17-Relational and Logical Operators

text: Chapter 6

ECEGR 101

Engineering Problem Solving with Matlab

Professor Henry Louie

Overview

- Relational Operators
- Logical Operators
- Order of Operations
- Logical Functions

Tools for Programming in MATLAB

Part A:

Relational and logical operators

- used in programming to make decisions

Part B:

Conditional statements: if, switch

- used to select a group of commands to execute

Part C:

Loops: for, while

- used to repeat commands several times

Flow control commands: break, continue

- stop and resume execution of commands

Relational Operators

- Used to make comparisons between arrays:

  < less than

  <= less than or equal to

  > greater than

  >= greater than or equal to

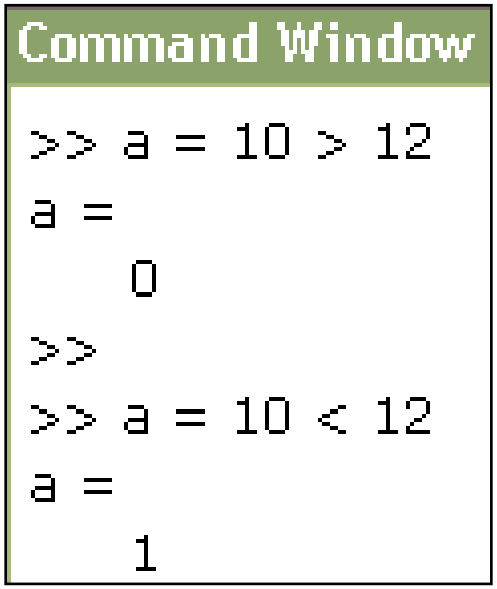
  == equal to

  ~= not equal to

- Result is of a logical type and can be stored in a variable:

  0 – false, 1 - true

Examples




```
>> a = 10 == 12
a =
     0
>>
>> a = 10 ~= 10
a =
     0
>>
```

Examples

```
>> x = [1 4 2 6 3 8 4 0]
x =
    1    4    2    6    3    8    4    0
>>
>> y = x >= 3
y =
    0    1    0    1    1    1    1    0
>>
```

Element-by-element comparison for arrays

```
>> x = [10; 0; 1]
x =
   10
    0
    1
>>
>> y = [10; -1; 1]
y =
   10
   -1
    1
>>
>> z = x ~= y
z =
    0
    1
    0
```

Examples

Command Window

```
>> x = [1 5 2; 5 2 5; 6 5 5]
x =
     1     5     2
     5     2     5
     6     5     5
>>
>> y = [8 5 9; 5 2 8; 6 0 0]
y =
     8     5     9
     5     2     8
     6     0     0
>>
```

```
>> r = x == y
r =
     0     1     0
     1     1     0
     1     0     0
```

```
>> r = x >= y
r =
     0     1     0
     1     1     0
     1     1     1
>>
```

Examples

Command Window

```
>> x = [1 5 2; 5 2 5; 6 5 5]
x =
     1     5     2
     5     2     5
     6     5     5
>>
>> y = [8 5 9; 5 2 8; 6 0 0]
y =
     8     5     9
     5     2     8
     6     0     0
>>
```

```
>> z = (x < 5) == (y >= 0)
z =
     1     0     1
     0     1     0
     0     0     0
>>
```

Examples

The result from relational operation can be used to address a matrix.

```
>> x = [1 -1 2 -4; 2 5 7 -9; 3 -3 4 4; -8 -7 4 5]
x =
     1    -1     2    -4
     2     5     7    -9
     3    -3     4     4
    -8    -7     4     5
```

Location of negative elements of x

```
>> i = x < 0
i =
     0     1     0     1
     0     0     0     1
     0     1     0     0
     1     1     0     0
```

```
>> x(i) = 10
x =
     1    10     2    10
     2     5     7    10
     3    10     4     4
    10    10     4     5
```

Examples

```
>> x = [1 -1 3 -3 4 8 -2 9 0]
x =
    1    -1     3    -3     4     8    -2     9     0
>>
>> y = [1 -2 4 -2 5 6 3 -2 -1]
y =
    1    -2     4    -2     5     6     3    -2    -1
```

```
>> x<y
ans =
    0     0     1     1     1     0     1     0     0
>> y(x<y)
ans =
    4    -2     5     3
```

Elements of y that are greater than x

```
>> y(x<y) = 0
y =
    1    -2     0     0     0     6     0    -2    -1
```

Exercise

Create:

X = [1 2 3 4 5 6 7 8 9 10]

Y = [10 9 8 7 6 5 4 3 2 1]

Do the following:

>>i = X<Y

>>class(i)

>>X(i)

>>Y(i)

since the class of i is logical, you can use i to index another array

Exercise

```
>> X = [ 1   2   3   4  5  6  7  8  9  10];
>> Y = [10  9   8   7  6  5  4  3  2  1];

>> i = X<Y
i =
    1    1    1    1    1    0    0    0    0    0

>> class(i)
ans =
logical
```

```
>> X(i)
ans =
    1    2    3    4    5
>> Y(i)
ans =
   10    9    8    7    6
```

Examples

```
>> x = [1 2; 3 4]
x =
     1     2
     3     4
>>
>> i = [1 0; 0 1]
i =
     1     0
     0     1
```

To address a matrix using another matrix, the addressing matrix has to be of a logical class.

```
>> class(i)
ans =
double
```

```
>> x(i)
??? Subscript indices must either be real positive integers or logicals.
```

Examples

```
>> x = [1 2; 3 4]
x =
     1     2
     3     4
>>
>> i = [1 0; 0 1]
i =
     1     0
     0     1
```

To address a matrix using another matrix, the addressing matrix has to be of a logical class.

```
>> i = logical(i);
>>
>> x(i)
ans =
     1
     4
```

Exercise

Create:

```
>> x = [1:3;  2:2:6;  linspace(4,10,3)]
```

Modify x so the elements that are less than 5 get replaced by 200.

Exercise

Create:

```
>> x = [1:3;  2:2:6;  linspace(4,10,3)]
```

Modify x so the elements that are less than 5 get replaced by 200.

```
>> x(x<5) = 200
```

Exercise

```
>> x = [1:3;  2:2:6;  linspace(4,10,3)]
x =
     1     2     3
     2     4     6
     4     7    10
>>
>> x(x<5) = 200
x =
   200   200   200
   200   200     6
   200     7    10
```

Relational Operators

- Order of operations:
  - Arithmetic operations (+, -, *, /) will be executed before relational operators.
- Example:

```
>> x = 2+3-6 < 4*5
x =
     1
```

```
>> x = 2+3-(6 < 4)*5
x =
     5
```

```
>> x = 2+(3-6 < 4)*5
x =
     7
```

```
>> x = 2+3-(6 < 4*5)
x =
     4
```

Logical Operators

- AND (&)
  - The result of x&y is true (1) if both x and y are true, otherwise the result is false (0).
- OR (|)
  - The result of x|y is true (1) if either one, or both are true, otherwise (both are false) the result is false.
- NOT (~)
  - The result of ~x is true (1) if x is false, and false (0) if x is true.

Examples

- Note that for logical operations in MATLAB:
  - x=0 is equivalent to logical false
  - x≠0 is equivalent to logical true

```
>> x = [1 0 1 1 0]
x =
     1     0     1     1     0
>>
>> y = ~x
y =
     0     1     0     0     1
```

```
>> x = [1 -3 0 9 0 0 3]
x =
     1    -3     0     9     0     0     3
>>
>> y = ~x
y =
     0     0     1     0     1     1     0
```

```
>> y = logical(x)
Warning: Values other than 0 or 1 converted to logical 1
y =
     1     1     0     1     0     0     1
```

Examples

```
>> x = [1 -3 0 9 0 0 3]
x =
     1    -3     0     9     0     0     3
>>
>> y = [3 0 5 2 -2 0 4]
y =
     3     0     5     2    -2     0     4
```

All non-zero values treated as true (1)

```
>> z = x&y
z =
     1     0     0     1     0     0     1
>>
>> z = x|y
z =
     1     1     1     1     1     0     1
```

Examples

```
>> x = [1 -3 0 9 0 0 3]
x =
     1    -3     0     9     0     0     3
>>
>> y = [3 0 5 2 -2 0 4]
y =
     3     0     5     2    -2     0     4
```

```
>> (x > 1) & ((y == 0) | (y < 3))
ans =
     0     0     0     1     0     0     0
```

```
>> (x > 1) | (y == 0)
ans =
     0     1     0     1     0     1     1
```

Examples

```
>> x = [1 -3 0 9 0 0 3]
x =
    1    -3     0     9     0     0     3
>>
>> y = [3 0 5 2 -2 0 4]
y =
    3     0     5     2    -2     0     4
```

```
>> (x > 1) & (y == 0) | (y < 3)
ans =
    0     1     0     1     1     1     0
```

```
>> ((x > 1) & (y == 0)) | ~(y < 3)
ans =
    1     0     1     0     0     0     1
```

Exercise

How would you write the relation $5 < x < 10$ in MATLAB?

Exercise

How would you write the relation $5 < x < 10$ in MATLAB?

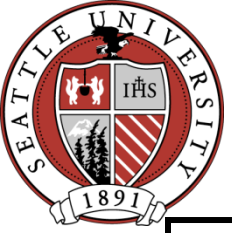
```
>> x = 7;
>>
>> (x > 5) & (x < 10)
ans =
     1
>>
>>
>> (5 < x) & (x < 10)
ans =
     1
```

Built-in Logical Functions

- and(x,y)
  - equivalent to x&y
- or(x,y)
  - equivalent to x|y
- not(x)
  - equivalent to ~x
- xor(x,y) exclusive or
  - Returns true (1) if x is true and y is false or if x is false or y is true.

The Truth Table

INPUT		OUTPUT				
X	**Y**	**AND X&Y**	**OR X|Y**	**XOR XOR(X,Y)**	**NOT ~X**	**NOT ~Y**
false	false	false	false	false	true	true
false	true	false	true	true	true	false
true	false	false	true	true	false	true
true	true	true	true	false	false	false



The Truth Table

INPUT		OUTPUT				
X	**Y**	**AND X&Y**	**OR X|Y**	**XOR XOR(X,Y)**	**NOT ~X**	**NOT ~Y**
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

Exercise

Generate the truth table in terms of ones and zeros in MATLAB.

Exercise

Generate the truth table in terms of ones and zeros in MATLAB.

```
>> x = [1; 1; 0; 0];
>> y = [1; 0; 1; 0];
>>
>> Truth_Table = [x y ~x x|y x&y xor(x,y)]
Truth_Table =
     1     1     0     1     1     0
     1     0     0     1     0     1
     0     1     1     1     0     1
     0     0     1     0     0     0
>>
```

Operator Precedence

Precedence	**Operator Type**
First	Parentheses: evaluated starting with the innermost pair.
Second	Exponentiation (exp).
Third	Logical NOT (~).
Fourth	Arithmetic operators: * /
Fifth	Arithmetic operators: + -
Sixth	Relational operators; evaluated from left to right.
Seventh	Logical AND (&).
Eighth	Logical OR (|)

Examples

```
>> z = 1&2+3
z =
     1
>>
>> z = 1&(2+3)
z =
     1
```

```
>> z = 5<6&1
z =
     1
>>
>> z = (5<6)&1
z =
     1
```

```
>> z = 3<5|4==7
z =
     1
>>
>> z = (3<5)|(4==7)
z =
     1
```

```
>> z = ~3==7|4==6
z =
     0
>>
>> z = ((~3)==7)|(4==6)
z =
     0
```

Exercise

If x = [5 -3 18 4] and y = [-9 13 7 4], what will be the result of the following operation?

z = x&y

```
>> x = [5 -3 18 4]
x =
    5   -3   18    4
>>
>> y = [-9 13 7 4]
y =
   -9   13    7    4
```

Exercise

If x = [5 -3 18 4] and y = [-9 13 7 4], what will be the result of the following operation?

z = x&y

```
>> x = [5 -3 18 4]
x =
     5    -3    18     4
>>
>> y = [-9 13 7 4]
y =
    -9    13     7     4
```

```
>> z = x&y
z =
     1     1     1     1
```

Built-in Logical Functions

all(x)

- Returns true (1) if all elements of x are true (1).
- Returns false (0) if one of more elements are false (0).

```
>> x = [1 7 4 5];
>>
>> all(x)
ans =
     1
```

```
>> x = [1 7 4 0];
>>
>> all(x)
ans =
     0
```

```
>> x = [0 0 0 0];
>>
>> all(x)
ans =
     0
```

```
>> y = [1 4; 6 7; 3 3]
y =
     1     4
     6     7
     3     3
>>
>> all(y)
ans =
     1     1
```

Built-in Logical Functions

any(x)

- Returns true (1) if any element of x is true.
- Returns false (0) if all elements are false.

```
>> x = [1 7 4 5];
>>
>> any(x)
ans =
     1
```

```
>> x = [0 0 0 0];
>>
>> any(x)
ans =
     0
```

```
>> x = [1 7 4 0];
>>
>> any(x)
ans =
     1
```

```
>> x = [1 0 4; 0 0 3; 2 0 0]
x =
     1     0     4
     0     0     3
     2     0     0
>>
>> any(x)
ans =
     1     0     1
```

Built-in Logical Functions

find(x)

- Returns the indices of nonzero elements.

```
>> a = [2 5 0 0; 0 6 0 0]
a =
     2     5     0     0
     0     6     0     0
>>
>> find(a)
ans =
     1
     3
     4
```

```
>> x = [1 5 8; 3 6 5; 8 9 0]
x =
     1     5     8
     3     6     5
     8     9     0
>>
>> find((x == 5) | (x==6))
ans =
     4
     5
     8
```

Built-in Logical Functions

- See also:
  - isnan (checks for Not a Number)
  - isfinite (checks for not inf)
  - isinf (checks for inf)