**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **15-July-2020** | **Name:** | **Bhuvanesh M** |
| **Course:** | **Matlab Onramp** | **USN:** | **4AL16EC015** |
| **Topic:** | **Calling Functions** | **Semester & Section:** | **8th sem & ‘A’ section** |
| **Github Repository:** | **Bhuvan** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –**  In my first session today I have studied about -Calling Functions Command vs. Function SyntaxCommand and Function Syntaxes In MATLAB®, these statements are equivalent:  load durer.mat % Command syntax  load('durer.mat') % Function syntax  This equivalence is sometimes referred to as command-function duality.  All functions support this standard function syntax:  [output1, ..., outputM] = functionName(input1, ..., inputN)  If you do not require any outputs from the function, and all of the inputs are character vectors (that is, text enclosed in single quotation marks), you can use this simpler command syntax:  functionName input1 ... inputN  With command syntax, you separate inputs with spaces rather than commas, and do not enclose input arguments in parentheses. Command syntax always passes inputs as character vectors. To use strings as inputs, use the function syntax. If a character vector contains a space, use the function syntax.  When a function input is a variable, you must use function syntax to pass the value to the function. Command syntax always passes inputs as character vectors and cannot pass variable values. For example, create a variable and call the disp function with function syntax to pass the value of the variable:  A = 123;  disp(A)  This code returns the expected result,  123  You cannot use command syntax to pass the value of A, because this call  disp A  is equivalent to  disp('A')  and returns  A Avoid Common Syntax Mistakes Suppose that your workspace contains these variables:  filename = 'accounts.txt';  A = int8(1:8);  B = A;  The following table illustrates common misapplications of command syntax.   | This Command... | Is Equivalent to... | Correct Syntax for Passing Value | | --- | --- | --- | | open filename | open('filename') | open(filename) | | isequal A B | isequal('A','B') | isequal(A,B) | | strcmp class(A) int8 | strcmp('class(A)','int8') | strcmp(class(A),'int8') | | cd matlabroot | cd('matlabroot') | cd(matlabroot) | | isnumeric 500 | isnumeric('500') | isnumeric(500) | | round 3.499 | round('3.499'), which is equivalent to round([51 46 52 57 57]) | round(3.499) | | disp hello world | disp('hello','world') | disp('hello world') | | disp "string" | disp('"string"') | disp("string") |  Passing Variable Names Some functions expect character vectors for variable names, such as save, load, clear, and whos. For example,  whos -file durer.mat X  requests information about variable X in the example file durer.mat. This command is equivalent to  whos('-file','durer.mat','X') How MATLAB Recognizes Command Syntax Consider the potentially ambiguous statement  ls ./d  This could be a call to the ls function with the folder ./d as its argument. It also could request element-wise division on the array ls, using the variable d as the divisor.  If you issue such a statement at the command line, MATLAB can access the current workspace and path to determine whether ls and d are functions or variables. However, some components, such as the Code Analyzer and the Editor/Debugger, operate without reference to the path or workspace. In those cases, MATLAB uses syntactic rules to determine whether an expression is a function call using command syntax.  In general, when MATLAB recognizes an identifier (which might name a function or a variable), it analyzes the characters that follow the identifier to determine the type of expression, as follows:   * An equal sign (=) implies assignment. For example:   ls =d   * An open parenthesis after an identifier implies a function call. For example:   ls('./d')   * Space after an identifier, but not after a potential operator, implies a function call using command syntax. For example:   ls ./d   * Spaces on both sides of a potential operator, or no spaces on either side of the operator, imply an operation on variables. For example, these statements are equivalent: * ls ./ d   ls./d  Therefore, the potentially ambiguous statement ls ./d is a call to the ls function using command syntax.  The best practice is to avoid defining variable names that conflict with common functions, to prevent any ambiguity. |