**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **07-July-2020** | **Name:** | **Raziya Banu** |
| **Course:** | **Matlab Onramp** | **USN:** | **4AL16EC058** |
| **Topic:** | **Calling Functions** | **Semester & Section:** | **8th sem & ‘B’ section** |
| **Github Repository:** |  |  |  |

|  |
| --- |
| **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. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:** | **07-July-2020** | **Name:** | **Raziya Banu** | |
| **Course:** | **Cisco** | **USN:** | **4AL16EC058** | |
| **Topic:** | [**Everything Becomes Programmable**](https://static-course-assets.s3.amazonaws.com/I2IoT20/en/index.html#2) | **Semester & Section:** | **8th sem & ‘B’ section** | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Flowcharts**  Flowcharts are used in many industries including engineering, physical sciences, and computer programming where a complete understanding of processes or workflows is required. Flowcharts are diagrams that are used to represent these processes or workflows.  Flowcharts illustrate how a process should work. Flowcharts should not require complex, industry-specific terminology or symbols. A flowchart should be easy to understand without having to be an expert in the chosen field.  Flowcharts should show input states, any decisions made, and the results of those decisions. It is important to show the steps that should be taken when the result of a decision is either yes or no.  It is common for programmers to create a first draft of a program in no specific programming language. These language-independent programs are focused on the logic rather than in the syntax and are often called algorithms. A flowchart is a common way to represent an algorithm. An example of a flowchart is shown in the figure. **System Software, Application Software, and Computer Languages** There are two common types of computer software: system software and application software.  Application software programs are created to accomplish a certain task or collection of tasks. For example, Cisco Packet Tracer is a network simulation program that allows users to model complex networks and ask “what if” questions about network behavior.  System software works between the computer hardware and the application program. It is the system software that controls the computer hardware and allows the application programs to function. Common examples of system software include Linux, Apple OSX, and Microsoft Windows.  Both system software and application software are created using a programming language. A programming language is a formal language designed to create programs that communicate instructions to computer hardware. These programs implement algorithms which are self-contained, step-by-step sets of operations to be performed.  Some computer languages compile their programs into a set of machine-language instructions. C++ is an example of a compiled computer language. Others interpret these instructions directly without first compiling them into machine language. Python is an example of an interpreted programming language. An example of Python code is shown in the figure.  When the programming language is determined and the process is diagrammed in a flowchart, program creation can begin. Most computer languages use similar program structures. **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. (Figure 1). * **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. (Figure 2). * **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.  **Lab - Create a Process Flowchart** Flowcharts are normally used to diagrammatically illustrate the process flow before a computer program is created. In this lab you will create a simple flowchart showing the process used to find a predetermined integer value. | | | |