Armstrong State University
Engineering Studies
MATLAB Marina – Logic Expressions Primer

Prerequisites
The Logic Expressions Primer assumes knowledge of the MATLAB IDE, MATLAB help, arithmetic operations, built in functions, scripts, variables, and 1D Arrays and Vectors. Material on these topics is covered in the MATLAB Marina Introduction to MATLAB module, MATLAB Marina Variables module, and MATLAB Marina 1D Arrays and Vectors module.

Learning Objectives
1. Be able to form logic expressions using logical and relational operators for evaluating conditions.
2. Be able to use the result of logic expressions and MATLAB's `find` function for indexing.

Terms
Boolean, logical operator, relational operator, logic expression, compound logic expression

MATLAB Functions, Keywords, and Operators
==, <, <=, >, >=, ~, ~=, &, &&, |, ||, find, true, 1, false, 0

Boolean Values
Boolean variables can take on one of two values: true (1) or false (0). Variables can take on a Boolean value either through direct assignment or as the result of a logic expression. Note: MATLAB variables of any type can be used in a logic expression and any nonzero value is treated as true.

Logic Expressions
Logic expressions are typically used for two things in MATLAB: selectively executing blocks of statements and indexing a portion of an array. Logic expressions used with `if-else` and `while` statements (covered in a later primer) for selectively executing blocks of code should result in a scalar Boolean result. Logic expressions used for indexing should result in an array of Booleans of the same size as the array to index.

Logic expressions typically consist of one or more variables, logical and relational operators, and constants (sometimes mathematical operations are also used). Logic expressions are evaluated from left to right adhering to operator precedence rules. A MATLAB logic expression is true when the result is nonempty and all elements are nonzero; otherwise the expression is false. Logic expressions used for selectively executing blocks of code do not typically need the result saved in a variable whereas logic expressions used for indexing typically need the result saved to a variable that will be used for the indexing. Table 1 provides a table of the commonly used MATLAB logical and relational operators.

MATLAB Logical and Relational Operators	
Symbol	Operation
&	Element by element logical AND
&&	Short circuit logical AND
|	Element by element logical OR
||	Short circuit logical OR
~	Element by element logical NOT
==	Equal to
~=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Table 1, Commonly Used MATLAB Logical and Relational Operators

Figure 1a shows examples of logic expressions used for selectively executing blocks of code and Figure 1b shows examples of logic expressions used for indexing arrays.

```
age >= 18
number < 0
length(arr) == 0         % isempty(arr)
```

Figure 1a, Logic Expressions used for Selectively Executing Blocks of Code

MATLAB has a built in function `isempty` that is more efficient to use than checking if the length or size of an array is zero.

```
>> tt = 0.0 : 0.1 : 10.0;
>> xx = 5*cos(2*pi*tt + pi/2);
>> loc = (tt > 5.0);                   % portion for t > 5
>> tt2 = tt(loc);
>> xx2 = xx(loc);
>> loc = (xx == 5.0);                  % find peaks of sinusoid
>> ttPeaks = tt(loc);
```

Figure 1b, Logic Expressions Used for Indexing

Checking for equality using real numbers often does not work well as rarely will a real number be exactly equal to the desired value. Either a logic array or array of indices can be used to index an array to extract the elements in the array matching the condition. When indexing with logic arrays, the elements corresponding to the true indices are returned. MATLAB's `find` function will convert the logic arrays to an array of indices corresponding to where the logic array is true.

Remember that the single equals (=) is for assignment and the double equals (==) is for comparisons in logic expressions.

Compound Logic Expressions
Logical operators (typically logical AND and logical OR) can be used to combine multiple logic expressions for complex conditions. Compound logic expressions used for selectively executing blocks of code should result in a scalar Boolean result and compound logic expressions used for indexing should result in an array of Booleans of the same size as the array to index.

Figure 2a shows examples of compound logic expressions used for selectively executing blocks of code and Figure 2b shows examples of compound logic expressions used for indexing arrays.

```
(age >= 10) & (age < 20)
(yearlyWage > 50000) | (hourlyWage > 18.00)
```

Figure 2a, Compound Logic Expressions used for Selectively Executing Blocks of Code

```
>> tt = 0.0 : 0.1 : 10.0;
>> xx = 5*cos(2*pi*tt + pi/2);
>> loc = (tt > 2.0) & (tt < 3.0);  % portion for 2 < t < 3
>> tt2 = tt(loc);
>> xx2 = xx(loc);
>> loc = (xx == 5.0) | (xx == -5); % find peaks and troughs
>> ttPeaks = tt(loc);
```

Figure 2b, Compound Logic Expressions Used for Indexing

Last modified Tuesday, September 09, 2014

This work by Thomas Murphy is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.