# LOGICAL OPERATIONS AND FLOW CONTROL:

## RELATIONAL OPERATORS:

Often we want to compare values of different variables in order to determine what to do next. There are many relational operators that help us compare values.

| | |
|---|---|
| > | greater than |
| < | less than |
| == | equal to (Notice the DOUBLE equal sign. One equal sign is the assignment operator. |
| >= | greater than or equal to |
| <= | less than or equal to |
| ~= | not equal to |

The relational operators above allow us to compare alphanumeric data. However, it we want to compare logical data, we need to use logical operators

| | |
|---|---|
| & | and |
| \| | or |
| ~ | not |

Let's go through some examples of using relational operators.

```
> x = 5.4
x =  5.4000
> y = 3.9
y =  3.9000
> x == y
ans = 0                 (0 means false)
> x > y
ans =  1                (1 means true)
> x >= y
ans =  1
> x ~= y
ans =  1
> x > y/50
ans =  1
```

When we compare two pieces of alphanumeric data Matlab/Octave returns 0 (false) or 1 (true).

Often we want to do some action if two or more conditions are met. For example, if you have done your homework and it is earlier than midnight, you will go to Taco Bell for a snack. This will require the use of logical operators. Here are the rules for using logical operators:

| | | |
|---|---|---|
| True and True → True | 1 & 1 → 1 | (are both statements true? true!) |
| True and False → False | 1 & 0 → 0 | (are both statements true? false!) |
| False and True → False | 0 & 1 → 0 | |
| False and False → False | 0 & 0 → 0 | |

| | | |
|---|---|---|
| True or True → True | 1 \| 1 → 1 | (is either statement true? true!) |
| True or False → True | 1 \| 0 → 1 | (is either statement true? true!) |
| False or True → True | 0 \| 1 → 1 | |
| False or False → False | 0 \| 0 → 0 | |

| | | |
|---|---|---|
| not True → False | ~1 → 0 | (the opposite of true? false!) |
| not False → True | ~0 → 1 | |

```
> 7==7 & 6>7          (true and false → false)
ans = 0
> 7==7 & 6<7          (true and true → true)
ans =  1
> 7==7 & 6>7
ans = 0
> 7==7 | 6<7          (true or false → true)
ans =  1
> 7==7 | 6>7
ans =  1
> 7==9 | 9>10
ans = 0
> ~(7==9)             (not false → true)
ans =  1
> ~(7==7)
ans = 0
```

Here are some examples with variables. It is a good idea to use parentheses to avoid confusion

```
>  x = 100;  y = 8.1;
>  (x>y) | (y==8.1)
ans =  1
>  ( (y*3)<(x*4) ) & (x>5)
ans =  1
```

## Comparing data in arrays:

You can compare entire arrays of data too.

```
> quiz1 = [99,88,77];
> quiz2 = [90,90,89];
```

```
> quiz1 < quiz2
ans =   0   1   1
```

An array of true/false values is returned. These can be stored in a logical array.

```
> a = quiz1 < quiz2
a =   0   1   1
```

You can compare all elements in an array to a single value.

```
> b = quiz1 > 79
b =   1   1   0
```

```
> whos
Variables in the current scope:
  Attr   Name        Size                Bytes  Class
  ====   ====        ====                =====  =====
         a           1x3                     3  logical
         b           1x3                     3  logical
         quiz1       1x3                    24  double
         quiz2       1x3                    24  double
```

See… **a** and **b** are logical arrays.

## IF STATEMENTS:

Often we want to execute a command only if a certain test condition is met. We use **if** statements to do this. There are a few different types of **if** statements.

### (A) Simple if statement

If you want to execute commands when a certain test condition is met, use a simple **if** statement.

```
if  ( test condition )
   statements
end
```

You don't need the parentheses, but it is a good idea to avoid confusion and mistakes.

In mfile.m:

```
x = 5.4
y = 10
if (x>4)              % if the test condition is true, the following two statements will be executed
  y = y + 1;          % if the test condition is false, the statements will not be executed
```

```
  x = 2;
end
x
y
```

At the command line:

```
> mfile
x = 5.4000
y =  10
x =    2
y =  11
```

**(B) If/Else statement**

An **if/else** statement will allow you to execute certain statements **if** a test condition is true and execute other statements **if** a test condition is false.

In mfile.m:

```
x = 5
y = 10
if (x>y)
  y = x                 % Execute the next two lines if the test condition is true
  z = 100;
else
  z = 200;              % Execute this line if the test condition is false
end
x
y
z
```

At the command line:

```
> mfile
x =  5
y =  10
x =  5
y =  10
z =  200
```

**(C) If/Elseif statement:**

If you have 2+ possible conditions that can be met, you can use an **if/elseif** statement.

In mfile.m:

```
x = 5;
if (x<0)                    % This is false, so the following commands are not executed
  z = 100;
  beep
elseif (x>=0 & x<10)        % This is true, so the following commands are executed
  z = 200;                  %     and the if statement ends
  beep, beep
elseif (x<=10)              % This is true, but is not executed
  z = 300;
  beep, beep, beep
end
z
```

At the command line:

```
> mfile
z =  200                (with 2 beeps)
```

Note that once a test condition is found to be true, the statements after the test condition are executed and the **if/elseif** statement ends even if later test conditions are true too.

**NESTED IF STATEMENTS:**

You can put **if** statements within **if** statements.

In mfile.m:

```
score = input('Enter your score: ');
disp('Your grade is:')
if(score<=100 & score>=90)
   grade(1) = 'A';
   if(score>=90 & score<93)
            grade(2) = '-';
         elseif(score>=93 & score<97)
            grade(2) = ' ';
         elseif(score>=97 & score<=100)
      grade(2) = '+';
         end
         disp(grade)
else
   disp('B or worse')
end
```

At command line:

```
> mfile
Enter your score: 93.2
```

```
Your grade is:
A
```

**Another example:**

In M-file:

```
score = input('What was your test score?: ');
age = input('How old are you?: ');
if(score>90)
  if(age<=18)
    disp('You did very good... for a young person')
  elseif(age<=21 & age>18)
    disp('You did pretty good... for a young person')
  else
    disp('You are old')
  end
else
  disp('Your score is too low')
end
```

Output:

```
What was your test score?: 91
How old are you?: 14
You did very good... for a young person

What was your test score?: 89
How old are you?: 55
Your score is too low
```

**(D) Switch and Case**

This is similar to **if/elseif** but may be more useful if there are multiple options. A useful application of **switch** is when comparing character strings. The general format of **switch** and **case** is the following.

```
switch variable
   case {options}
      statements
   case {options}
      statements
    …
   otherwise
      statements
```

end

The braces **{ }** aren't strictly necessary, but they help keep your code organized.

In mfile.m:

```
animal = 'dog';
switch animal
  case {'cat'}
    disp('meow')
  case {'dog' , 'canine'}
    disp('woof')
  case {'sheep'}
    disp('baaaa')
  case {'duck'}
    disp('quack')
  otherwise
    disp('I have no idea')
end
```

At the command line:

```
> mfile
woof
```