Aaron Goldsmith
Lab 4
How can Math, Philosophy and Computer Science possibly relate?

I was interested in seeing the discrete time differences between various ways of achieving the same goal (writing/reading to a file) given different approaches to execute system calls. Python is one of the UPnP languages, meaning that it's used to communicate between devices (usually on a local network).

My initial hypothesis was that a character stream would write faster than a string. My prediction about which would be faster came from the idea that completing an action many times one after another could prevent the pointers from traveling too far from the original code (minimizing speed). Additionally, we know a string is a disguised character array, so one might think that writing the { Int → cast to Char → write as String} may require less time than {Int → char → concatenate to my seeded string}, but my experiments would prove otherwise.

My file naming convention included the time of day and byte size to uniquely identify each file. Additionally this made finding specific files much easier, and didn't require picking through metadata. Since I was interested in identifying the subtle differences in processing speeds of different low level system calls, I wanted to see how the total write time would differ if I altered one variable that would ultimately give me the final result that I wanted. Math and computer Science have so much overlap that I wanted to test certain theories that I have noticed in class, and am only really now able to test out.

For example, in linear algebra when talking about matrices, we have to remember that the configuration of numbers on a matrix are really a way of interpreting a system. A system whose instructions are really a set of equations abstracting how variables should change. Moreover, since an infinite number of systems can exist, and we often must transform one system to determine the solution to an equation, I theorized that these different transformations could show all the different paths that could be taken to reach a solution. When thinking about why we are able to do this, I realized that these systems have constraints, just like programs do. These systems are representing how the variables within a system are going to change and affect one another. Often the path to reach a solution may seem to huge, too countably infinite, however given enough time, we could reach it. We are really determining how we can "abstract" the variables to represent them from one dimension to another.

Some examples we see this: Quaternion math which is heavily used in graphics — (rotating an arbitrary point in 3D space around a 1D Vector an arbitrary number of degrees), the library of babel, and quantum computing. All these examples have solutions to problems by using variables that we assign meaning too, abstracting it down to 0's and 1's (atomic level), then programming the computer on how to interpret that data.

Since randomness and “guessing and checking” was such a huge component in this theory, I wanted to create a program that was generative and useful. The main job of the computer program was to spit out a random integer between 65 and 125 as fast as it can (quanta). Since these numbers are representing ascii characters, the “interpretation” step isn’t as heavily important on speed since the same type cast will apply for each integer. Since physical electrons are traveling through the computer, there has to exist an initial configuration, and path such that the electrons take a “path of least resistance” to reach its goal.