A logical operator is used to write a statement whose answer is a **boolean**: that is, the answer is either true (1) or false (0). The following are boolean operators:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Less than** | **Greater than** | **Less than or equal to** | **Greater than or equal to** | **Equal to** | **Not equal to** |
| < | > | <= | >= | == | ~= |

Note that the **boolean equal to ‘==’** is different from ‘=’, which is used to assign values to variables (eg, x = 3).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| You can combine boolean operators to create more specific conditional statements using ‘and’/‘or’. | | | | **And** | **Or** |
| && | || |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ex 1.**  x = 2; y = 2;  x+y < 5  ans =  logical  1 | | **Ex 2.**  x = 2; y = 2;  x+y < 1 || x == y  ans =  logical  **1** | | **Ex 3.**  x = 2; y = 2;  x+y ~= 4 && 2\*x == 4 || y >= x  ans =  logical  **1** | |

The **if statement** directs Matlab to execute certain lines of code depending on whether a certain condition is satisfied. That condition is usually specified using a boolean statement. If the first boolean condition is not true, then you may specify any number of additional boolean statements using elseifs. Lastly, you have the option to specify an else, which will execute if neither the first if or following elseif boolean statements are true.

|  |  |
| --- | --- |
|  | if <boolean statement>  <executable>  elseif <boolean statement>  <executable>  elseif <boolean statement>  <executable>  …  else  <executable>  end |

|  |  |  |
| --- | --- | --- |
| **Ex 1.**  function y = f(x)  if x ~= 0  y = 1/x;  end  end | **Ex 2.**  function y = g(x)  if x >= 2  y = x+2;  else  y = x-2;  end  end | **Ex 3.**  function inform = reactor(T)  if T > 100 || T < 0  instruction = ‘Danger’;  elseif T > 40 && T < 75  instruction = ‘Optimal T’;  else  instruction = ‘Acceptable T’;  end  end |
| **f(3) = 0.333**  **f(0) = no output is assigned to y; calling f(x) will not throw an error, but setting y = f(x) will since no output value gets assigned to y** | **g(2) = 4**  **g(-3.8) = -5.8** | **reactor(20) = ‘Acceptable T’ reactor(150)= ‘Danger’**  **reactor(40) = ‘Acceptable T’ reactor(-10)= ‘Danger’** |

**Nested If Statements**

Sometimes you will need to specify conditions within other conditions (ie., a domain within a domain). You can do this by nesting if statements.

|  |  |
| --- | --- |
|  | if <boolean statement>  <executable>  if <boolean statement>  <executable>  elseif <boolean statement>  <executable>  end  else  <executable>  end |

|  |  |  |
| --- | --- | --- |
| **Ex 1.**  function source = refuel(tank\_1, tank\_2)  if tank\_1 < 50  tank\_1 = tank\_1 + 50;  if tank\_1 > tank\_2  source = ‘Use tank 1’;  elseif tank\_1 < tank\_2  source = ‘Use tank 2’;  end  else  source = ‘Use tank 1’;  end | | If tank 1 has less than 50 units,  Add 50 units of fuel to the existing fuel in tank 1.  If tank 1 now has more fuel than tank 2,  Use tank 1.  If tank 1 still has less fuel than tank 2,  Use tank 2.  Otherwise, (ie, if tank 1 didn’t have less than 50 fuel units in the first place)  Use tank 1. |
| **refuel(30,70) = ‘Use tank 1’** | **refuel(50,50) = ‘Use tank 1’** |  |