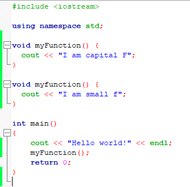
## C++ (Bitan)

## Findings:

There is one thing certain about a program and that is the programmer is going to change and update the code of the program eventually. A good program always evolves with time. That is why developers prefer a consistent and reliable language that will evolve with time as well to stay relevant in the future. C++ is a programming language that has proven itself to be adaptive for a long time.

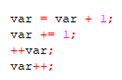
**Readability:**

C++ is not a very reading friendly language for several reasons. It is a case sensitive language that also supports operator overloading. Which is a deadly combination for reading the code.



Another example, ampersand (&) in unary (&a) and binary operator (a & b) has totally different meanings in C++. To avoid confusion it is important for the programmer to implement good commenting.[2]

Another reason is that it has feature multiplicity. Which means there are several different ways to do one thing.



Also if a program has a rich library of reserved words like the ADA programming language, it is known to be more readable than when a programming language has fewer reserved words as in C++. Overall complexity of the language causes it to be less readable as well.[1]

**Writability:**

The reason that makes C++ not a reader friendly language also makes the language harder to write. Because a programmer has to keep track of the code he has written and have to keep in mind not to have many similar attributes. The most important feature that makes a language less writable is its use of abstract data. With the use of data abstraction a programming language can easily become difficult to write. C++ does however have some characteristics that make it a little more writable than some languages. For example, since C++ has a small library of reserved words, there aren’t as many words for the programmer to remember. Overall, C++ is a complex language and is not known for its ease of writability.[1]

**Reliability:**

A programming language is reliable when it does what it is supposed to do at all times. Syntax design, type checking, exception handling, aliasing, readability and writability are major factors that affect a program’s reliability. C++ is a dominant object oriented programming language because if programmers use it carefully C++ can be reliable. Although C++ on flexibility and efficiency more than reliability. C++ is a very flexible language. It includes all features as well as bad features from c programming language.[2] Pointer is a powerful feature that programmers can use while using C++. But unless the programmer uses this feature carefully memory leakage and dangling pointer issues can occur which can affect reliability greatly.[1]

Exception handling is a great feature that improves the reliability of the language. But aliasing which means two or more distinct referencing names or methods for the same memory cell, can hamper programs reliability. For example, consider the statements int x = 7 and int \*y = &x, both refer to the same memory cell. C++ allows aliasing because its designer trades a decrease in reliability for an increase in efficiency and flexibility.[1]

## References:

1. <http://campus.murraystate.edu/academic/faculty/wlyle/415/2011/Hall.docx>
2. <http://campus.murraystate.edu/academic/faculty/wlyle/415/2014/Peng.pdf>