**What to Choose, How to Choose:**

**A Comparative Analysis of Different Programming Languages**

A Research Paper Presented to

Prof. Anna Liza M. Barcelona

In Partial Fulfilment

of the Requirement for the Course

English 10

Proponents:

Jamie Anacleto

Alexandra Degala

Flouelin Malfarta

February 2013

**CHAPTER I**

Introduction

A. Statement of the Problem

Our research problem aims to compare and contrast four high-level programming languages according to five specified criteria.

Specifically, this study aims to answer the following questions:

1. What are the pros and cons of the following programming languages: Java, C++, Python, PHP?

2. How suitable and applicable is each programming language for a specific problem domain according to the following criteria: (a) versatility; (b) program length and; (c) programming effort?

3. What are the differences of the given languages in terms of memory consumption and run time efficiency?

B. Definition of Terms

A **programming language** is an artificial language designed to express computations that can be performed by a machine, particularly a computer (Oguntunde, 2012).

**High-level programming languages** allow the specification of a problem solution in terms closer to those used by human beings (King, 1999).

**Syntax** is the form or structure of expressions, statements, and program units (Sebesta, 2012).

**Semantics** is the meaning of those expressions, statements, and program units (Sebesta, 2012).

**Compiler** translates a program written in a high level language into the sequence of machine instructions required by a specific computer to accomplish the task (Ercolessi, 1997).

An **interpreter** will translate each line of source code into machine language and then execute that machine code directly (“Interpreter and Virtual Machines”, n.d.).

**Runtime** is when a program is running or being executable (Rouse, 2007).

**Machine code** statements are written in a binary-coded (low-level) computer language. Programmers usually write computer programs in a high-level language (such as FORTRAN or C), which a compiler program then translates into code for execution (“World Encyclopedia”, 2005).

**Software** refers to computer programs that are designed by a computer programmer or, more likely, a team of computer programmers, to perform a particular function (“World Encyclopedia”, 2005)

**Objects** have states and behaviors (“Java Encapsulation”, n.d.).

A **class** can be defined as a template/ blue print that describe the behaviors/states that object of its type support (“Java Encapsulation”, n.d.).

**Encapsulation** can be described as a protective barrier that prevents the code and data being randomly accessed by other code defined outside the class (“Java Encapsulation”, n.d.).

**Abstraction** refers to the ability to make a class abstract in OOP (“Java Encapsulation”, n.d.).

An **abstract** **class** is one that cannot be instantiated (“Java Encapsulation”, n.d.).

**Inheritance** can be defined as the process where one object acquires the properties of another (“Java Encapsulation”, n.d.).

**Polymorphism** is the ability of an object to take on many forms (“Java Encapsulation”, n.d.).

**Variables** are nothing but reserved memory locations to store values (“Python Programming”, n.d.). This means that when you create a variable you reserve some space in memory.

A **package** is a namespace that organizes a set of related classes and interfaces (“The Java Tutorials”, 1995).

A function is a piece of code which takes one more input in the form of parameter and does some processing and returns a value (“PHP Tutorial”, n.d.).

C. Hypotheses

The following hypotheses were formulated:

1. Python has the shortest program length due to the abundance of built-in functions available and thus has the smallest memory consumption.

2. Python has the shortest runtime because it has the shortest program length and Java’s runtime is the longest.

3. C++ is most versatile because it offers both procedural and object-oriented syntax.

D. Review of Related Literature

**Introduction**

This section will give a brief history on high-level languages, how they came about and how important they are in engineering software. This section will also discuss about the different categories of programming languages and give brief descriptions of the programming languages Java, C++, Python, and PHP. This will give necessary information to further understand the differences of each language.

**History of High-Level Languages**

According to the study of Kulkarni, et al. (2010) on their comparative study on programming languages:

The first high-level programming languages were designed during the 1950s. Ever since then, programming languages have been a fascinating and productive area of study. Programmers endlessly debate the relative merits of their favorite programming languages, sometimes with almost religious zeal.

On a more academic level, computer scientists search for ways to design programming languages that combine expressive power with simplicity and efficiency. The complexity of engineering software has increased dramatically in the past decade. In the early years most engineering applications were concerned solely with solving difficult numerical problems, and little attention was paid to man-machine interaction, data management, or integrated software systems. Now, computers are expected to solve a much wider variety of problems, particularly those in which numerical computations are less predominant. With the continuing increase in the variety, functionality, and complexity of engineering software, with its more widespread use, and with its increasing importance, more attention must be paid to programming language suitability so that rational decisions regarding language selection may be made. (p. 1)

These languages were designed to make programming far easier, less error-prone and to remove the programmer from having to know the details of the internal structure of a particular computer (King, 1999).

**Categories of Programming Language**

According to Suh (2011), there are four categories of programming languages – procedural, functional, object-oriented, scripting and logic. These programming languages can generally be divided into a few basic types with many languages supporting more than one programming style.

**Procedural.** Procedural languages execute a sequence of statements that lead to a result. In essence, a procedural language expresses the procedure to be followed to solve a problem. Procedural languages typically use many variables and have heavy use of loops and other elements of "state", which distinguishes them from functional programming languages.

**Functional.** The primary focus of functional programming is on the return values of functions, and side effects and other means storing state are strongly discouraged. Functional languages are often simpler syntactically and make it easier to work on abstract problems, but they can also be "further from the machine" in that their programming model makes it hard to understand exactly how the code is translated into machine language (which can be problematic for system programming).

**Object-oriented**. Object-oriented programming views the world as a collection of objects that have internal data and external means of accessing parts of that data. The goal of object-oriented programming is to think about the problem by dividing it into a collection of objects that provide services that can be used to solve a particular problem. One of the main tenets of object oriented programming is encapsulation -- that everything an object will need should be inside the object. Object-oriented programming also emphasizes reusability through inheritance and the ability to extend current implementations without having to change a great deal of code by using polymorphism.

**Scripting**. Scripting languages are often procedural and may contain elements of object-oriented languages, but they fall into their own category because they are typically not meant to be full-fledged programming languages with support for large system development. For instance, they may not have compile-time type checking or require variable declarations. Typically, scripting languages require little syntax to get started but make it very easy to make a mess.

**Logic**. Logic programming languages allow programmers to make declarative statements (possibly in first-order logic: "grass implies green" for example) and then allow the computer to reason about the consequences of those statements. In a sense, logic programming is not telling the computer how to do something, but placing constraints on what it should consider doing.

**Programming Languages**

Programming languages can be used to create programs that control the behavior of a machine, to express algorithms precisely, or as a mode of human communication (Oguntunde, 2012).

**PHP.** Naim, et al. (2010) described PHP as the following:

PHP is a powerful scripting language that can be run by itself in the command line of any computer with PHP installed. PHP was originally created by Rasmus Lerdorf in 1995 and stood for "Personal Home Page" and was released as a free, open source project. In 1997, PHP was renamed to "PHP: Hypertext Preprocessor". PHP is especially well-suited for creating dynamic web pages with connectivity to various database systems (MySQL is the most widely used because PHP provides native support for it and the database is free and an open-source project). PHP runs on different platforms and is compatible with almost all servers used today. PHP is easy to learn and runs efficiently on the server side. (p. 4)

**C++.** C++ is a fairly complicated object-oriented language derived from C (Kulkarni et al., 2010). According to Dwarampudi, Shah, Dhillon, Sebastian, and Kanigicharla (2010), “C++ is a statically typed, free-form, multi-paradigm, compiled, general-purpose programming language. It is regarded as a middle-level language, as it comprises a combination of both highlevel and low-level language features.” (Dwarampudi et al., 2010, p.2).

**Java.** Java is a simple, portable object-oriented language designed by research staff at Sun Microsystems (Kulkarni et al., 2010).

**Python.** Python is a remarkably powerful dynamic programming language that is used in a wide variety of application domains. (“About Python,” n.d.). It is a simple, easy to learn, interpreted, object oriented high-level language (S. Reddy, M.R.P. Reddy, Prakesh, 2009).

E. Scope and Limitations

The study is limited to comparing four programming languages specifically Java, C++, Python, and PHP. The criteria for comparison include versatility, program length, programming effort, memory consumption and run time efficiency. A calculator program is written in each language and is compiled, interpreted and/or ran.

The comparison for versatility revolves around whether the programming language belongs to only one or more than one of the following categories – procedural, functional, object-oriented, scripting, and logic. Program length refers to the number of non-comment lines of the source code written using each of the given languages. Programming effort takes into consideration the packages and built-in functions available for each programming language. Memory consumption refers to the amount of memory the program requires for itself. Run time efficiency includes the speed of the interpreter or compiler and runtime system during program execution.

F. Significance of the Study

Computer programmers nowadays suffer from the dilemma of having many languages to choose from. As of May 2006, Diarmuid Pigott’s Encyclopedia of Computer Languages lists 8512 computer languages. With the advent of numerous programming languages, it is difficult to comprehend the edge of one language over another. Prechelt (2000) found that “often heated, debates about different programming languages remain inconclusive”. Since there are thousands of programming languages existing, it is impossible and inefficient to put effort on analyzing each language. This study therefore seeks to provide some objective information on four among seven programming languages that are widely known and used in commercial applications (Cevasco, 2008) – Java, C++, Python, and PHP.

Choosing the best language to use on a given problem domain can be a difficult task. According to Naim, Nizam, Hanamasagar, Noureddine, and Miladinova (2010), the suitability and applicability differ from one language to the other. In order to select the proper one for a specific problem domain, one has to know what features it provides to support the requirements and whether or not it targets technical or non-technical users (Naim et al., 2010, p.1). This study could serve as guide for novice programmers by determining the similarities and differences among the given languages, finding out what these languages are used for, and categorizing them in useful ways.