Miguel Reyes

Matlab Tutorial for Absolute Beginners: Learn Matlab via Octave in 1 hr and 30 min by Mr. STEM EDU TV

I have opted to use Octave for my plotter, salter, and smoother, providing an alternative to Matlab since Octave is free to use. Prior to working with Octave, I have viewed a beginner tutorial to familiarize with the application. The variable assignment is similar to the way BlueJ is set up, the application I am using for my original plotter, salter, and smoother. An example would be “a = 3” assigning the value of 3 to “a”, available for use after assignment as long as the program remains unchanged or “a” isn’t assigned another value. Along with assignment variables being the same, the operations that can be performed on them line up with Java. Addition, subtraction, multiplication, division, and other different operations similar to it all work similar like “a = 3” and “b = 7” can go through operations such as “c = a + b”, “c = a - b”, “c = a * b”, “c = a / b”, and “c = a ** b”, all working similar to their Java counterparts. Comments are done using the percentage sign and putting the text for the comment after the percentage sign on the same line.

The next section was on matrix operations that also work similar to the Java counterpart. The creation of a matrix is done by using brackets and putting values within the bracket, separating each row using a semicolon so long as the dimensions are consistent throughout the declaration. An example would be “A = [1 2 3; 4 5 6; 7 8 9]” being a working matrix. However “A = [1 2; 3 4 5; 6 7 8 9]” would not work because of the consistency being set by the first row “[1 2]”. You can combine matrices that comply with each other’s dimensions by setting the declaration up as “A = [B C]” so long as B and C have the dimensions mxn with both values in the variables matching up with the other matrix. You can add matrices as well using this same rule of thumb, with no change to the dimension of the matrix occurring since it’s an addition operation, not combining the matrices. Multiplication works different since the matrix dimensions that can be multiplied by each other have to follow mxn * nxp, with n being the same value. Scalar multiplication works a lot simpler since a singular number has the ability to multiply all numbers in a matrix without having to match up the column and row dimensions. Transposing a matrix is simple since you can take an existing matrix, A, and just add an apostrophe, transposing the matrix after running this.

Moving on, the next topic discussed was plotting with Octave. A simple plot command would be “plot(x,y)”, making a graph with the point you plotted. The appearance of the point can be customized using “plot(x,y,property)” with property following the format “property = ‘[line style][marker type][color]’. The different potential properties can be found via different charts that provide these values online. An example would be “plot(x,y,b-o)” making the points have a blue color with a straight line and looks like a circle for each point. Having a grid on the graph can be done using “grid on”. Linspace is an alternative to setting up points with the format being “x = linspace(-1,1,5)” which generates 5 points from -1 to 1 that increase 0.5 with each point to form a linear line. The amount of points you’d like and the range can be different from the example I provided. The default size is 100 points without declaring the size in the last parameter. Graph customization options can be done with the commands “title, xlabel, ylabel” being the basic forms of customization. The font size of the labels can be changed using ‘FontSize’ with the command “set(....)”.

Scripts was the next topic in this tutorial and it was compared to how functions operate. According to the narrator, scripts require no input since it is pre-defined, with functions requiring an input from a user and no input means the function won't run. When a script is saved, it has the file format "randomname.m" with m symbolizing that the file is Matlab related. A basic way to set up a function is "function[output1,output2,...] = functionname(input1,input2,...)" followed by Octave/Matlab code. To end a function the command "end" has to be put after all the function code. A function can be used within a function, or nested functions, by calling the function in the new function being coded. When discussing scripts, the "fprintf" command was used to show how to format a piece of code so it has a certain appearance, whether it is for how many decimal places there are or how you'd like the code output to appear. This command could be used to be more specific with output over using the print command.

Moving onto conditional statements, the setup for a conditional is more similar to how Python declares it compared to how Java does it. An example of an if-statement in Octave would be "if a >= 50" being the beginning of a statement followed by more code within the statement. The other commands that can be used with an if-statement are the "elseif" and "else" command, displaying how similar Octave is to Python for conditionals. An if-statement is ended using the code "end". For loops operate similarly to Python as well in Octave since there's no need for parentheses, following the example format "for [index] = [row vector]" which is ended with the code "end". The while loop holds up the same compared to for loops. The format used for writing while loop code is "while [conditional statement]" followed by Octave commands and ending with the command "end" once the while loop is supposed to terminate. The remainder of the video covered example code that can be written up for Octave practice but I will be ending here since the purpose was to discuss how Octave operates compared to how my original plotting, salting, and smoothing program operated in BlueJ using Java.