L06: Branching

If-statements

Lucas A. J. Bastien

E7 Spring 2017, University of California at Berkeley

January 30, 2017

Version: release

Announcements

Lab 02 is due on February 3 at 12 pm

Today:

- Branching (if-statements)
- Comparing character strings (`strcmp` and `strcmpi`)
- The `fprintf` function

Wednesday:

- Matrices and matrix multiplication (maybe on chalk board)

Friday:

- Mostly practice questions

Your written feedback

Many of you said that:

\+ Lectures are well organized; I am well prepared

\+ You like the diaries

\+ You appreciate the questions and interactions

\− It goes too fast

You will learn a lot in E7 this semester
I will try to post slides and diaries earlier

\− I switch between slides and Matlab too quickly

I will try to improve

You want more practice questions and problems

I will keep asking many questions in class
Remember that you have 4 hours of lab section per week!
In lab section, you will spend time practicing concepts seen in lecture

What is branching?

We as humans often make decisions based on certain conditions

For example:

When I go to work in the morning, **if** it is raining outside **then** I take my umbrella with me. If it is not raining, then I don't take my umbrella

Here, my decision depends on the weather

Similarly, computer programs often need to take different actions depending on certain conditions

Image by Acagastya, retrieved from https://en.wikipedia.org/wiki/File:Water_Drop_on_rose_leaf.JPG on January 29 2017

Syntax of if-statements

```
if condition1
    % What to do if condition1 is true
elseif condition2
    % What to do if condition2 is true
else
    % What to do otherwise
end
```

Notes:

- `condition1` and `condition2` must be expressions that evaluate to logical values (either true or false)
- **Important:** Statements under `condition2` will not be executed if `condition1` is `true`, even if `condition2` if `true`. **Only the first clause which condition is true is executed**
- There can be as many `elseif` clauses as desired
- `elseif` and `else` clauses are optional

if-statements: example 1

```
function [result] = my_cos_1(theta, is_degrees)
% Calculate cosine of theta. If is_degrees is true,
% assume that theta is in degrees. If is_degrees is false,
% assume that theta is in radians.
%
% In this implementation of the function, we use an if-
% statement with an else clause

if is_degrees
    result = cosd(theta);
else
    result = cos(theta);
end

end
```

if-statements: example 2

```
function [result] = my_cos_2(theta, is_degrees)
% Calculate cosine of theta. If is_degrees is true,
% assume that theta is in degrees. If is_degrees is false,
% assume that theta is in radians.
%
% In this implementation of the function, we use an if-
% statement without an else clause. The variable theta keeps
% its original value if is_degree is false

if is_degrees
    % Convert to radians
    theta = theta * pi / 180;
end

result = cos(theta);

end
```

if-statements: example 3

```
function [result] = my_cos_3(theta, is_degrees)
% Calculate cosine of theta. If is_degrees is true,
% assume that theta is in degrees. If is_degrees is false,
% assume that theta is in radians.
%
% In this implementation of the function, we use an if-
% statement with an else clause. The if-statement is used
% to create a handle to the appropriate function to use, given
% the units of theta

if is_degrees
    cos_function = @cosd;
else
    cos_function = @cos;
end

result = cos_function(theta);

end
```

if-statements: practice question 1

What will the value of the variable "hot" be after executing the following code?

```
>> temperature = 85;
>> if temperature > 90
>>     hot = true;
>> else
>>     hot = false;
>> end
```

(A) true

(B) false

if-statements: practice question 2

What will the value of the variable "hot" be after executing the following code?

```
>> temperature = 85;
>> hot = false;
>> if temperature > 90
>>     hot = true;
>> end
```

(A) true

(B) false

(C) Matlab will throw an error because there is no `else` clause

Note: the `else` clause is optional; the code above is correct

if-statements: practice question 3

What will the value of the variable "warm" be after executing the following code?

```
>> temperature = 95;
>> if 70 < temperature < 90
>>     warm = true;
>> else
>>     warm = false;
>> end
```

(A) true

(B) false

Relational operators of equal precedence are evaluated from left to right. In the example above, "`70 < temperature`" evaluates to logical 1 (true). The remaining expression is "`1 < 90`", which evaluates to true. Correct code:

```
70 < temperature & temperature < 90
```

Comparing character strings

Character strings are arrays. Using the relational operator == between two character strings will result in an element-wise comparison:

```
>> 'abcde' == 'aaaee'
ans =
  1x5 logical array
   1   0   0   0   1
```

and will result in an error if the strings have different lengths:

```
>> 'abcde' == 'abc'
Matrix dimensions must agree
```

DO NOT use the relational operator == to check whether two character strings are the same!

Comparing character strings (continued)

Instead, use the functions `strcmp` (case sensitive) or `strcmpi` (case insensitive). For example:

```
>> strcmp('abcde', 'abc')
  logical
    0
>> strcmp('abcde', 'abcde')
  logical
    1
>> strcmp('abcde', 'AbCdE')
  logical
    0
>> strcmpi('abcde', 'AbCdE')
  logical
    1

>> % The functions "lower" and "upper" may be useful
>> lower('AbCdE') % Convert character string to lower case
    abcde
>> upper('AbCdE') % Convert character string to upper case
    ABCDE
```

The fprintf function

Use the function `fprintf` to print information into a file (later in the semester) or to screen. Examples for printing to screen:

```
>> fprintf('Hello World!\n')
Hello World!

>> fprintf('Hello\nWorld!\n')
Hello
World!

>> a = 2;
>> b = 3;
>> fprintf('%d multiplied by %d equals %d\n', a, b, a*b);
2 multiplied by 3 equals 6
```

Notes:

- `\n` is used to add a "newline" character
- Matlab replaces the tokens that start with % by the values listed as the second, third, ..., arguments

The fprintf function (continued)

Useful conversion specifications (*i.e.* the tokens that start with %):

%d	integer number
%f	floating point number, decimal notation
%.10f	floating point number, decimal notation with ten decimals
%e	floating point number, scientific notation
%.10e	floating point number, scientific notation with ten decimals
%s	character string

Example:

```
>> fprintf('%s\n%f\n%.10f\n%e\n%.10e\n', 'pi=', pi, pi, pi, pi)
pi=
3.141593
3.1415926536
3.141593e+00
3.1415926536e+00
```

The sprintf function

The syntax of using `sprintf` is similar to the syntax of using `fprintf` to print information to screen. The difference between these two functions is that `fprintf` prints information to screen or to a file, whereas `sprintf` returns a character string. For example:

```
>> my_string = sprintf('The value of pi is %.15f', pi)

my_string =

The value of pi is 3.141592653589793
```