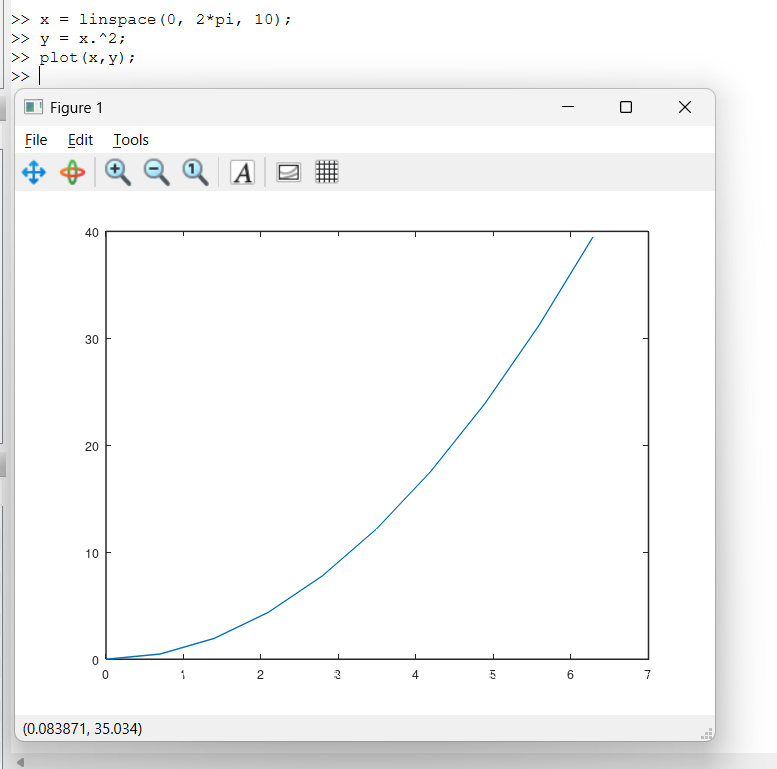
Cachary Tolentino

3/27/2025

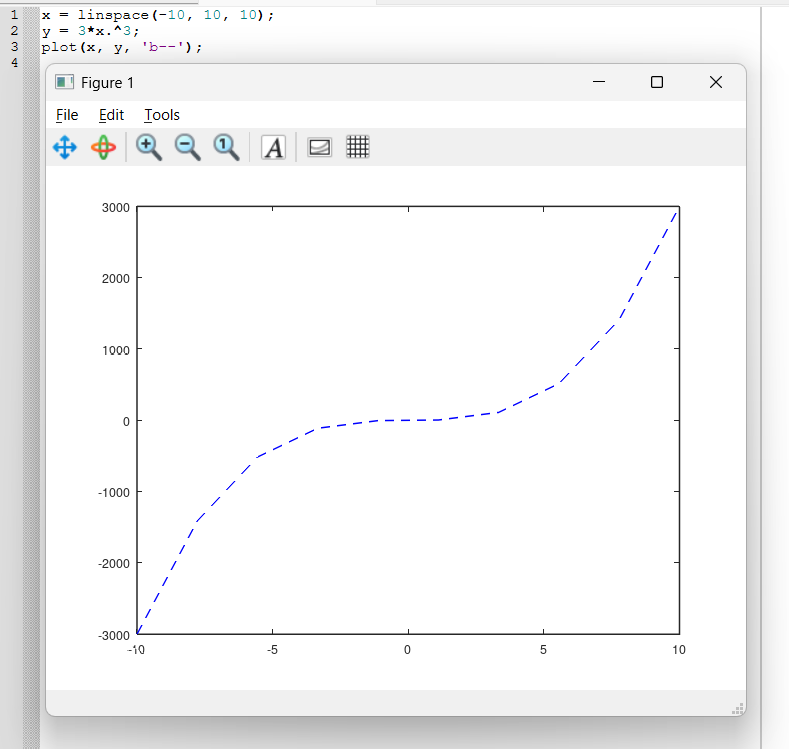
To learn GNU Octave, I started with [Octave Programming Tutorial - Wikibooks, open books for an open world](https://en.wikibooks.org/wiki/Octave_Programming_Tutorial) to learn about the basic commands and syntax. I specifically used the tutorials for “Getting started”, “Plotting”, “Text and file output”, Loops and conditions”, and “Writing functions”.

In “[Getting started](https://en.wikibooks.org/wiki/Octave_Programming_Tutorial/Getting_started)”, I learned that in the very beginning, when opening the application, the first line is called the “Octave prompt” (octave:#”); this is where all the code being run will be placed. Similar to Vscode, this could be the terminal. Octave contains many built-in mathematical functions, such as the trigonometry functions and more. Octave can plot any set of x and y values that are defined by the user. A graph can be shown by plotting the values with “plot(x,y)”. To further organize multiple plots, we can use “figure” to define a new window for a specific plot or group of plots. For example, in the following, I graphed the curve of y = x2 from 0 to 2𝜋, with 10 x values:

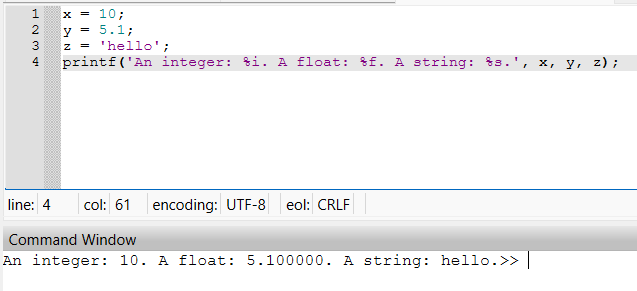


I also learned that, like vscode, to make a saveable file containing the code, we can make a script and write the code there.

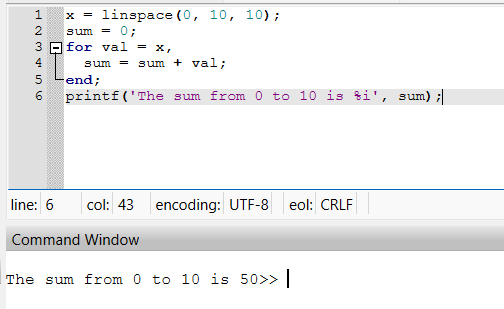
In “[Plotting](https://en.wikibooks.org/wiki/Octave_Programming_Tutorial/Plotting)”, I learned more about how the “plot()” function works. It has two different input styles. With a single parameter input, this will automatically use the given values as a y value and match with x values based on its indices. Another is with two inputs, simply the x and y values. Both inputs can either be vectors or matrices or a mix of either one. A useful function for defining the x values for a y input is using “linspace(b, 1, n)”; this essentially creates a vector of n spaced numbers between b and 1. In addition to plotting, we can also add a format to the plot, This is done by adding a FMT value within the “plot()” function. For example, in the following, we plot the function “y = 3x3” from x = -10 to 10 with 10 x value points. We formatted the output to become blue and dashed lines using “b–” as an extra formatting input:



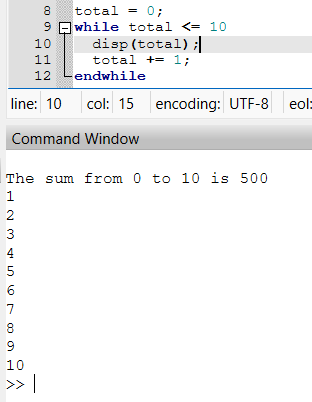
In “[Text and file output](https://en.wikibooks.org/wiki/Octave_Programming_Tutorial/Text_and_file_output)”, I learned more about how to output results for variables and others. To check what the values are for a variable, we can use “disp(variable)” to show the current values stored. We can use the disp function as a way to convert the variable into a string type by storing it in another variable. Furthermore, we can save the value to a file using “fdisp(file\_id, value)”. The proper way of outputting a value result is using the “printf” function, which also shares similar formatting options to traditional coding languages. A useful variant of printf is using “fprintf()” which prints a value into a file. For example, using printf:



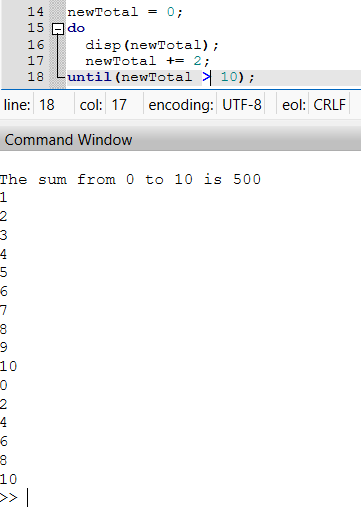
In “[Loops and conditions](https://en.wikibooks.org/wiki/Octave_Programming_Tutorial/Loops_and_conditions)”, I learned that the for loop is very similar to Java or even Python’s for loop, in which for Octave, you essentially have a loop based on a defined vector (set of numbers) to loop through, which is similar to a for each loop. For example:



While loops also perform similarly by continuously looping through a block of code until the given condition is not true. For example:

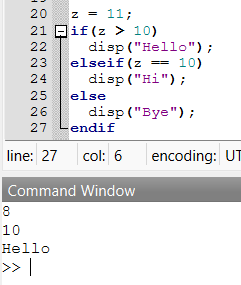


The do until loop follows the same structure as a do while loop in Java in which it performs a block of code, then it checks the condition after. For example:



Similarly, “break” and “continue” work as Java’s version would. Break would end the current loop, and the continue would allow for the loop to continue. Furthermore, either break or continue can be used in all types of loops.

Finally, to check for conditions, we would use the “if…elseif…else”. It simply checks the given condition; if true, then it will execute the block of code within; otherwise, it can continue checking the other conditions in order. If none of the conditions are fulfilled, then the else statement will be executed. For example:



Finally, in “[Writing functions](https://en.wikibooks.org/wiki/Octave_Programming_Tutorial/Writing_functions)”, I learned that it also shares a similar body structure as Java functions do, but with some slight differences. For one, the body structure of a function on Octave is structured in the following:

Function[return val,...] = name ([arg1,...])

Body of code

endFunction

It is also good practice to keep only a single function for each script while naming the script to the role of the function. For example:

