Department of Chemical and Biomolecular Engineering

MATLAB Tutorial - Programming Logic Structures (for loops, while loops, if/elseif/else)

These tutorials are designed to show the introductory elements for any of the topics discussed. In almost all cases there are other ways to accomplish the same objective, or higher level features that can be added to the commands below.

Any text below appearing after the double prompt (>>) can be entered in the Command Window directly or in an m-file.

==

This tutorial contains the following sections;

==

Introduction

Like other programming languages, MATLAB allows the control of command execution based on a decision making structure. MATLAB has three structures to control decision making:

for loops
while loops
if-else-end structures

Each of these structures often involves numerous commands. Therefore, they should always be put in m-files. Information related to the use of these commands is located in Chapter 11 of the MATLAB manual.

==

for Loops

for loops allow for a group of MATLAB commands to be repeated a fixed, predetermined number of times. The general form is:

```
for x = array
     group of commands
end
```

The commands are executed once for every column of the array. At each iteration, x is assigned *to the next column of the array. It is best to use an integer counter array.*

```
for n = 1:6
    x(n) = 2^n (Note that the MATLAB prompt ( >> ) doesn't appear.)
end
x =
    2
x =
   2   4
x =
   2   4   8
x =
   2   4   8   16
x =
   2   4   8   16   32
x =
   2   4   8   16   32   64
```

Notice that MATLAB prints x at every loop. You should suppress printing during the calculation, then print ***x*** *after exiting the loop.*

Write the simple m-file given here.

```
for n = 1:6
    x(n) = 2^n;
end
x
```

The Command Window output looks like this

```
x =
   2   4   8   16   32   64
```

You cannot short-circuit the loop by reassigning the loop variable, n, within the loop.

While using an array such as ***n = 1:6*** *is the most common way of using a for loop, any array can be used. Here is a short m-file.*

```
data  =  [ 3  9   45  6 ];
for n = data
    result = n*3
end
```

Here is the output

```
result =
    9
result =
   27
result =
   135
result =
   18
```

While this is possible, it is not recommended.   Use the following structure to accomplish the same calculation.

```
data = [ 3   9   45   6 ];
for n = 1:4
     result(n) = data(n)*3;
end
>> result
result =
   9   27   135   18
```

By keeping 'n' as an integer loop counter it is much easier to organize and access arrays.

***for loops** can be nested, ie. stacked within each other.*

```
for n = 1:4
    for m = 1:3
        c(n,m) = 2*n + 3*m;
   end
end

>> c
c =
       5    8    11
       7   10    13
       9   12    15
     11   14    17
```

*Note that **for loops** are not necessary for all repeated calculations. In some cases the element-by-element dot notation can be used.*

```
 n = 1:6;
 x = 2 .^ n        % Note the ( .^ ) notation.
x =
   2   4   8   16   32   64
```

==

while Loops

As opposed to **for loops** which execute a fixed number of times, **while loops** evaluate a group of commands an indefinite number of times. The general form of the while loop is:

```
while expression
     commands
end
```

The commands between the **while** and **end** statements are executed as long as all elements in **expression** are True.

==

Example Problem - while Loop

The following example increases a number by 50 percent until the value exceeds 100.

- Click on FILE / NEW / M-FILE. If you are editing a previously created m-file click on FILE / OPEN M-FILE, then select the file from the list. Alternately you can choose the file from the quick list at the bottom of the FILE menu.
- Type the following list of MATLAB commands They aren't started with >> because the MATLAB prompt doesn't appear in an m-file.

Here is the m-file

```
clear         %   This statement clears any old values. (not required, but a good idea
n = 1;         %  n and t are initialized
t =1;
value(n) = t;
while t < 100         %  The indenting is only to clarify where the while loop begins and ends
   t = t*1.5;
   n = n + 1;
   value(n) = t;
end
value'                  %  This statement displays the created array of values
```

- Save the m-file to the any directory with any name you want.
- Return to the MATLAB workspace, change to the proper directory and enter the following:

```
>> filename
value =
   1.0000
   1.5000
   2.2500
   3.3750
   5.0625
   7.5938
   11.3906
   17.0859
   25.6289
   38.4434
   57.6650
   86.4976
   129.7463
```

==

Breaking Out of for and while loops
You can jump out of loops by using the break command. Inside the loop you would use a section of code looking something like this;

```
if t > 100
  break
end
```

If your not in a loop the **break** command stops execution of the program.
==

if-else-end Structures
This structure causes a sequence of commands to be conditionally evaluated based on a relational test. In the simplest form we decide whether or not to evaluate a list of commands. In more advanced structures we can choose which of many command lists to evaluate.

The simplest if-else-end structure is:

```
if expression
    commands
end
```

The commands are evaluated if all elements of expression are True.

For two alternatives the structure is:

```
if expression
    commands evaluated if expression is True
else
    commands evaluated if expression is False
end
```

For three or more alternatives:

```
if expression1
    commands evaluated if expression1 is True
elseif expression2
    commands evaluated if expression2 is True
elseif expression3
    commands evaluated if expression3 is True
else
    commands evaluated if no expression is True
end
```

==

Example Problem - for Loop and if-else-end Statement

Type the following commands into an m-file. Remember to use semi-colons to suppress display until the calculations are complete. Indenting is not necessary, but helps the readability.

```
for n = 1:20
   if n < 6
       x(n) = n;
   elseif n < 11
      x(n) = 2*n;
   elseif n < 16
      x(n) = 3*n;
   else
      x(n) = 4*n;
   end
end
x
```

Save the file as test.m. and return to the Command Window.

```
>> test
x =
Columns 1 through 12
  1   2   3   4   5   12   14   16   18   20   33   36
Columns 13 through 20
  39   42   45   64   68   72   76   80
```

Note that once an expression is ***True****, all following expressions are ignored even if they are also* ***True****. When n < 6 the n < 11, n < 16 etc. expressions are ignored. Note that you need an* ***end*** *statement for both the* ***if structure*** *and the outer* ***for loop****.*