**ASSESSMENT 43**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | 08-07-2020 | **Name:** | Sheela Golasangi |
| **Course:** | Matlab Onramp | **USN:** | 4AL16EC068 |
| **Topic:** | Logical arrays, programming, final project, conclusion.  CERTIFICATE | **Semester & Section:** | VIII  ‘B’ |
| **Github Repository:** | Sheela-Course |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **REPORT**  **C:\Users\india\Pictures\Screenshots\Screenshot (1193).png**  C:\Users\india\Pictures\Screenshots\Screenshot (1174).png  C:\Users\india\Pictures\Screenshots\Screenshot (1177).png  MATLAB offers two types of logical operators and functions −   * Element-wise − these operators operate on corresponding elements of logical arrays. * Short-circuit − these operators operate on scalar, logical expressions.   Element-wise logical operators operate element-by-element on logical arrays. The symbols &, |, and ~ are the logical array operators AND, OR, and NOT.  Short-circuit logical operators allow short-circuiting on logical operations. The symbols && and || are the logical short-circuit operators AND and OR. Example Create a script file and type the following code −  a =5;  b =20;  if( a && b )  disp('Line 1 - Condition is true');  end  if( a || b )  disp('Line 2 - Condition is true');  end  % lets change the valueof a and b  a =0;  b =10;  if( a && b )  disp('Line 3 - Condition is true');  else  disp('Line 3 - Condition is not true');  end  if(~(a && b))  disp('Line 4 - Condition is true');  end  When you run the file, it produces following result −  Line 1 - Condition is true  Line 2 - Condition is true  Line 3 - Condition is not true  Line 4 - Condition is true Functions for Logical Operations Apart from the above-mentioned logical operators, MATLAB provides the following commands or functions used for the same purpose −   |  |  | | --- | --- | | **Sr.No.** | **Function & Description** | | 1 | **and(A, B)**  Finds logical AND of array or scalar inputs; performs a logical AND of all input arrays A, B, etc. and returns an array containing elements set to either logical 1 (true) or logical 0 (false). An element of the output array is set to 1 if all input arrays contain a nonzero element at that same array location. Otherwise, that element is set to 0. | | 2 | **not(A)**  Finds logical NOT of array or scalar input; performs a logical NOT of input array A and returns an array containing elements set to either logical 1 (true) or logical 0 (false). An element of the output array is set to 1 if the input array contains a zero value element at that same array location. Otherwise, that element is set to 0. | | 3 | **or(A, B)**  Finds logical OR of array or scalar inputs; performs a logical OR of all input arrays A, B, etc. and returns an array containing elements set to either logical 1 (true) or logical 0 (false). An element of the output array is set to 1 if any input arrays contain a nonzero element at that same array location. Otherwise, that element is set to 0. | | 4 | **xor(A, B)**  Logical exclusive-OR; performs an exclusive OR operation on the corresponding elements of arrays A and B. The resulting element C(i,j,...) is logical true (1) if A(i,j,...) or B(i,j,...), but not both, is nonzero. | | 5 | **all(A)**  Determine if all array elements of array A are nonzero or true.   * If A is a vector, all(A) returns logical 1 (true) if all the elements are nonzero and returns logical 0 (false) if one or more elements are zero. * If A is a nonempty matrix, all(A) treats the columns of A as vectors, returning a row vector of logical 1's and 0's. * If A is an empty 0-by-0 matrix, all(A) returns logical 1 (true). * If A is a multidimensional array, all(A) acts along the first non-singleton dimension and returns an array of logical values. The size of this dimension reduces to 1 while the sizes of all other dimensions remain the same. | | 6 | **all(A, dim)**  Tests along the dimension of A specified by scalar *dim*. | | 7 | **any(A)**  Determine if any array elements are nonzero; tests whether any of the elements along various dimensions of an array is a nonzero number or is logical 1 (true). The any function ignores entries that are NaN (Not a Number).   * If A is a vector, any(A) returns logical 1 (true) if any of the elements of A is a nonzero number or is logical 1 (true), and returns logical 0 (false) if all the elements are zero. * If A is a nonempty matrix, any(A) treats the columns of A as vectors, returning a row vector of logical 1's and 0's. * If A is an empty 0-by-0 matrix, any(A) returns logical 0 (false). * If A is a multidimensional array, any(A) acts along the first non-singleton dimension and returns an array of logical values. The size of this dimension reduces to 1 while the sizes of all other dimensions remain the same. | | 8 | **any(A,dim)**  Tests along the dimension of A specified by scalar *dim*. | | 9 | **false**  Logical 0 (false) | | 10 | **false(n)**  is an n-by-n matrix of logical zeros | | 11 | **false(m, n)**  is an m-by-n matrix of logical zeros. | | 12 | **false(m, n, p, ...)**  is an m-by-n-by-p-by-... array of logical zeros. | | 13 | **false(size(A))**  is an array of logical zeros that is the same size as array A. | | 14 | **false(...,'like',p)**  is an array of logical zeros of the same data type and sparsity as the logical array p. | | 15 | **ind = find(X)**  Find indices and values of nonzero elements; locates all nonzero elements of array X, and returns the linear indices of those elements in a vector. If X is a row vector, then the returned vector is a row vector; otherwise, it returns a column vector. If X contains no nonzero elements or is an empty array, then an empty array is returned. | | 16 | **ind = find(X, k)**  **ind = find(X, k, 'first')**  Returns at most the first k indices corresponding to the nonzero entries of X. k must be a positive integer, but it can be of any numeric data type. | | 17 | **ind = find(X, k, 'last')**  returns at most the last k indices corresponding to the nonzero entries of X. | | 18 | **[row,col] = find(X, ...)**  Returns the row and column indices of the nonzero entries in the matrix X. This syntax is especially useful when working with sparse matrices. If X is an N-dimensional array with N > 2, col contains linear indices for the columns. | | 19 | **[row,col,v] = find(X, ...)**  Returns a column or row vector v of the nonzero entries in X, as well as row and column indices. If X is a logical expression, then v is a logical array. Output v contains the non-zero elements of the logical array obtained by evaluating the expression X. | | 20 | **islogical(A)**  Determine if input is logical array; returns true if A is a logical array and false otherwise. It also returns true if A is an instance of a class that is derived from the logical class. | | 21 | **logical(A)**  Convert numeric values to logical; returns an array that can be used for logical indexing or logical tests. | | 22 | **true**  Logical 1 (true) | | 23 | **true(n)**  is an n-by-n matrix of logical ones. | | 24 | **true(m, n)**  is an m-by-n matrix of logical ones. | | 25 | **true(m, n, p, ...)**  is an m-by-n-by-p-by-... array of logical ones. | | 26 | **true(size(A))**  is an array of logical ones that is the same size as array A. | | 27 | **true(...,'like', p)**  is an array of logical ones of the same data type and sparsity as the logical array p. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | 08-07-2020 | **Name:** | Sheela Golasangi |
| **Course:** | Cisco Certification Course:  Introduction to Internet of Things | **USN:** | 4AL16EC068 |
| **Topic:** | Chapter 3 | **Semester & Section:** | VIII  ‘B’ |
| **Github Repository:** | Sheela-Course |  |  |
| **AFTERNOON SESSION DETAILS** | | | |
| C:\Users\india\Pictures\Screenshots\Screenshot (1199).pngC:\Users\india\Pictures\Screenshots\Screenshot (1197).png C:\Users\india\Pictures\Screenshots\Screenshot (1198).png What is Big Data?Data is information that comes from a variety of sources, such as people, pictures, text, sensors, and web sites. Data also comes from technology devices like cell phones, computers, kiosks, tablets, and cash registers. Most recently, there has been a spike in the volume of data generated by sensors. Sensors are now installed in an ever growing number of locations and objects. These include security cameras, traffic lights, intelligent cars, thermometers, and even grape vines! Big Data is a lot of data, but what is a lot? No one has an exact number that says when data from an organization is considered “Big Data.” Here are three characteristics that indicate an organization may be dealing with Big Data: They have a large amount of data that increasingly requires more storage space (volume). They have an amount of data that is growing exponentially fast (velocity).They have data that is generated in different formats (variety).How much data do sensors collect? Here are some estimated examples: Sensors in one autonomous car can generate 4,000 gigabits (Gb) of data per day. An Airbus A380 Engine generates 1 petabyte (PB) of data on a flight from London to Singapore. Safety sensors in mining operations can generate up to 2, 4 terabits (TB) of data every minute. Sensors in one smart connected home can produce as much as 1 gigabyte (GB) of information a week.Large Data bases While Big Data does create challenges for organizations in terms of storage and analytics, it can also provide invaluable information to fine-tune operations and improve customer satisfaction. Companies do not necessarily have to generate their own Big Data. Smaller organizations might not have the sensors, the volume of customers, or the ability to generate the variety of information that could benefit their company. There are sources of free data sets available, ready to be used and analyzed by anyone willing to look for them. Many companies of various sizes believe they have to collect their own data to see benefits from big data analytics, but it is simply not true. What Are the Challenges of Big Data? IBM’s Big Data estimates conclude that “each day we create 2.5 quintillion bytes of data”. To put this into context, every minute of every day: We upload over 300 hours of YouTube video. We send over 3.5 million text messages. We stream over 86thousand hours of Netflix video. We like over 4 million Facebook posts. We request over 14 million forecasts from The Weather Channel. To see more live Internet statistics. The rapid growth of data can be an advantage or an obstacle when it comes to achieving business goals. To be successful, enterprises must be able to easily access and manage their data assets. With this enormous amount of data being constantly created, traditional technologies and data warehouses cannot keep up with storage needs. Even with the cloud storage facilities that are available from companies like Amazon, Google, Microsoft, and many others, the security of stored data becomes a big problem. Big Data solutions must be secure, have a high fault tolerance, and use replication to ensure data does not get lost. Big Data storage is not only about storing data, it is also about managing and securing | | | |