Decoding Coding:

A Study of Programming Languages

William Jessop

Frontiers of Science Institute

**Abstract**

 This project studied three separate coding languages and developed a pseudo-code. Python 3.6 was developed in the IDE, IDLE. Java was developed in the IDE, Eclipse Oxygen. C++ was developed in the IDE, CLion. These languages were chosen based on their regular use in the programming field. After the languages were studied a pseudo-code was developed to analyze the underling logic chains that are found in all programming languages.

*Keywords:* Python, Java, C++, Assembly Language, Machine language, Syntax, Pseudo-code, Compiler

**Table of Contents**

**Abstract……………………………………………………..............2**

**Introduction……………………………………………………..….4**

**Review of Literature……………………………………….............5**

*Figure 1: The operation A×B in Machine Code……………...5*

*Figure 2: The operation A×B in Assembly Code………….....6*

**Methods and Materials…………………………………................8**

**Results and Conclusion…………………………………................9**

*Figure 3: The Ham Sandwich program in Python Code…..10*

*Figure 4: The Ham Sandwich program in Java Code……..11*

*Figure 5: The Ham Sandwich program in C++ Code…….13*

*Figure 6: The Ham Sandwich program in Pseudo-code….14*

**Reference…………………………………………………….…...15**

Decoding Coding: A Study of Programming Language

 A programming language is a system of commands that are translated through a compiler to give a computer specific instructions to execute. Today there are many different languages that are used to accomplish similar goals. These languages are part of a group known as advanced languages (Zelkowitz 2014). Some examples would include the languages C++, Java, and Python. These languages are all capable of doing pretty much the same thing; they just do it slightly different ways.

In computer programming, syntax is the set of rules by which each programming language is bound to determine what is the correct way to interpret commands. Each language has a unique syntax. One could look at it much in the same way that spoken languages vary. Speaking the sentence “It is a nice day outside” will sound different in French and German but they still have the same meaning. When computers were first used, syntax was far too simplistic, in that there was not any. Programming was strictly made up of long sequences of 1’s and 0’s used to execute commands (Glass 2014). Today, programming languages have a much more complex syntax. They have a look and feel much like English, using words and statements to run specific commands.

All the languages discussed above are fundamentally similar. They all eventually lead back to the language known as the Machine Language, the sequences of ones and zeros discussed above. There are certain fundamentals that are expressed in all advanced languages of code. This project will identify the similarities common to all programming languages and develop a pseudo-code, or template code, that can accurately describe the processes observed in all of them.

**Review of Literature**

A program, by definition, is “a set of miniscule tasks, in a framework of logic, such that the computer knows exactly what to do and when to do it” (Glass, 2014). Programming languages, henceforth, are specifically designed to create programs. More simply put, a program is a set of instructions within a programming language, written so that a computer can execute them. These languages are all ways to simplify Machine Language which is discussed below.

Machine Language was the first major revolution in programming languages. It consisted of long sequences of ones and zeros corresponding to values, locations, and operations that a machine can use to preform functions (Glass, 2014). With this being the case, it was still quite difficult for a person to write a program to perform the simple operation of A × B using only

THE OPERATION A×B IN MACHINE CODE



*Figure 1*. An example of Machine code. This code executes the operation A×B. (Kumar 2016)

long lines of numbers. (In Figure 1) In response to this complexity, programming languages evolved.

The second computer language evolution was to a format known as Machine Assembler, also known as the Assembly Language. This major shift provided several benefits. Programmers were no longer required to write operations as sequences of numbers. The assembler would automatically translate the assembly code, written by a human, into commands that a machine could execute. (In Figure 2) The Assembler would convert the function ADD into the machine code “101” which is the equivalent (Zelkowitz, 2014). This began the long journey of bridging the gap between what is easily read by computers and what is easily written by humans.

THE OPERATION A×B IN ASSEMBLY CODE



*Figure 2.* An example of Assembly code. This code executes the operation A×B. (Kopella 2013)

The third and final evolution is what is known as advanced languages. These are higher level languages such as Python, Java, and C++. Language developers began adapting more advanced concepts into the code, such as IF statements, WHILE loops, and functions. With these tools came a program known as a compiler. Unlike an assembler, a compiler must analyze the code to generate the correct lines of Machine Language. A compiler is essentially a translator for advanced coding languages which takes complex code and converts into machine code, so that a computer can execute it (Zelkowitz, 2014). This process of compilation is what has allowed us to do amazing things with computers.

Computers are very rigid in their logic, and therefore they must have a structure in which to operate (Lee, 2013). Computers are designed to interpret commands literally, and are incapable of making assumptions or using common sense. It is quite difficult to write a program to tell the computer every possible task to do, and how to do it correctly. This is the reason programming languages were created, to provide a structure that is not too difficult for a person understand. This structure is then analyzed by a computer to execute what the programmer wants the program to do. Programs are written similar to English, left-to-right top-to-bottom so that a person can read, write and understand it.

Today we use many complex programs in our daily lives. Programs allow Google to create Chrome, Microsoft to create the Office Suite and Texas Instruments to create Computer Algebra System. We can now use computers to do complex tasks such as preforming algebra symbolically (Sochacki, 2014). What has allowed this to become a reality is the language that it is written on. As programming languages have become more complex we have been able to perform tasks in fewer and fewer lines of code. This means that, thanks to modern programming, we are able to use computers for much more practical applications. Whether that means writing a paper or solving mathematical equations it is all possible. Modern coding has opened the door for us to use computers to a much fuller potential.

This research aims to illustrate the commonalities between the coding languages C++, Java and Python. These languages will be compared on the basis of syntax and function. This research will focus on the strengths and weaknesses of the varying languages, as well as the similarities and differences between them. This research hopes to generate a pseudo-code that can be used as a template for all languages studied.

**Methods and Materials**

This project involved implementing the same program using three different modern coding languages, to illustrate their similarities and differences. A Microsoft Surface Pro 3 was used for the majority of the programming. This is a very basic computer with four gigabytes of RAM and a 64-bit processor. In addition to the Surface, a portion of the coding was done on a cellular device, using an application called QPython3.

Each language was unique and therefore required its own compiler. This software is known as an integrated development environment or IDE. For the project, the language Python 3.6 used the IDE called IDLE. The language Python was focused on first because it is the simplest of the languages that were chosen. After Python, Java was researched. Java was chosen second due to its relatively common use. Java was coded using the IDE known as Eclipse with the version Oxygen. The final language that was focused on was C++. C++ is very similar to Java and is still commonly used. The language C++ was coded using Microsoft’s Visual Studio.

In this project, a specific task had to be selected as a standard for comparison. The task that was selected was a simple guessing game. This game was programed with two modes: a casual mode and a challenge mode. The casual mode makes use of a WHILE loop for the main code. Conversely the challenge mode makes use of a FOR loop in the main code. The WHILE loop repeats the code until a particular parameter is met. This was used to repeat the guessing mechanic until the answer was guessed. The FOR loop repeats the main code a set number of times. This made it easy to limit the number of guesses a user could make. Both games also make strong use of two more key concepts. Firstly, they use Input/Output systems to take information from the user and give information back. This is used to take the user’s guess. Secondly, they use IF statements. If statements are used to compare the user’s input to the answer. After each guess, the computer outputs information to the user to tell the user whether they guessed the answer, too high, or too low. This basic program demonstrates the main programing concepts, WHILE, IF, and FOR, along with user input.

**Results and Conclusion**

 Many discoveries were made when studying these languages. All three languages were observed as having strengths and weaknesses as well as underlying commonalities. Python as a language was found to be the simplest to learn, but due to this it loses the ability to be as specific as C++ or Java. Java as a language was found to be strong with compartmentalization and object-oriented programming, while lacking in the ability to easily take input. C++ as a language was found to be strong in being very specific, but was the most complex to write.

 Python was the simplest of the languages studied. This makes Python an excellent language for beginning programmers. It is easy to learn, allowing the programmer to focus on learning the concepts of computer logic instead of the language syntax. It allows anyone to start writing code very quickly and requires only the mastery of basic programming concepts. It takes inputs directly and has more intuitive commands. Figure 3 is an example of a Python program. This is a basic program that asks the user weather they prefer ham or turkey, giving a humorous result if neither input is entered correctly.

THE HAM SANDWHICH PROGRAM IN PYTHON CODE

*Figure 3*. An example of Python code. This code executes the ham sandwich program.

This program prompts a user for input and compares it with predetermined correct answers using an IF-ELIF-ELSE statement. If the input entered is the same as one of the answers, it will run the code in that part of the statement which prints the corresponding response. If the input does not match, it will run the else section and print a different response.

 Java was the next simplest language to learn. That being said, it is still fairly complex. This language was found to be useful for being able to reference external files in the same directory without needing to import them within the code. This can be an advantage for large projects with multiple programmers. Java was written to be completely object-oriented, so it is naturally more suited for that. This form of programming, as it may sound, consists of setting up code relating to objects and how they change. Although Java was found to be a strong language for object-oriented programming, it was seen to have its limitations in traditional programming due to the requirement of creating objects. The main difficulty for Java in traditional programming was user inputs, through an object called a scanner. In Java, a programmer must create a scanner as an object and call it to accept input. Figure 4 is an example of the same program as above written in Java instead of Python.

THE HAM SANDWHICH PROGRAM IN JAVA CODE



*Figure 4*. An example of Java code. This code executes the ham sandwich program

This program accomplishes the same task as the Python program in figure 3, but in a slightly different manner. Java does not process inputs the same way as Python. Instead of taking it directly, a scanner must be imported and initialized to take user input. After it receives the input it goes through a similar process of comparing it to find a match and print the corresponding answer. This process is called string comparison and is done with the .equals command instead of the double equals operator (= =) in Python. Java also has one more large difference form Python in how blocks of code are signified. Java uses braces to signify the beginning and end of a statement, whereas Python uses indentation.

 C++ was the most complex language of those chosen. This language has a reason for its complexity. What it lacks in simplicity, it makes up for in descriptiveness. C++ has a higher capacity for specificness in coding. It is very similar to Java in that it also uses braces to signify code blocks. Its first major difference however is that there are lines of code in C++ that do not make sense logically. In figure 5 these lines are line 2, 4, and 15. These lines do not make sense in terms of the logic however they do serve a purpose. These lines are required to run and simply C++ code but for the purposes of this project it is irrelevant to understand how they do that. Figure 5 is the C++ version of the example program that is seen in Python and Java above. C++ also requires more external libraries than the other languages. Generally, C++ has to import many more sets of functions before they can be used to accomplish a given task. This is done through the command #include as seen in line 1 of figure 5. This operation imports the library that is responsible for taking user input and giving output. After that library is imported it is possible to call the functions cin and cout. These functions are used to take user input and print output respectively. This program does the same thing as the Python program in figure 3 and the Java program in figure 4.

THE HAM SANDWHICH PROGRAM IN C++ CODE



*Figure 5.* An example of C++ code. This code executes the ham sandwich program.

 This project investigated the commonalities between languages, resulting in a pseudo-code that represents tasks accomplishable in other languages. The pseudo-code was written to avoid being in any specific syntax. Therefore, this code provides a base code as a reference for writing the program in any other language. Figure 6 below shows the sandwich meat program written in the pseudo-code, developed for this research. This program is not code that could be run, but rather provides a reference and description of the logic chains that go into making a program to request deli meat preferences.

THE HAM SANDWHICH PROGRAM IN PSUEDO-CODE



*Figure 6.* An example of pseudo-code. This code executes the ham sandwich program.

 Overall this project was fruitful. It concluded that of these languages Python is the simplest to learn, followed by Java and C++. It discovered various strengths and weaknesses of the languages studied. It found that Python is quick to learn but can be limited by its simplicity. It found that Java is strong for object-oriented programming but can be limited when it comes to traditional programming. It found that C++ can be very specific in its code but is more complex to write for that reason. It also developed a pseudo-code that can be applied to all three. This was a topical study and should be repeated in the future more in depth into traditional programming as well as into object-oriented and event-driven programming.

**Reference:**

Glass, Robert L. (2014). Software. In *AccessScience*. McGraw-Hill Education. https://doi.org/10.1036/1097-8542.757374

Kopella, G. (2013, November 15). An Assembly program to add two numbers present in (variables) memory locations and store the result in next (variable) memory location. Retrieved July 05, 2017, from http://cssimplified.com/computer-organisation-and-assembly-language-programming/an-assembly-program-to-add-two-numbers-present-in-variables-memory-locations-and-store-the-result-in-next-variable-memory-location

Kumar, P. (2016, September 01). Computer Languages. Retrieved July 05, 2017, from https://computerscience4beginners.wordpress.com/2016/08/31/computer-languages/

Lerner, K. L., & Lerner, B. W. (2013). Procedural Languages. Retrieved July 5, 2017, from http://go.galegroup.com/ps/retrieve.do?tabID=T003&searchId=R3&searchType=BasicSearchForm&userGroupName=uncol&inPS=true&prodId=GVRL&contentSet=GALE&docId=GALE|CX2761000135

Sochacki, James. (2014). Symbolic computing. In *AccessScience*. McGraw-Hill Education. https://doi.org/10.1036/1097-8542.757385

Zelkowitz, Marvin V. (2014). Programming languages. In *AccessScience*. McGraw-Hill Education. <https://doi.org/10.1036/1097-8542.547550>