**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **08-07-2020** | **Name:** | **Karthik J** |
| **Course:** | MATLAB Onramp from MathWorks | **USN:** | **4AL16EC030** |
| **Topic:** | Calling Functions,  Obtaining Help,  Plotting Data. | **Semester & Section:** | **8TH A** |
| **GitHub Repository:** | Karthik-J |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image Section**     **Calling Functions** MATLAB provides a large number of functions that perform computational tasks. Functions are equivalent to subroutines or methods in other programming languages.  To call a function, such as max, enclose its input arguments in parentheses:  A = [1 3 5];  max(A)  ans = 5  If there are multiple input arguments, separate them with commas:  B = [10 6 4];  max(A,B)  ans = 1×3  10 6 5  Return output from a function by assigning it to a variable:  maxA = max(A)  maxA = 5  When there are multiple output arguments, enclose them in square brackets:  [maxA,location] = max(A)  maxA = 5  location = 3  Enclose any character inputs in single quotes:  disp('hello world')  hello world  To call a function that does not require any inputs and does not return any outputs, type only the function name:  clc  The clc function clears the Command Window. **Plotting Data**Introduction After you import data into the MATLAB® workspace, it is a good idea to plot the data so that you can explore its features. An exploratory plot of your data enables you to identify discontinuities and potential outliers, as well as the regions of interest.  The MATLAB figure window displays plots. See [Types of MATLAB Plots](https://in.mathworks.com/help/matlab/creating_plots/types-of-matlab-plots.html) for a full description of the figure window. It also discusses the various interactive tools available for editing and customizing MATLAB graphics. Load and Plot Data from Text File This example uses sample data in count.dat, a space-delimited text file. The file consists of three sets of hourly traffic counts, recorded at three different town intersections over a 24-hour period. Each data column in the file represents data for one intersection.  **Load the count.dat Data**  Import data into the workspace using the load function.  load count.dat  Loading this data creates a 24-by-3 matrix called count in the MATLAB workspace.  Get the size of the data matrix.  [n,p] = size(count)  n = 24  p = 3  n represents the number of rows, and p represents the number of columns.  **Plot the count.dat Data**  Create a time vector, t, containing integers from 1 to n.  t = 1:n;  Plot the data as a function of time, and annotate the plot.  plot(t,count),  legend('Location 1','Location 2','Location 3','Location','NorthWest')  xlabel('Time'), ylabel('Vehicle Count')  title('Traffic Counts at Three Intersections')   |  |  |  |  | | --- | --- | --- | --- | | **Date:** | **08-07-2020** | **Name:** | **Karthik J** | | **Course:** | Cisco Networking Academy | **USN:** | **4AL16EC030** | | **Topic:** | Introduction to IoT | **Semester & Section:** | **8TH A** | | **GitHub Repository:** | Karthik-J |  |  |      Programming Variables Programming languages utilize variables as dynamic buckets to hold phrases, numbers, or other important information that can be used in coding. Instead of repeating specific values in numerous places throughout the code, a variable can be used. Variables can hold the result of a calculation, the result of a database query, or some other value. This means that the same code will function using different pieces of data without having to be rewritten.  For instance “x + y = z” is an example of a programming expression. In this expression, x, y and z are variables which can represent characters, character strings, numeric values or memory addresses.  A variable can refer to a value. For instance the expression “a = 10” associates the value 10 to variable a.  A variable can also represent a memory location. The expression “a = 10” represents that the value 10 is stored in some location of the computer memory, which is referred to as ‘a’.  Variables can be classified into two categories:   * **Local Variables** - These are variables that are within the scope of a program / function / procedure. * **Global Variables** - These are variables that are in the scope for the time of the program’s execution. They can be retrieved by any part of the program.   Variables allow programmers to quickly create a wide range of simple or complex programs which tell the computer to behave in a pre-defined fashion. Basic Program Structures People impart logic to computers through programs. Using specific logic structures, a programmer can prepare a computer to make decisions. The most common logic structures are:   * **IF – THEN** - This logic structure allows the computer to make a decision based on the result of an expression. An example of an expression is myVar > 0. This expression is true if the value stored in the myVar variable is greater than zero. When an IF-THEN structure is encountered, it evaluates the provided expression. If the expression is false, the computer moves on to the next structure, ignoring the contents of the IF-THEN block. If the expression is true, the computer executes the associated action before moving on to the next instruction in the program. * **FOR Loops** – These are used to execute a specific set of instructions a specific number of times, based on an expression. The term loop comes from the fact that the set of instructions is executed repeatedly. While the syntax of FOR loops varies from language to language, the concept remains the same. A variable acts as a counter inside a range of values identified by a minimum and a maximum. Every time the loop is executed, the counter variable is incremented. When the counter is equal to the defined maximum value, the loop is abandoned and the execution moves on to the next instruction. * **WHILE Loops** – These are used to execute a specific set of instructions while an expression is true. Notice that often the instructions inside the loop will eventually make the expression evaluate as false. |