Input and Output

Week 3

Loosely follows Chapter 2

Input

- Assign values using the equal sign (static/hard-coded input)
- Use the input() function (dynamic input)

Exercise

- Type the following into an m-file (area_of_circle.m)
  - radius = input("Please enter the radius: ")
  - area = pi * radius^2
- Execute and you should see the following

```
Please enter the radius: 5.2

radius =
        5.2000

area =
        84.9487
```

Might you want a semicolon somewhere?

Exercise

- Free falling object (free_falling_object.m)
- Type the following into a new m-file
  - height0 = input("Please enter the initial height (in meters): ");
  - time = input("Please enter the time the object was in the air (in seconds): ");
  - velocity0 = input("Please enter the initial velocity (in m/s): ");
  - ACC = 9.81;
  -
  - heightFinal = height0 + velocity0*time + (1/2)*ACC*time^2

Input Strings of Text

- Standard format
  - stringInput = input("Please enter your name: ", "s")
- Used when expecting a string as an input
- A string is a sequence of characters (s-t-r-i-n-g)
- Some examples for usage
  - Names
  - Months
  - A Letter

Capturing Output

- The diary function can be used to capture Command Window output.
- Creates a new file or appends to an existing file
- Example:
  - diary 'name_of_output_file.txt'
  - 5+10
  - diary off
- Results in output file named "name_of_output_file.txt"

Formatting Output

- format
  - Affects all output
  - Default: doubles show 4 sig figs, integers too large use scientific notation with 5 sig figs
  - format (long || short || shortEng || compact)
  - format alone or format default resets the default

```
format short e
% Any standard output here will display in scientific notation
format default % Always reset back to the default
```

Formatting Output

- disp()
  - Can display text and variables using a vector
  - Always ends with a carriage return
  - Variable formats follow the format spec

```
someVariable = 25;
disp("some message")
disp(someVariable)

someText = 'text';
disp(['some message with ', someText])
```

Formatting Output

- fprintf(formatSpec,A1,...,An)
  - Better suited to formatting output that might differ from the format command
  - Some common format specifiers %5.2f, %i, %s, %g
  - i can also take a number (%5i)
  - Don't forget '\n'
  - Can print to the console or a file
  - Requires special formatting to escape "special characters"

Formatting Output

- sprintf(formatSpec,A1,...,An)
  - Same as fprintf but stores resulting value in a variable
  - Better suited to formatting output that might differ from the format command
  - Some common format specifiers %4.2f, %d, %i, %s
  - Don’t forget ‘\n’

```
fileIncrement = 1;
fileName = sprintf("My file (%i)", fileIncrement)
```

```
fileName =
    "My file (1)"
```

Width and Precision

- The width specifies the minimum number of characters to be printed

- The precision field specifies the number of values to show after the decimal point.

Exercise

- %8.2f specifies that there can be no less than 8 characters displayed and that two of them are to follow the decimal point.
- Enter the following into a new m-file
  - weight = 57638.3333;
  - fprintf("The weight is %8.2f pounds\n", weight)
  - fprintf("The weight is %50.2f pounds\n", weight)
- Notice the blank spaces in the second output

Exercise

- Enter the following into a new m-file
  - username = input("Please enter your name: ", "s");
  -
  - fprintf("You said your name was %s, hello %s!\n", username, username)
- Note the %s
- Note the usage for displaying two variables

Exercise

- Enter the following into a new m-file
  - month = input("Please enter your month of birth (i.e. June): ", "s");
  - day = input("What day of the month were born? ");
  -
  - fprintf("Your birthday is %s %i!\n", month, day)

- Remember one of the inputs is a string
- The order of arguments is the same order as placeholders

Gotchas

- “ vs "
  - stringInput = input("Please enter your name: ", "s")
  - stringInput = input(“Please enter your name: “, “s”)
- ' vs "
  - String vs Char vector
- disp() Variable Types
  - someText = 'text';
  - disp(["some message with ", someText])
  - disp(['some message with ', someText])
  - disp(['some message with ', 25])

Control Structures

Week 3 (continued)

Control Structures

- Control
  - Conditionals (if, switch)
  - Loops (while, for)
- Condition
- Indenting is standard practice
  - MATLAB does this for you
  - ctrl+i will “smart indent” your script

	Control	Condition
Beginning	if	1 > 2
	- elseif → Check another - else → If nothing else - while → So long as it’s true - for → For each in list	
Statements	*Indent →*	…
		…
		…
End	end	

Comparative/Relational Operators

- x == y (true if x equals y)
- x ~= y (true if x does not equal y)
- x < y (true if x is less than y)
- x > y (true if x is greater than y)
- x <= y (true if x is less than or equal to y)
- x >= y (true if x is greater than or equal to y)

WARN: Do not use ‘=’ as it is for assignment and will *almost* always be true

Logical Operators

- | (bar) - OR
- & - AND
- ~ - NOT

|| (bar) - OR
&& - AND

Example conditions:

This and that → this & that
This or that → this | that
This and not that → this & ~that
Red or white and high or medium → (red | white) & (high | medium)

if - elseif - else

Syntax

```
if x < y
    // do this
end
```

```
if x < y
    // do this
elseif x > y
    // do this
end
```

```
if x < y
    // do this
elseif x > y
    // do this
else
    // do this
end
```

What must x and y be in order to execute the else statement?

if - elseif - else

Multiple Conditions

```
if x < y | force
    // do this
end
```

```
if x ~= y & z == 0
    // do this
end
```

while loop

- Executes ONLY if the condition is met
- Continues to execute while the condition is met
- Can use while true for an endless loop

```
x=1;
y=12;
while x < y
        x = x + 1;
end
```

for loop

Can be used for a set number of iterations
Can be used to parse an array or vector

```
for x = 1:10
        % Take and sum a sensor reading
end
reading = reading / 10;
```

We could also use the value of x if we wanted to

for loop

Parse a list/vector

```
myVector = [10, 12, 32, 5, 7];
for x = myVector
        fprintf("The current value is %i", x);
end
```

How about going backwards?

```
myVector = [10, 12, 32, 5, 7];
for x = fliplr(myVector)
        fprintf("The current value is %i", x);
end
```

switch case

The switch case statement is much like an if-elseif-else statement.
Useful when comparing values with ==

```
switch x
        case 6
                disp("X is 6")
        case 3
                disp("X is 3")
        otherwise
                disp("X is not 6 or 3")
end
```

switch case

The switch case can also perform an action for multiple cases, like an OR | with an if statement.

```
switch x
        case 0:9
                disp("x is a single digit")
        case 11:20
                disp("x is between 11 and 20")
        case [25,49,38,30]
                disp("x is one of my favorite numbers")
        otherwise
                disp("x is not in the range of 0-20 or one of my favorite numbers")
end
```

break and continue

- break - Forces the code to “break out” of a loop
  - The use of break is similar to using “And” to start a sentence. It’s generally frowned upon but there are valid use-cases for it.
  - The book’s author even refuses to show an example.

- continue - Skips the remaining code in a single loop iteration.

break and continue

break example

```
sum = 0;
failed = false;
% Try to take 10 readings from a sensor
for 1:10
        reading = analogRead(SENSOR);

        % If it's a bad reading, mark a failure and leave
        If reading < 0
                failed = true;
                break;
        end

        sum = sum + reading;
        numReadings = numReadings + 1;
end
```

break and continue

continue example

```
sum = 0;
numReadings = 0;
% Try to take 10 readings from a sensor
for 1:10
        reading = analogRead(SENSOR);

        % If it's a bad reading, skip and try again
        If reading < 0
                continue;
        end

        sum = sum + reading;
        numReadings = numReadings + 1;
end
average = sum / numReadings;
```

Single Line Control Structures

- This is only so you know how to read it if you see it

```
if x>4; disp("x is greater than 4"); end

for index = 1:5; disp(index); end
```

Nested Control Structures

You may have noticed some examples have nested control structures.

```
if x > 4
        disp("x is greater than 4")
        if y > 4
                disp("y is also greater than 4")
        end
else
        disp("x is less than or equal to 4")
end
```

Gotchas

ctrl+c will stop MATLAB code execution