**The Difference Between Scripting and Non-Scripting**

One important differentiation to make in the world of coding is the difference between a scripting language and a non-scripting language. In this brief tutorial, we will define this difference and make a connection to how this influences which programming languages we choose to utilize within the context of this class.

A scripting language is a programming language that does not require compilation in order to be executed. By compilation, we mean transforming our human based code into code that a machine or electronic entity can understand. This type of code is generally referred to as machine-readable code or MRC. What a scripting language uses instead is an interpreter or system of frameworks to translate code into MRC on the fly. Examples of scripting languages include Python, Ruby, and even R (the R language is used within the R environment). Given that scripting languages use an interpreter, simple tasks can be carried out such as typing 1+2 without the need to set the answer in a variable like in a non-scripting language (see my Java example). One really nice thing about scripting languages is that you can write them in a simple text editor (or environment such as R). Also, it is very easy to fix errors/bugs in scripting languages because there is no compilation needed. The challenge here is that, if you do have an error/bug, it can be highly difficult to pinpoint where it is unless you run your code through an error-detection tool. Scripting languages are generally used for websites and online applications.

A non-scripting language is a programming language that requires compilation in order to be executed. Essentially, our human based code is translated into bytes and bits of binary code in which it is then executed by some means of initialization. Examples of non-scripting language include Java, C, and C++. One of the perks of a non-scripting language is that performance is generally better than a scripting language. This is due to the fact that the code is interpreted before it is executed. It is also worth noting that non-scripting languages are compiled in a coding environment. For example, Java can be compiled in a program called jGRASP, which can determine if errors are present in a program and even provide feedback as to where they might have initially occurred. Scripting languages, on the other hand, rely upon on the fly interpretation since they do not require compiling. One gripe against non-scripting languages, however, can come in the form of bug/error testing. For example, if you encounter an error or bug within a Java program you must stop the application from executing, go into the source code, find the error/bug, recompile the code, and rerun the application. Not only does this take a fair amount more time to fix compared to a scripting language, but it also can become vary tedious if you encounter multiple errors/bugs as you have to repeat the above process all over again per error/bug.

So there you have it, the main difference(s) between scripting and non-scripting programming languages. If you all have any questions, please feel free to ask. If we cannot immediately come to an answer for your question, we will absolutely do our best to find you one.