Logical Data Type

- A special type of data that
  - can have one of only two possible values
    - true (displayed as 1) and false (displayed as 0)

- These values are produced by
  - special functions: true and false
    - $x = \text{true}$
    - $y = \text{false}$
    - $x = 1$
    - $y = 0$
  - relational and logical operators
Logical Data Type

- Logical and numerical values can be used in place of another
  - If a logical value is used in a place where a numerical value is expected
    - true values are converted to 1
    - false values are converted to 0

- a = true;
  b = a *3
  b = 3

- x = false;
  y/x
  ans = Inf
Logical Data Type

- Logical and numerical values can be used in place of another
  - If a numerical value is used in a place where a logical value is expected
    - non-zero values are converted to true
    - zero values are converted to false
  ```
  a = -5;
  a & true
  ans =
  1
  x = false;
  a = -5;
  y = (a > 0) | x
  y =
  0
  ```
Relational Operators

- Relational operators are used to represent conditions
  - such as “space $\leq 0$” and “result $\neq 25$”
- They take two numerical (or string) operands
- They yield a logical result (true or false)
Relational Operators

- The general form is: $a_1 \text{ op } a_2$
  - $a_1$ and $a_2$ are arithmetic expressions, variables, or strings.
  - op is one of the following

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>Equal to</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
</tbody>
</table>
## Relational Operators

<table>
<thead>
<tr>
<th>Operation</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 &lt; 4</td>
<td>true (1)</td>
</tr>
<tr>
<td>3 &lt;= 4</td>
<td>true (1)</td>
</tr>
<tr>
<td>3 == 4</td>
<td>false (0)</td>
</tr>
<tr>
<td>3 ~= 4</td>
<td>true (1)</td>
</tr>
<tr>
<td>3 &gt; 4</td>
<td>false (0)</td>
</tr>
<tr>
<td>4 &gt;= 4</td>
<td>true (1)</td>
</tr>
<tr>
<td>'A' &lt; 'B'</td>
<td>true (1)</td>
</tr>
</tbody>
</table>

Characters are evaluated in alphabetical order
Relational Operators

- Relational operators may be used
  - To compare a scalar value with an array
    - \( a = [1 \ 10; -2 \ 3]; \)
    - \( b = 3; \)
    - \( c = (a >= b) \)
    - \( c = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \)
  - To compare two arrays
    - \( x = [1 \ 10; -2 \ 3]; \)
    - \( y = [4 \ 5; 6 \ 7]; \)
    - \( x < y \)
    - \( \text{ans} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \)
    - \( x = [1 \ 10; -2 \ 3]; \)
    - \( y = [4 \ 5; 6 \ 7; 2 \ 0]; \)
    - \( x < y \)
    - ??? Error using \( == > < \)
    - Matrix dimensions must agree.
Relational Operators

- Don’t confuse equivalence (==) with assignment (=)
- Relational operations have lower priority than arithmetic operations
  - i.e., relational operators are evaluated after all arithmetic operators have been evaluated
  - You can use parentheses to be safe
    - $14 + 4 < 3 \times 5$
      - ans = 0
    - $(14 + 4) < (3 \times 5)$
      - ans = 0
Relational Operators

- Be careful about roundoff errors during numeric comparisons (== and ~= operators)
  - a = 0;
  - b = sin(pi);
  - a == b
  - ans = 0 (since sin(pi) calculation yields 1.2246e-016)

- You may use “abs(a-b) < small number” instead of “a == b”
  - abs(a-b) < 1.0E-14
    - ans = 1
  - abs(a-b) < eps
    - ans = 1
Logical Operators

- More complex conditions can be represented
  - by combining relational operations using logical operators
    - “temperature ≠ 25” AND “humidity < 60 %”
    - “exam grade < 45” OR “attendance ≤ 75”

- They take one or two logical operands
- They yield a logical result (true or false)
**Logical Operators**

- The general form of a binary logic operation: \( b_1 \, \text{op} \, b_2 \)
- The general form of a unary logic operation: \( \text{op} \, b_1 \)
  - \( b_1 \) and \( b_2 \) are expressions or variables
  - \( \text{op} \) is one of the following

<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Logical AND</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Logical AND with shortcut evaluation</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>xor</td>
<td>Logical exclusive OR</td>
</tr>
<tr>
<td>~</td>
<td>Logical NOT</td>
</tr>
</tbody>
</table>
Logical ANDs

<table>
<thead>
<tr>
<th>b1</th>
<th>b2</th>
<th>b1 &amp; b2</th>
<th>b1 &amp;&amp; b2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Differences between & and && operators
  - && only works between scalar values, while & works with either scalar and array values (arrays should be compatible)
  - && evaluates first b1 and then b2 only if b1 is true, otherwise it returns false immediately without evaluating b2, while & operator evaluates both b1 and b2 before returning an answer
### Logical ANDs

- Most of the time, it does not matter which AND operation is used.
  - \( b = 4; \ a = 8; \)
  - \((b \sim= 6) \& (a > 4)\)  
    \[
    \text{ans} = 1
    \]
  - \((b \sim= 6) \&\& (a > 4)\)  
    \[
    \text{ans} = 1
    \]

- If the comparison is between arrays, we have to use & operator
  - \(x = [\text{true false; false true}]; \ y = [\text{false false; true true}];\)
  - \(x \& y\)  
    \[
    \text{ans} =
    \begin{array}{c c c}
    0 & 0 & 0 \\
    0 & 1 & 1 \\
    \end{array}
    \]
  - \(x \&\& y\)
    
    ??? Operands to the | | and & & operators must be convertible to logical scalar values.

- Sometimes it is important to use shortcut expressions
  - \(b = 0; \ a = 4; \)
  - \((b \sim= 0) \& (a/b > 10)\)
    
    Warning: Divide by zero.
  - \((b \sim= 0) \&\& (a/b > 10)\)  
    \[
    \text{ans} = 0
    \]
Logical ORs

|   |   | b1 | b2 | b1 | b2 | b1 | b2 |
|---|---||||||||
| 0 | 0 | 0   | 0   | 0   | 0   | 0   | 0   |
| 0 | 1 | 1   | 1   | 1   | 1   | 1   | 1   |
| 1 | 0 | 1   | 1   | 1   | 1   | 1   | 1   |
| 1 | 1 | 1   | 1   | 1   | 1   | 1   | 1   |

- Differences between | and || operators
  - || only works between scalar values, while | works with either scalar and array values (arrays should be compatible)
  - || evaluates first b1 and then b2 only if b1 is false, otherwise it returns true immediately without evaluating b2, while | operator evaluates both b1 and b2 before returning an answer
Logical XOR and NOT

<table>
<thead>
<tr>
<th>b1</th>
<th>b2</th>
<th>xor(b1, b2)</th>
<th>~b1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Hierarchy of Operators

1) Parenthesis (starting from the innermost)
2) Exponentials (left to right)
3) Multiplications and divisions (left to right)
4) Additions and subtractions (left to right)
5) Relational operators (==, ~=, >, >=, <, <=) (left to right)
6) ~ operators
7) & and && operators (left to right)
8) |, ||, and xor operators (left to right)
Matlab includes a number of logical functions, which can be used with relational and logical operators

<table>
<thead>
<tr>
<th>Function</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ischar(a)</td>
<td>Returns true if ( a ) is a character array</td>
</tr>
<tr>
<td>isempty(a)</td>
<td>Returns true if ( a ) is an empty array</td>
</tr>
<tr>
<td>isinf(a)</td>
<td>Returns true if the value of ( a ) is ( Inf ) (infinite)</td>
</tr>
<tr>
<td>isnan(a)</td>
<td>Returns true if the value of ( a ) is ( NaN ) (not a number)</td>
</tr>
<tr>
<td>isnumeric(a)</td>
<td>Returns true if ( a ) is a numeric array</td>
</tr>
</tbody>
</table>
Examples

- To count the occurrence of digits in a sentence (represented by array B)
  - \((B(i) \geq '0') \& (B(i) \leq '9')\)

- To count the occurrence of letters in a sentence (represented by array B)
  - \(((B(i) \geq 'a') \& (B(i) \leq 'z')) \mid \ldots \mid ((B(i) \geq 'A') \& (B(i) \leq 'Z'))\)
Examples

- To determine if a quadratic equation has two distinct real roots
  - \((b^2 - 4ac) > 0\)

- To determine if the specified year is a leap year
  - \(\text{mod(year,4)} == 0 \ & \ ...\)
    - \((\text{mod(year,100)} \sim = 0 \ | \ ...\)
    - \(\text{mod(year,400)} == 0)\)