**Input, Output, & Built-In, Programmer-Defined Functions**

**ITM205: Object Oriented Programming**

**Professor: Dr. Santos Galvez**

**By Odiscious Dozier**

**Trident University**

**SLP 02 – Summary**

**Summary**

Because of my practice with the string formatter in Java, this assignment was really easy to understand and execute. The “%d” formats a string to a decimal value, provided the variable passed is appropriate.

I found lists to be interesting in that the data structure that is accepted into a variable is almost explicitly like an array except that it is not wrapped in curly brackets but rather square brackets. This gives the variable a much different behavior than if the data was passed in within curly brackets. In my exploring, I have realized that Python assumes an unordered set if curly brackets are used. To add data at the end of a set, a Developer can either use the add() function for a single add or the update() to modify the set. In a list, the append() function appears to append values in the list. If the data structure is a numerical list, the list seems to put multiple integer values at one index in a given append() function call as seen in exercise 19 – 3.

Additionally, lists allow for different datatypes within the list. So, adding strings, integers, and doubles, into a single list is easily done. I have a couple of different opinions about this, but I’m too junior at programming to formulate a solid stance on the usage of such a data structure.

Exercise 20 was a real treat, for me. I even ended up helping a fellow classmate understand how to use tools like git-bash and the command prompt to not only execute files with Python but also to pass in arguments into a file while executing the file from the command prompt. This new level of knowledge is helping me start to see how I could create a package of Python files and write larger more sophisticated programs.

My understanding of Iterators over the console in Java really helped me with understanding what was going on in this assignment. The seek() function on a file places the iterator at the top row or row at index 0. That’s how the readline() function is able to continue reading from the file after the rewind() function is called. If the readline() function was never called then, the iterator would have continued to read from the bottom of the file (or wherever it left off, in our case it was after the last line of data).

Exercise 21 was something of an eye opener for me. I had never seen a function return a value where a return value was not specified in the signature of the function. It is a significant time saver, but I’m not sure how things like the copy constructor and memory space allocation are taken into consideration with Python. I do look forward to my continued studies in this area to illuminate such areas of concern for myself and others.