Sam Ghent

CSC344

Assignment: **Programming languages I Might Like To Learn**

Abstract:

This assignment presents a short text for each of 6 programming languages that I might like to learn for one reason or another. Collectively, the short texts are intended to highlight some of the more salient features of programming languages.

---

## Language 1) **Lua**

The Lua language was developed in Brazil 1993 by a committee of three people: Roberto Lerusalimshy, Luiz Henrique de Figeiredo, and Waldemar Celes, while working at TeCGraf Institute. A project from a Brazilian oil company sent them on their way to developing a data-entry application. Due to request to keep adding more front-end features, TeCGraf decided to create a new declarative language to help them achieve this called DEL (Data-Entry Language). At the same time the oil company was working on another project and asked TeCGraf to develop a configurable report generator for lithology profiles. To configure this application the language SOL (Simple-Object Language) was born. The syntax of SOL was influenced by BiBTeX and UIL (User-Interface Language).

The actual implementation of SOL never came to be due to the committee realizing the application for lithology profiles would require way more features than they had in their existing languages already and combining SOL and DEL into a new language would be the best course of action. The committee developing Lua decided that instead of fully specifying the language before it even exist, they were going to follow a bottom-up process. They wanted to keep their language simple, small, fast, portable, and free. TeCGraf supported the development of Lua to allowed them complete freedom on the implementation of the language, so they set their goals modestly and let the experience of people actually using it dictate the direction the language should be brough in. The committee knew of the many types of people that would potentially use their language, people with a FORTRAN background, programmers who are use to C, and many potential users were not going to have expert experience programming. Consequently the language avoids cryptic syntax and semantics, but at the same time also offers optional semicolons in order to help not confuse people from different programming backgrounds. Throughout its development you can see lots of languages that have influenced Lua such as Modula, CLU, C++, SNOBOL, AWK, LISP, and Scheme. (Wikipedia)

## Why Learn **Lua**?

The implementation and development of Lua shows the effort put in to making this language as versatile as possible while simultaneously being as small as it can be. It is a language that someone with moderate programming experience should be able to pick up on quickly, only having 21 reserved words. It also has its own compiler called the Just-In-Time Complier or LuaJIT that is considered to be the fastest scripting language in the world. (LUA).

Lua can be used to develop games, web apps, tools, or extend an existing project. Popular apps today that were created using Lua were Venmo, Shopify Angry Birds, and Roblox. (LUA)

My own interest for wanting to learn Lua is for configuring a specific window manager called Awesome. It is a dynamic tiling window manager, and my hobby is customizing window managers to look however I would like.

(https://www.lua.org/history.html)

(https://en.wikipedia.org/wiki/Lua_(programming_language))

(https://staff.fnwi.uva.nl/h.vandermeer/docs/lua/luajit/luajit_intro.html)

---

## Language 2) **Kotlin**

The Kotlin language is a cross platform, statically typed, high level programming language developed in 2011 by JetBrains. There were glaring flaws with the design of Java and JetBrains decided to expand the capability of their projects by trying to use another language, but no language met their requirements. The closest was Scala but due to its long compilation time, they opted for a new language. It was then when they created their philosophy behind their Kotlin project. It gets its name from Kotlin Island, similarly to how Java was also named after an island. Kotlin is concise, safe, pragmatic, and 100% interoperable with Java, stressing that importance that Kotlin must work with 100% of all existing Java code. They achieved this by creating a the Kotlin compiler, kotlinc where its bytecode output is exactly the same. It supports object-orientated and functional programming. (OpenClassroom)

## Why Learn **Kotlin**?

Kotlin is great to learn if you are someone who wants to get into multi-platform development. It makes more sense to just code something once in Kotlin and then export it across multiple platforms rather than to learn several languages to code the same thing twice.

Kotlin is an extremely popular language and with this comes lots of documentation. If you are someone with a Java background, then you are off to an even better start to dive right into the language. The JVM environment should feel familiar and if you have any old Java projects that you wanted to expend capabilities of then Kotlin is perfect for that.

Kotlin ranks fifth in Most In-Demand Coding Languages from a survey in 2019 and in 2020 it was ranked third as a language people were planning on learning next. (KOTLINLANG)

https://kotlinlang.org/education/why-teach-kotlin.html
https://openclassrooms.com/en/courses/5774406-learn-kotlin/5930526-discover-the-history-of-kotlin

---

## Language 3) **Swift**

The Swift language is a high level complied programming language developed by Apple and the open-source community since 2010. Swift began its development all the way back in 2010 by Chris Lattner, known previously for his work on creating LLVM. Accompanying Lattner was Doug Gregor, Ted Kremenek, and Joe Groff. The development behind Swift was influenced by Object-C, Rust, Haskell, Ruby, Python, C#, and CLU and Swift own development influenced Rust. Its intended development was to replace Object-C and consequently takes a lot of its inspiration from Object-C. You can use Swift to develop mobile apps for IOS, desktop apps on macOS, wearable apps on watchOS, and is also open source so it can be used outside the Apple ecosystem as well. (Wikipedia)

## Why Learn **Swift**?

If you are creating an application that is going to run on Apple product. Be it an iPhone, iPad, Apple watch, or MacBook, you are going to need to code them in Swift in order for them to work. My personal reason for wanting to learn Swift is because I've made apps before on android studio and I would like to see those apps on iPhones as well. Since I own a mac anyway might as well give Xcode a try and see what the development flow is like for Apple software and applications.

https://en.wikipedia.org/wiki/SWIFT

---

## Language 4) **PHP**

The PHP language is a scripting language that has a focus on web development. It began development in 1993 by Rasmus Lerdorf and released in 1995. At the time Lerdorf had no intention on creating a programming language but rather was experimenting with his personal website by making it so his web server could process user request and extended them to work with web forms so he could interact with databases. PHP had a radical change in 1997 with the introduction of PHP 3. Zeev Suraski and Andi Gutmans rewrote the PHP parser, creating a more concrete foundation to build PHP off of. They also changed the name at this time, previously PHP stood for "Personal Home Page" and now was a recursive acronym "PHP: Hypertext Preprocessor". Over the years PHP has changed to add support for features and improvements such as support for object-orientated programming, native Unicode support, added a just-in-time compiler, and much more. PHP is influenced by Perl, HTML, C, C++, Java, Tcl, JavaScript, and Hack. PHP has influenced languages such as Hack, JSP, and ASP. (Wikipedia)

## Why Learn **PHP**?

PHP is a very popular language and is used by 77.8% of all the websites whose server-side programming language we know. They are a lot of practical uses for PHP. For example if you create a website for someone who is not technical you can also create a CRM for them to use with PHP so that way they can edit contents of their web pages without knowing how to program. It seems like a language that would pay off to know in the short run and the long run.

https://en.wikipedia.org/wiki/PHP

---

## Language 5) **Perl**

The Perl language is a high-level, interpreted, dynamic programming language. The development of Pearl began back in 1987 by Larry Wall while he worked at Unisys. Pearl saw rapid growth in the early days, with Pearl 3 being released in 1989. Pearls development was influenced by AWK, BASIC, C, C++, Lisp, sed, and UNIX shell.  The next major development cam with Perl 5 when they added 64-bit support and Unicode. Perl 6 was also where we saw a split though in Perl. In 2000 Larry Wall wanted to redesign Perl and announced that instead of changing Perl, he was going to create a new language called Raku with the intention “Keep Raku Perl”. He meant that he wanted Raku to be seen as clearly a Perl programming language and both have developed side by side over the years. Perl has not only influenced Raku though, it has also influenced CoffeeScript, Groovy, JavaScript, Juila, LPC, PHP, Python, Ruby, and PowerShell.

## Why Learn **Perl**?

Perl is a language that  has been around for a long time and because of that has wonderful documentation. With the help of using the Comprehensive Perl Archive Network (CPAN) you can implement Perl modules in your project all under open source licenses. Perl seems like it would be fun to learn because if you have a question it’s probably already answered on the internet or you could find the answer by searching and using CPAN. Perl is a very versatile language as well. You could use it for text processing, system administration, or web automation, just to name a few.(Perl)

http://perl-begin.org/learn/why-perl/

---

## Language 6) **Go**

The Go language is a statically typed, compiled high-level programming language designed in 2007 by Robert Griesemer, Rob Pike, and Ken Thompson. The intention behind the project was to improve programming productivity, addressing problems with other languages and keeping what they liked about them. Go was influenced by a load of languages especially C, but also Oberon-2, Limbo, Active Oberon, Pascal, Smalltalk, Newsquek, Alef, APL, BCPL, and occam. Go was finally available and released in 2012, being used today in many of Googles productions.

## Why Learn **Go**?

Go was designed to help productivity and depending on your project it absolutely can. Go could help maintain your server, create logs for you to look at, send out regular tweets, and manage data from lots of concurrent users. If you are someone who has learned C and want a similar language but with the benefits of features such as a garbage collector then it just might be right for you.

https://en.wikipedia.org/wiki/Go_(programming_language)