

Math 338: Programming

1. Logical and relational tests

Relational Operators		
operator	usage	meaning
<	$a < b$	is a less than b?
<=	$a \leq b$	is a less a \leq b?
>	$a > b$	is a greater than b?
>=	$a \geq b$	is a \geq b?
==	$a == b$	is a equal to b?
~=	$a \sim b$	is a \neq b?

Logical Operators		
operator	usage	meaning
&	$A \& B$	are statements A and B <u>both</u> true?
	$A B$	is <u>either</u> statement A <u>or</u> statement B true?
~	$\sim A$	is the negation of statement A true?

Typically, logic and relational statements return a 1 for true statements and a 0 for false statements:

```
>> 1<2
```

```
ans =
```

```
1
```

```
>> 1>2
```

```
ans =
```

```
0
```

This works on vectors also:

```
>> x=[0:5]
```

```
x =
```

```
0 1 2 3 4 5
```

```
>> x<3
```

```
ans =
```

```
1 1 1 0 0 0
```

Notice that the `x < 3` command checked to see whether each entry of `x` was less than 3. More will be said about this later when we talk about “vectorizing” `if...end` statements. You may also type `help relop` to get more information about these operators.

2. Loops

There are two kinds of loops in Matlab: **for...end** loops and **while ...end** loops. Using **for** gives you a predetermined number of loops to run. Using **while** will run a loop until a certain condition is satisfied.

Example: A **for** loop in a function m-file

```
function y = chap03ex01(n)
%This is chap03ex01.m
%This will compute n!

y = 1;
for i = 1:n
    y = y*i;
end
```

Example: A **while** loop in a function m-file

```
function y = chap03ex02(n)
%This is chap03ex02.m
%This will compute some factorials also

y = 1;
max_it = n;
test = 1;
while test <= max_it
    y = y*test;
    test = test + 1;
end
```

Notice that `max_it` controls how many times the **while** loop will be executed. Each time through the loop, `test` is increased by one. Once the statement `test <= max_it` is false, the loop will no longer execute.

3. Decision Making and Logic Statements

There are two kinds of decision making commands: **if...end** and **switch...end**.

Example: `chap03ex03.m` – the **if** command

```

%This is chap03ex03.m
%A simple script with if...else...end
age = input('What is your age?');
name = input('What is your name?', 's');
%For the young people
if age < 18
    disp(['You are too young, ', name])
end
%For the students
if (age >= 18) & (age < 23)
    disp(['You must be a student, ', name])
    disp(['You are ', num2str(age), ' years old'])
else
    disp([name, ' you must not be between 18 and 23 years old.'])
end
% For everyone else
if (age < 25)
    disp('You are younger than 25 years old')
else if (age >= 25)
    number = age - 25;
    disp(['You are ', num2str(number), ' years past 25 years old, ', name])
end
end
end

```

Comments:

- Using **if** requires at least a matching **end**.
- The other versions are

if test	if test
statement	statement
else	elseif test
statement	statement
end	end

- There can be as many **elseif** conditions as you want between the **if** and the **end**.
- If there is an **else** then it must come after all of the **elseif** conditions and before the **end**.

Example: the **switch** command in a script m-file:

```

%This is chap03ex04.m
%This example script m-file uses the switch command.

```

```
clear all
```

```

%ask for a function to plot
%user should input 1 or 2
fun_num = input(['What do you want to plot? \n(1) sin(x) \n(2) cos(x) \nyour choice?
']); % the \n simply means put the following on a new line

x = [0:0.01:1];
switch fun_num % this is the value we are testing
    case 1 % this says that fun_num == 1
        y=sin(x);
    case 2 % this says that fun+num == 2
        y=cos(x);
    otherwise % this says fun_num is something else
        warning('You did not pick 1 or 2')
end
plot(x,y)

```

4. Breaking out of loops and function m-files

Sometimes you may want to jump out of a loop, or a function m-file early. To do so requires the **break** and **return** commands. The **break** command will “break” you out of a loop and start with the first statement after the loop.

Example: Breaking out of a loop

```

% This is chap03ex05.m
% This m-file determines at what point y=x^2
% is equal to or greater than a value of 3.5

x=[0:0.001:10];
m=length(x); % tells how long x is
y = x.^2;

%%%%%%%%%%
for i = 1:m % this loop will cycle through
            % all of the x-values

    if y(i)>=3.5 % tests whether y >= 3.5
        last = i;
        break % breaks out of loop
    end

end % end of loop
%%%%%%%%%%
fprintf(['\ny=%5.3f when x=%5.3f'], y(last), x(last))

```

Notice that even though the **break** is inside of the **if...end** statement, it still breaks out of the **for...end** loop. Also notice that this loop has a definite stopping point since the value of **i** will not go past **m**.

Example: Using **break** with a **while** loop

```
% This is chap03ex06.m
% This m-file determines at what point y=x^2
% is equal to or greater than a value of 3.5

x = 0;
y = x;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while y>=0                % will loop as long as y>=0

    if y>=3.5              % tests whether y >= 3.5
        break            % breaks out of loop
    end
    x = x+0.001;          % increments x-value
    y = x.^2;            % calculates new y-value

end                        % end of loop
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf(['\ny=%5.3f when x=%5.3f'], y, x)
```

In this m-file, the loop would continue endlessly if the **break** statement were removed. It is usually good to have some sort of **break** in a **while** loop so that if things go wrong it will stop for some reason.

The **return** statement works like a **break** but instead it causes a function m-file to stop and Matlab “returns” to the place where the m-file was called.

Example: Using **return** in an m-file.

```
function y = chap03ex07(n)
% This is chap03ex07.m
% It should compute n! where n is a
% nonnegative integer.
%
% usage:
% y = chap03ex07(n)
%
% inputs:
% n -- some integer
% outputs:
% y -- should equal n!
```

```

%
% Note: this m-file calls chap03ex01.m

if n<0
    warning('n<0 So n! cannot be computed');
    y=0;
    return
end
y = chap03ex01(n);

```

5. Optional inputs, outputs and default values

The built-in variables **nargin** and **nargout** and the logical relation **isempty** can be used to alter the way inputs have to be given to a function m-file or give the user the choice of default values.

Example: Using isempty

```

% This is chap03ex08.m
% Another name program
% It will ask for someone's name
% five times and then quit.

name = [];
flag = 0; loop = 0;
while isempty(name)
    if loop >=5
        disp('I wish you would have told me your name :(')
        break
    end
    if flag == 0
        name = input('What is your name? ','s');
    else
        name = input('No really, what is your name? ','s');
    end
    if ~isempty(name)
        disp(['Hi ',name,' it is a pleasure to meet you'])
    end
    loop = loop + 1;
    flag = 1;
end

```

The **isempty(name)** condition is asking if name is empty or not. If it is empty then the loop will continue to run. The **flag** variable is used to tell the script if it is in the first loop or not. The *loop* variable is used to determine if too many loops have been used.

Example: Using **nargin** in a function m-file.

```
function [y,x] = chap03ex09(a,b,c,h)
% This is chap03ex09.m
% usage:
% [y,x] = chap03ex09
% [y,x] = chap03ex09(a,b)
% [y,x] = chap03ex09(a,b,c)
% [y,x] = chap03ex09(a,b,c,h)
%
% inputs:
% a -- the left endpoint of the interval
% b -- the right endpoint of the interval
% Note: a and b are optional but must
% be input together or not at all.
%   (defaults are a=0 and b=1);
% c -- This is the location of the jump
%   (default is 0.5);
% h -- This is the step-size to use in the
%   vector x (default = 0.1);
% outputs:
% y -- the values of the function that
%   correspond to the x-values
% x -- The values where y is output
%   x = [a:h:b];
%
% Note: nargin stands for 'Number of ARGuments IN'
%
if nargin < 4      % are there less than four inputs?
    h = 0.1;
end
if nargin < 3      % are there less than three inputs?
    c = 0.5;
end
if nargin < 1      % are there no inputs?
    a = 0; b = 1;
end

% there is a problem if a > b
if a > b
    warning('I can"t do that! a must be less than b!')
    x=[];y=[];
    return
end
```

```
x = [a:h:b];  
m = length(x);  
y = zeros(size(x)); % or y = zeros(1,m);
```

```
for i = 1:m  
    if x(i) > c  
        y(i) = sin(x(i));  
    end  
end
```

Note: function m-files that have no inputs must be called without parenthesis. In other words, whenever parenthesis are used in calling a function m-file, Matlab expects there to be inputs of some type.

```
>>[y,x] = chap03ex09
```

instead of

```
>>[y,x] = chap03ex09()
```