ENME 303 LAB

Week 2: If Statement

TA: Karla Negrete

UMBC

Week 2: Control Flow I

I. Refresher
II. Relational/Logical Operators
III. Control Structure
IV. Branch Statements: If-statements

I. Refreshers: Administrative

01. Assignments are ONE .m file with various exercises within
    a. You can use %% [SPACE] [EXERCISE NAME] to create **sections**
        i. If doing so, do not start each section with clear, clc all. Just once at the top.
        ii. You can run the code BY SECTION this way to check each exercise
02. Assignments are titled [LAST NAME]_[FIRST NAME]_LABHW_#
03. Office Hours are Wednesdays 12-1PM in ENG229B

I. Refreshers: Data Stored in Arrays

- Data structure is an organized way of storing data
- MATLAB's primary data structures is the **double array**
  - A double array has a 2D structure; 1 or more rows and 1 or more columns
  - At the intersection of a col and row is an **element**
- Arrays have different names depending on their dimensions
  - A ***scalar*** is a single-element array (1 row and 1 column).
  - A row ***vector*** is a single-row array (1 row and multiple columns).
  - A column ***vector*** is a single-column array (multiple rows and 1 column)
  - The term ***matrix*** generally refers to an array with 2 or more non-singular dimensions (not a vector)
  - The term ***array*** is a general term for all of the above

I. Refreshers: Arrays Creation

- Arrays can be constructed using:
  - Square brackets

MATLAB command	Description
`[1 2 3 4 5]`	% A row vector with 1 row and 5 columns
`[1; 2; 3]`	% A column vector with 3 rows and 1 column
`[1 2; 3 4]`	% A matrix with 2 rows and 2 columns

  - Built-in functions

MATLAB command	Description
`zeros(4)`	% Creates a 4x4 matrix (4 rows, 4 columns) with each element equal to 0
`ones(3,4)`	% Creates a 3x4 matrix (3 rows, 4 columns) with each element equal to 1
`ones(3,4)*2`	% Creates a 3x4 matrix (3 rows, 4 columns) with each element equal to 2

  - Or a combination of the two

MATLAB command	Description
`[1:3; 4:6]`	% Creates the 2x3 matrix `[1 2 3; 4 5 6]`
`[zeros(3) ones(3)]`	% Creates the 3x6 matrix `[0 0 0 1 1 1; 0 0 0 1 1 1; 0 0 0 1 1 1]`

I. Refreshers: Variables in Arrays

- What goes inside these arrays? Variables aka “value holders”
  - Recall,
    - Variables are **assigned** in the form *<variable name> = <expression>*
    - Variable names:
      - Must begin with a letter (a-z, A-Z).
      - May include only letters (a-z, A-Z), digits (0-9), and the underscore character (_)
      - Are case sensitive
  - If assignment **doesn’t** end with a semicolon, its echoed in the command window
    - Semicolon restricts the output.
    - Get in the habit of ending assignment lines with ;

MATLAB command	Description
`Count = 5;`	% The variable `Count` now has a scalar value of 5
`Matrix = [1 2; 3 4];`	% The variable `Matrix` now contains a 2-row by 2-column array
`Twos = ones(4)*2;`	% The variable `Twos` now contains a 4-row by 4-column array
`Age = input('Enter your age: ');`	% The "user" will be prompted to enter a value for `Age` and the value entered by the user will be placed into the variable.

I. Refresher: MATLAB Operations

01. Scalar Operations
    a. A **scalar value** is a single-element value (variable that is in a 1x1 array)
    b. Has scalar value as one or both operands
02. Try the following:

Table 1.1: Basic arithmetic operators

SYMBOL	OPERATION	EXAMPLE
+	Addition	$2+3$
−	Subtraction	$2-3$
*	Multiplication	$2*3$
/	Division	$2/3$

```
Scale = 2                                   % scalar
Dimensions = [10 20 30]                     % vector
Four = Scale * 2                            % scalar * scalar
Doubled = Dimensions * Scale                % vector * scalar
Tens = 10*ones(5)                           % scalar * matrix
```

I. Refreshers: Logical Data Types

- Recall MATLAB has *logical data types,* AKA Boolean variables
  - Logical data types host two possible values: true and false
    - True = 1
    - False= 0
  - Some examples of logical values:
    - Rocks= true; or 123_girls=0;
    - Can be scalar, matrix, vector
- Logical values are generated by
  - Relational operators--compares two values of same data type, usually #s
  - Logical operators--combines multiple true/false values into a single true/false value
  - Logical functions--returns logical true/false value

II. Relational & Logical Operators

- Relational Operators
  - Produce a single logical values and are:

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to (equivalent)
~=	not equal

```
1 < 2
1 > 2
1 == 2
Count = 0  % notice the assignment statement!
Count == 1
```

  - Know difference between the assignment operator (=) and the equality operator (==).
- Try ex/
  - Relational operators in arrays are applied element by element

```
[1 2 3] < [3 2 1]
'abcd' < 'cbaq'
[false false true] == [true false true]
[1 2; 3 4] > [5 6; 1 2]
[1 2 3 4] > [1 2]   % error -- both operands must have the same dimensions
```

II. Relational & Logical Operators

- Logical Operators
  - Has logical values as operands and returns single logical value

`&`	logical AND	`true` only if **both** values are `true`
`&&`	logical AND with shortcut evaluation	`true` only if **both** values are `true`
`|`	logical OR	`true` if **either** value is `true`
`||`	logical OR with shortcut evaluation	`true` if **either** value is `true`
`~`	logical NOT, unary	if `true`, then `false`; if `false`, then `true`
`xor`	logical EXCLUSIVE OR (actually a *logical function*)	`true` only if **both** values are different

- Ex/

```
x = input('Enter a value for x: ');
(x >= 0) && (x <= 10)
(x < 0) || (x > 10)
```

What do each mean?
What's returned?

II. Relational & Logical Operators

In Class Exercise: Logical Operations

Set the following variables to logic values:

A = true;

B = false;

C = true;

a. Write a script that evaluates and displays the logical value (0 or 1) for each of the following expressions. All outputs should be formatted using `fprintf` and the format descript `%d` should be used to display logical values. Comment why the answer is what it is.

   i. The logical AND of A and B

   ii. The logical AND of A and B combined with the logical OR of C

   iii. The logical not of A

II. Relational & Logical Operators

In Class Exercise: Logical Operations

```
%% Exercise Logical Operations


A= true;
B= false;
C= true;


qi = fprintf ('The logical AND of A and B is %d \n', (A & B));
qii = fprintf ('The logical AND of A and B combined with the logical OR of
C %d \n', ((A & B)|C));
qiii = fprintf ('The logical not of A is %d \n', (~A));
```

II. Relational & Logical Operators

- The shortcut operations (&& or ||) operations will stop the eval of an expression as soon as the results of expression are known
  - **Reads left → right**. If operand on the left is false, whole expression is too
    i. x = (b ~= 0) && (a/b > 18.5)
- When to use which?
  - Use shortcut operators (&&,||) when comparing single logical values (scalars)
  - Use non-shortcut operators (&,|) when comparing arrays of logical values

II. Relational & Logical Operators

Logical functions--a function that returns a logical value

Built-ins! Lets try. Type help for each in your command.

1. isnumeric
2. ischar
3. islogical
4. Isempty

Quick. Make an array A = [1 2 3] and ask if its a numeric array.

III. Control Structure

A ***control structure*** is a programming language mechanism for changing the order in which statements in that program are executed

Branches (selection statements): Structure that causes execution to jump *forwards* in program
Skipping over some code

Branch statements: **If**, **switch**, and **try/catch**

Loops (repetition statements): Structure that causes execution to jump *backwards* in program
Same code executed more than once

Loop statements: **for** and **while**

IV. Branch Statements: If-Statements

MATLAB'S selection of if-statements:

1. One-alternative **if**
2. Two-alternative **if**
3. Multiple-alternative **if** with else
4. Multiple-alternative **if** without else

You always pick the simplest form that will accomplish the task of your code

IV. Branch Statements: One-alternative if

One-alternative **if** has the syntax:

```
if  <logical expression>
    <one or more statements>
end
```

For example,

```
if Body_temp > 98.6
     fprintf('The patient has a fever.\n')
end
```

- If the logical expression evaluates to **true**, statement(s) until **end** are executed.
- Otherwise (if logical expression **false),** statements before **end** not executed

Tip// Indenting statements doesn't matter in MATLAB, but will make your life easier in terms of readability.
Use MATLAB's smart indent feature (Text -> Smart Indent or Ctrl-I) to automatically indent

IV. Branch Statements: Two-alternative if

Two-alternative **if** has the syntax:

```
if  <logical expression>
    <one or more statements>
else
    <one or more statements>
end
```

For example,

```
if Body_temp > 98.6
     fprintf('The patient has a fever.\n')
else
     fprintf('The patient does not have a fever.\n');
end
```

- If the logical expression evaluates to **true**, statement(s) before the **else** are execute and execution skips to **end**
- Otherwise (if logical expression **false),** statements before the else are skipped and statements after the **else** and before **end** are executed
- Statements before and after the **else** will never both be executed

IV. Branch Statements: Multiple-alt if with else

Multiple-alt **if** with else has the syntax:

```
if  <logical expression>
    <one or more statements>
elseif <logical expression>
    <one or more statements>
<0 or more additional elseif blocks>
else
     <one or more statements>
end
```

For example:

```
if Body_temp >= 102
      fprintf('The patient has a high fever.\n')
elseif Body_temp >= 99
      fprintf('The patient has a low grade fever.\n');
elseif Body_temp >= 92
      fprintf('The patient has a low body
temperature.\n');
      else
      fprintf('The patient is hypothermic.\n');
end
```

Use a multiple-alternative if with else statement when your code should do something for all possible cases.

IV. Branch Statements: Multiple-alt if without else

Multiple-alt **if** without else has the syntax:

```
if  <logical expression>
    <one or more statements>
elseif <logical expression>
    <one or more statements>
<0 or more additional elseif blocks>
end
```

For example:

```
if Body_temp >= 102
     fprintf('The patient has a high fever.\n')
elseif Body_temp > 98.6
     fprintf('The patient has a low grade fever.\n');
elseif Body_temp < 95
     fprintf('The patient has a low body temperature.\
end
```

Use a multiple-alternative if without else statement when the code should take no action in at least one case

IV. Branch Statements: Nested if

You can have nested if statements, such as:

```
if Body_temp >= 102
    fprintf('The patient has a high fever.\n')
    if Body_weight > 200 && Body_height < 64
      fprintf('Send the patient to the hospital\n');
    end
elseif Body_temp > 98.6
    fprintf('The patient has a low grade fever.\n');
elseif Body_temp < 95
    fprintf('The patient has a low body temperature.\n');
    if Temperature_time > 90
      fprintf('Send the patient to the hospital\n');
    end
end
```

IV. Branch Statements: Reminder

01. Design your **if** statements to minimize logical tests
02. Don't be redundant, you'll risk having code that is:
    a. Difficult to understand
    b. Harder to maintain
    c. Slower to execute

Not great:

```
if Score >= 90
   disp('A')
elseif Score >= 80 & Score < 90
   disp('B')
elseif Score >= 70 & Score < 80
   disp('C')
elseif Score >= 60 & Score < 70
   disp('D')
elseif Score < 60
   disp('F')
end
```

Great:

```
if Score >= 90
   disp('A')
elseif Score >= 80
   disp('B')
elseif Score >= 70
   disp('C')
elseif Score >= 60
   disp('D')
else
   disp('F')
end
```

IV. Branch Statements: If Statement

Exercise: If-Statement

a. Write a script that asks the user for two numeric values, one for x and one for y. Then displays a message saying whether x is greater than y, equal to y, or less than y. The output should be formatted using `fprintf` to display answers.

IV. Branch Statements: If Statement

```
%% Exercise If-statement 1

   x = input('Enter x: ');
   y = input('Enter y: ');

   if x > y
       fprintf('x is greater than y \n')
   elseif x < y
       fprintf('x is less than y \n')
   else
       fprintf('x is equal to y \n')
   end
```

Hint for Assignment Exercise 1

Solving a system of n equations with n unknowns:

$$5x+2y+7z=13$$

$$6x-4y+9z=21$$

$$8x-12y-7z=3$$

Let :

Matrix_Coeff= [5 2 7; 6 -4 9; 8 -12 -7];

%Unknowns = [x y z]; We don't create this actually

RHS = [13; 21; 3]

Thus, Matrix_Coeff * Unknowns= RHS

How can we solve for unknowns? Does the order of matrix multiplication matter?