ENME202 Matlab

CONDITIONALS

IF / ELSEIF / ELSE

The "if" statement is a foundational element of programming, allowing us to decide on different courses of action based on the state of a variable or other data element with our code.

Syntax:

```
%   if LOGICAL_EXPRESSION
%     DO_IF_TRUE
%   end
```

Example:

```
x = sqrt(28)/5.6;
if x > 2
  disp('x is greater than 2')
end
```

We can also define a path that occurs only when the argument in the if statement is false using if/else:

```
if x > 2
  disp('x is greater than 2')
else
  disp('x is NOT greater than 2')
end
```

```
x is NOT greater than 2
```

Example: compute the total cost of buying N widgets, if the cost per widget is $0.90 for N<=50 but $0.70 for N>50.

```
N = 56;
if ( N <= 50 )
  totalCost = 0.90*N;
else
  totalCost = 0.70*N;
end
```

Another version of the same example:

```
if ( N <= 50 )
  costPerWidget = 0.90;
else
  costPerWidget = 0.70;
end
totalCost = costPerWidget * N;
```

Yet another version, without an else statement:

```
totalCost = 0.90 * N ;
if ( N > 50 )
  totalCost = 0.70 * N ;
end
```

The elseif statement enables >2 possible outcomes (vs. just 2 with if/else):

```
if x > 10
  disp('x is greater than 10')
elseif x >= 5
  disp('x is between 5 and 10, inclusive')
elseif x == 0
  disp('x is zero')
else
  disp('x is less than 5 and non-zero')
end
```

```
x is less than 5 and non-zero
```

If statements can use complex Boolean expressions as the conditional, for example:

```
x = 10*rand();     % random number from 0-10
if ((x > 2) & (x < 5)) | x > 9.9
  disp(x)
end
```

Below are several exaples showing the use of if statements in different contexts, combined with functions and for() loops.

Example 1

Compute the value of a piecewise function of x. Use an if() statement with appropriate logical expressions to break down each piece of the function:

```

if (x<-1)
  y = 1;
elseif (x>=-1) & (x<=2)
  y = 4;
elseif (x>2) & (x<3)
  z = x-2;
  y = 4-2*z^2;
elseif (x>=3)
  z = x-3;
  y = sqrt(z);
end
```

Note: be careful when setting up the conditions for situations like this to make sure that all cases (here, all possible values of x) have been covered. Otherwise, y will not have a proper numerical value assigned to it, and you will get errors.

Clauses in conditionals are evaluated from top to bottom in the order they appear. Only the code block in the first "true" clause found is executed, the others are skipped.

Knowing this we can often simplify our conditions by changing the order in which they are tested. For example, the elseif/else statements below are equivalent to the previous example:

```
if (x<-1)
  y = 1;
elseif (x<=2)  % if we get here we already know x>=-1
  y = 4;
elseif (x<3)  % if we get here we already know x>2
  z = x-2;
  y = 4-2*z^2;
else          % if we get here we already know x>=3
  z = x-3;
  y = sqrt(z);
end
```

Example 2

The power of your code increases vastly by *combining* different techniques. Here is an example of using a loop and a conditional together, to evaluate our piecewise function at an *array* of points x:

```

x = [10 -9 8 -7 6 -5];
for k = 1:length(x)
  if x(k)<-1
    y(k) = 1;
  elseif x(k)<=2
    y(k) = 4;
  elseif x(k)<3
    z = x(k)-2;
    y(k) = 4-2*z^2;
  else
    z = x(k)-3;
    y(k) = sqrt(z);
  end   % if
end   % for
disp(y)
```

```
  Columns 1 through 4

    2.6458    1.0000    2.2361    1.0000

  Columns 5 through 6

    1.7321    1.0000
```

SWITCH / CASE

The switch() statement provides another conditional that makes it easy to choose different outcomes based on the value of an input. Only the first matching case will be activated (this is very different from other languages like C++)

```
color = 'red';   % the quotes denote a variable of type "string"

switch color
```

```
  case {'Blue' 'blue'}  % use {} for multiple cases
    disp('blue sky')
  case 'red'
    disp('red fire')
  case 'green'            % more than one line of code can run per case
    disp('green tree')
    disp('green tea')
  otherwise               % do if no case comparison is true (optional)
    disp('Not blue, red, or green')
end
```

```
red fire
```

Note that the "otherwise" statement is optional. Also note that a switch statement can always be performed using if/else statements, but if/else offers functionality that cannot be replicated using switch.

FLAGS

If statements make it possible to use Boolean variables as "flags". A flag is a true/false variable (i.e. a Boolean) that can be flipped "on" or "off" to keep track of the state of some property within our code. We often need use a conditional to decide what value to assign to the flag variable.

For example, here is a loop that checks to check if an array contains any even values:

```
a = [1.2 4.33  5 9  7.1  8.1 21 -3];

% Create a flag to hold the state of the query (initialize
% as false, i.e. assume there are no even values)
isEven = false;

for i = a
  if mod(i,2) == 0
    isEven = true;
  end
end

if isEven
  disp('There was at least one even value')
else
  disp('There were no even values')
end
```

```
There were no even values
```