Introduction to Programming for Biology Research
Introduction to MATLAB: part II

MATLAB Basics II
- Quick review
- While-loops
- Functions
  - internal
  - custom
- Plotting commands
Introduction to MATLAB: part II

MATLAB Basics II
- Quick review
- While-loops
- Functions
  - internal
  - custom
- Plotting commands
Introduction to MATLAB: part II

MATLAB Basics II
- Quick review
- While-loops
- Functions
  - internal
  - custom
- Plotting commands
Problem 1
Create a new script and save it as “yourName_P1”. Execute the commands listed below, and use comments to explain what each command is doing.

1) M1 = [ 3 5 1; 2 8 4];
2) m1 = [ 80 75 6; 100 34 8 ];
3) m1 == M1
4) matrix3 = [ 89 4 9; 1 -80 503 ];
5) matrix3(1,1) = m1;
6) store(2,1) = matrix3
7) store(2,2) = M1
8) store(1,1) = ‘Data of matrix3’;
9) store(1,2) = ‘Data of M1’
10) for i = 1 : 0.75 : 75
    savel(cnt) = 1;
end

Question 2
Suppose you are a young and talented scientist who is interested in behavioral neurobiology of humans. It’s your first day in a new lab and your PI throws some EMG data at you. They had recorded activity data for 300 seconds at 10Hz (100samples/sec). Your PI suspects that there is a single spike in neural activity somewhere in the data set, and needs you to tell them when this spike occurs. They tell you that the spike resulted in a peak at least 10x that of baseline, so simply finding the position of the maximum value in this array will correspond to the time of the spike. Your PI doesn’t know how to code, so they have been looking at each time-point in this dataset manually.

They wrote a basic algorithm for finding the spike, but need your expertise in coding to implement it in MATLAB.

PI Algorithm:
1) Set LargestValue equal to 0
2) Move to element 1 in the dataset
3) If element i is greater than LargestValue
   a) If so, store position of element 1 as LargestValue
5) Else, do not change LargestValue
6) Move to next element
8) If so, then store position of element 2 as LargestValue
9) Else, do not change LargestValue
10) Repeat steps 3-8 for every element in the array

Your PI seems to think their algorithm is correct, however, it would be much too time consuming to do this manually. They come to you and beg for you to execute this algorithm (or something like it) in MATLAB.

Please refrain from using MATLAB’s internal functions for this exercise
1) What are the dimensions of p2_vector?
   a) This variable should have been imported into MATLAB’s workspace after you opened the p2_data.mat file.
2) Write a for-loop to index through every position in the p2_vector.
3) Inside your for-loop write a conditional statement to find the maximum value of the p2_vector.
4) Write a conditional statement to find the time-point (index) for when the LargestValue occurs in p2_vector, and save this position as, spikeTime. This will be the time at which the spike occurs.
5) Execute the code and take a screenshot of the output:
   figure; plot(p1_vector(spikeTime-25:spikeTime+25))

Question 3 (Challenge)
This question is a continuation of Question 3 from problem set 1. The details are listed below:

Suppose you are interested in mouse behavior, and you want to know what odors cause mice. When mice are afraid, they stop moving and freeze. To test which odors are fear inducing, you record a movie of the mouse walking around its cage, and randomly release one of ten odors. You record the mouse for 2 hours, and expose the mouse to a randomly selected odor once every 2 minutes (for 30 sessions).

Somebody else’s program automatically records the results in two variables. The first variable contains the coordinates of the mouse, recorded every second. The second variable records which odor is being released (stored as the numbers 1-10, with a number 0 if no odor is being released), also recorded every second.

Your goal is to create a bar graph that plots the average speed of the mouse when each odor is exposed.

Create a new script titled “yourName_P3”. Write a script to convert the two variables you’ve received into a bar graph.

a. What type of variable is odorExp?

b. What data is stored in odorExp?

c. Identify all time-points that have odor 1 presented. (Hint: one method is to write a for-loop to search through each position of the odor array. In the for-loop you’ll provide a conditional statement if the odor presented == 1, then store that time point in an array. Your output should be an array containing all the time points in which odor 1 was presented.)

d. Find the speeds of the mouse when odor 1 is presented. To do this: use the array created in part e to index the speed array (in odorExp) and return only speeds when odor 1 is presented.

e. Use the mean function to find the average value of the array created in part d. This single value will be the mean speed of the mouse when odor 1 is presented.

f. Perform steps c-e for all odor values (0-10) and produce an array that contains the mean speed for each odor, and title it AvgSpeed. This array should have 11 elements, one value for each odor.

g. Execute the command:
   figure; bar(AvgSpeed)

Please don’t hesitate to send any questions to matthew.smith01@harvard.edu.

Attend the review session on Monday @7pm for assistance in person.

Good luck!
Review
MATLAB: the interface

4 Default windows:

1) Current folder:
2) Workspace:
3) Editor:
4) Command window:
MATLAB: the interface

4 Default windows:

1) Current folder:
2) Workspace:
3) Editor:
4) Command window:

The current folder (directory) shows where MATLAB will read and write files to
MATLAB: the interface

4 Default windows:

1) Current folder:
2) Workspace:
3) Editor:
4) Command window:

The workspace consists of the variables you create and store in memory during a MATLAB session.
MATLAB: the interface

4 Default windows:

1) Current folder:
2) Workspace:
3) Editor:
4) Command window:

The Editor window is where one can write functions or scripts, which can be saved and executed by calling by name.
4 Default windows:

1) Current folder:  
2) Workspace:  
3) Editor:  
4) Command window:

The command window is where one can type single line commands to MATLAB.
Scripts

Scripts are used to write multiple lines of code. These commands can be saved as a .mat file and loaded for later use.
Scripts

Scripts are used to write multiple lines of code

These commands can be saved as a .mat file and loaded for later use
Scripts

Under the editor tab
Click on the “new editor tab” symbol if the editor window isn’t displayed.
Scripts

Press the “Run” button

Once the script is saved, the user can recall the saved commands by calling the script. Type the saved name in the command window and press enter.
Logical (boolean):

A variable that has two values:

true or false

MATLAB interprets this as either:

1 or 0
Logical (boolean):

A variable that has two values:

**true** or **false**

MATLAB interprets this as either:

1 or 0

- `==` is equal to
- `~=` is not equal to
- `<` is less than
- `>` is greater than
- `>=` is greater or equal to
- `<=` is less or equal to
- `&` and
- `|` or
Conditional statements:
Conditional statements are if-then statements. The simplest conditional statement is an *if* statement. *If* this conditional is true, *then* execute the below code.
Conditional statements:
Conditional statements are if-then statements. The simplest conditional statement is an if statement. If this conditional is true, then execute the below code.

```
Will I go to class today?
if freeFood == true
    MattAttend = true
elseif interestingSpeaker == true
    MattAttend = true
else
    MattAttend = false
end
```
Conditional statements:
Conditional statements are if-then statements. The simplest conditional statement is an if statement. If this conditional is true, then execute the below code.

Classifying neurons:
if sum(Recordedneuron) > 5*baseline
    spike = true;
    performSpikeAnalysis
elseif sum(Recordedneuron) < -5*baseline
    inhibition = true;
    performInhibitionAnalysis
end
Loop Control

Used to repeatedly execute a block of code

Two loop control operators:

While

For
For-loops

cnt = 1;
for i = [2,4,8]
    class(cnt) = i;
    cnt = cnt+1;
end
For-loops

cnt = 1;  <- define variable: cnt
for i = [2,4,8]  <- start for-loop, set variable i = 2,4,8
  class(cnt) = i;  <- In position (cnt) of variable (class),
  cnt = cnt+1;    assign the value (i)
end  <- add 1 to cnt
Loop Control

For-loops

cnt = 1;
for i = [2,4,8]
    class(cnt) = i;
cnt = cnt+1;
end

n = 1;
for i = [2,4,8]
    class(n) = i;
n = n+1;
end
Loop Control

For-loops

cnt = 1;
for i = [2,4,8]
    class(cnt) = i;
cnt = cnt+1;
end

n = 1;
for i = [2,4,8]
    class(n) = i;
n = n+1;
end

1) define variable: n
2) start for-loop, set variable i = 2,4,8
3) In position (cnt) of variable (class), assign the value (i)
4) add 1 to cnt
Loop Control

For-loops

cnt = 1;
for i = [2,4,8]
    class(cnt) = i;
    cnt = cnt+1;
end

n = 1;
for i = [2,4,8]
    class(n) = i;
    n = n+1;
end

n = 1;
for i = [2,4,8]
    class(i) = n;
    n = n+1;
end
Loop Control

For-loops

cnt = 1;
for i = [2,4,8]
class(cnt) = i;
cnt = cnt+1;
end

n = 1;
for i = [2,4,8]
class(n) = i;
n = n+1;
end

n = 1;
for i = [2,4,8]
class(i) = n;
n = n+1;
end

1) define variable: n 2) start for-loop, set variable i = 2,4,8 3) In position (i) of variable (class), assign the value (n) 4) add 1 to n
New MATerial
while Loops

While some conditional statement holds true, execute the code inside the loop.
while Loops

While some conditional statement holds true, execute the code inside the loop.

cnt = 1;
while cnt < 100
    disp('Count is less than 100')
    cnt = cnt+1;
end
**while Loops**

**While** some conditional statement holds true, execute the code inside the loop.

```plaintext
cnt = 1;
while cnt < 100
   disp('Count is less than 100')
   cnt = cnt+1;
end
```

While loops are commonly used if there is a time threshold to perform operations, or for optimization.
**while Loops**

**While** some conditional statement holds true, execute the code inside the loop.

```matlab
cnt = 1;
while cnt < 100
    disp('Count is less than 100')
    cnt = cnt+1;
end
```

While loops are commonly used if there is a time threshold to perform operations, or for optimization.

Imagine you’re simulating a bacterial infection that you must treat with antibiotics. **While** the cell count is above 11,000 cells per liter, you must treat with antibiotics. If the cell count drops below 11,000, you break the **while** loop and stop treating bacterial infection.
Logical indexing

When making a logical comparison between a vector or matrix and a scalar MATLAB will evaluate the truth of each element.
Logical indexing

When making a logical comparison between a vector or matrix and a scalar MATLAB will evaluate the truth of each element

```
xPos = [ 15 20 34 40 45 60 80 90 120 150 134 20 34 ]
```
Logical indexing

When making a logical comparison between a vector or matrix and a scalar MATLAB will evaluate the truth of each element

\[
xPos = [15 \ 20 \ 34 \ 40 \ 45 \ 60 \ 80 \ 90 \ 120 \ 150 \ 134 \ 20 \ 34]
\]

\[
sideA = (xPos \geq 90)
\]
Logical indexing

When making a logical comparison between a vector or matrix and a scalar MATLAB will evaluate the truth of each element

```matlab
xPos = [ 15 20 34 40 45 60 80 90 120 150 134 20 34 ]
sideA = (xPos >= 90)
```

MATLAB indexes through every position in the array and asks if each element is greater than 90, and returns a logical evaluated at every position.
Logical indexing

When making a logical comparison between a vector or matrix and a scalar MATLAB will evaluate the truth of each element

\[ \text{xPos} = [15 \ 20 \ 34 \ 40 \ 45 \ 60 \ 80 \ 90 \ 120 \ 150 \ 134 \ 20 \ 34] \]

\[ \text{sideA} = (\text{xPos} \geq 90) \]
Logical indexing

When making a logical comparison between a vector or matrix and a scalar MATLAB will evaluate the truth of each element

\[
\text{xPos} = [15 20 34 40 45 60 80 90 120 150 134 20 34]
\]

\[
\text{sideA} = (\text{xPos} >= 90)
\]

\[
\text{sideA} = [0 0 0 0 0 0 0 1 1 1 1 0 0]
\]

MATLAB returns a logical vector, the positions that met the conditional statement are true (1)
When making a logical comparison between a vector or matrix and a scalar MATLAB will evaluate the truth of each element.

\[
xPos = [ 15 \ 20 \ 34 \ 40 \ 45 \ 60 \ 80 \ 90 \ 120 \ 150 \ 134 \ 20 \ 34 ]
\]

\[
sideA = (xPos >= 90)
\]

\[
sideA = [ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 ]
\]

\[
xPos(sideA)
\]
Logical indexing

When making a logical comparison between a vector or matrix and a scalar MATLAB will evaluate the truth of each element

\[
\text{xPos} = [ 15 \ 20 \ 34 \ 40 \ 45 \ 60 \ 80 \ 90 \ 120 \ 150 \ 134 \ 20 \ 34 ]
\]

\[
\text{sideA} = (\text{xPos} \geq 90)
\]

\[
\text{sideA} = [ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 ]
\]

\[
\text{xPos(sideA)}
\]

\[
= 90, \ 120, \ 150, \ 134
\]

By passing the logical back into MATLAB, the user tells MATLAB to only return positions that correspond to a true (1) value in the logical
Writing functions
Commenting

Whether you’re writing a script or function it’s important to add comments
%Commenting

Whether you’re writing a script or function it’s important to add comments

Comments are meant to be descriptions or notes to the programmer reading/using your code
Whether you're writing a script or function, it's important to add comments. Comments are meant to be descriptions or notes to the programmer reading or using your code. Dr. Strange's famous quote: "I don't remember being here."
%Commenting

Add a comment to your script by placing a % at the beginning of a line

```
% Below is my example of a script
myVariable = 'Yo. This is my text variable'
```
Using functions
Using functions

What is a function?
Using functions

What is a function?

A series of operations (complex or simple) that can be recalled and iterated
Using functions

What is a function?

A series of operations (complex or simple) that can be recalled and iterated.

Functions can perform tasks more efficiently because they use a local workspace instead of the global workspace.
Using functions

What is a function?

A series of operations (complex or simple) that can be recalled and iterated.

Functions can perform tasks more efficiently because they use a local workspace instead of the global workspace.
Using functions

What is a function?

A series of operations (complex or simple) that can be recalled and iterated.

Functions perform operations in their own local workspace and interact with the global workspace via inputs and outputs.
Using functions

What is a function?

A series of operations (complex or simple) that can be recalled and iterated

Functions can perform tasks more efficiently because they use a local workspace instead of the global workspace

Functions can have specific inputs/outputs
Using functions

Function to find the sum of all elements in a vector or matrix
Using functions

Function to find the sum of all elements in a vector or matrix
Using functions

sum

function name

Function to find the sum of all elements in a vector or matrix
Using functions

```
sum( input )
```

- **function name**: `sum`
- **function will perform operations on the variable(s) inside the parenthesis**: `input`

Function to find the sum of all elements in a vector or matrix
Using functions

output = sum( input )

function name

variable name(s) of output data

function will perform operations on the variable(s) inside the parenthesis

Function to find the sum of all elements in a vector or matrix
Using functions

output = \text{sum}( \text{input} )

- variable name(s) of output data
- function name
- function will perform operations on the variable(s) inside the parenthesis

$$x = [4, 5, 10, 40, 7]$$
Using functions

output = sum( input )

variable name(s) of output data

function name

function will perform operations on the variable(s) inside the parenthesis

x = [4, 5, 10, 40, 7]

sumX = sum( x )
Using functions

output = sum( input )

variable name(s) of output data

function name

function will perform operations on the variable(s) inside the parenthesis

\[ x = [4, 5, 10, 40, 7] \]

\[ \text{sumX} = \text{sum}( x ) \]

\[ \text{sumX} = 66 \]
Writing functions

\[ \text{output} = \text{function}(\text{input}) \]

Function: local workspace

Command window

Global workspace
Writing functions

\[ \text{output} = \text{function}(\text{input}) \]

Function: local workspace

Command window
Writing functions

\[
\text{output} = \text{function}(\text{input})
\]

Function: local workspace

Command window

Global workspace
Writing functions

\[
\text{output} = \text{function}(\text{input})
\]

Function: local workspace

- \[\text{function operations}\]
- \[\text{output} = \text{input} \times 2\]
Writing functions

output = function(input)

Function: local workspace

function operations
output = input x 2

Command window

value = 5

Global workspace
Writing functions

\[ \text{output} = \text{function}(\text{input}) \]

Function: local workspace

---

**function operations**

\[ \text{output} = \text{input} \times 2 \]

---

Command window

- value = 5

Global workspace

- value = 5
Writing functions

`output = function(input)`

Function: local workspace

```
function operations

output = input x 2
```

Command window

```
out1 = function(value)
```

Global workspace

```
value = 5
```
Writing functions

_output = function(input)_

Function: local workspace

```
function operations
output = input \times 2
```

Command window

```
value = 5
out1 = function(value)
```

Global workspace

```
value = 5
```
Writing functions

output = function(input)

Function: local workspace

input = 5

function operations

output = input x 2

Command window

value = 5

out1 = function(value)

Global workspace

value = 5
Writing functions

\[ \text{output} = \text{function}(\text{input}) \]

Function: local workspace

\[
\begin{align*}
\text{input} &= 5 \\
\text{function operations} \\
\text{output} &= \text{input} \times 2 \\
&= 5 \times 2 \\
&= 10
\end{align*}
\]

Command window

\[
\begin{align*}
\text{value} &= 5 \\
\text{out1} &= \text{function}(\text{value})
\end{align*}
\]

Global workspace

\[
\begin{align*}
\text{value} &= 5
\end{align*}
\]
Writing functions

\[ \text{output} = \text{function}(\text{input}) \]

Function: local workspace

\[ \text{input} = 5 \]

\[ \text{function operations} \]

\[ \text{output} = \text{input} \times 2 \]

\[ = 5 \times 2 \]

\[ = 10 \]

Command window

\[ \text{value} = 5 \]

\[ \text{out1} = \text{function}(\text{value}) \]

Global workspace

\[ \text{value} = 5 \]

\[ \text{out1} = 10 \]
Writing functions

output = function(input)

Function: local workspace

input = 5

function operations

output = input * 2

= 5 * 2

= 10

Global workspace

value = 5

out1 = function(value)

out1 = 10

Command window

value = 5

out1 = 10
Writing functions

output = function(input)

Function: local workspace

function operations

output = input x 2

Command window

Global workspace

value = 5

out1 = 10
MATLAB predefined functions

**sum( )**

- Specify what variable to perform this operation on
- Tell MATLAB what function to call by typing its name
MATLAB predefined functions

\[ x = [1, 5, 10, 15, 30] \]

Specify what variable to perform this operation on

\[ \text{sum}() \]

Tell MATLAB what function to call by typing its name
MATLAB predefined functions

x = [1, 5, 10, 15, 30]  Specify what variable to perform this operation on

sum(x)  Tell MATLAB what function to call by typing its name
MATLAB predefined functions

\[ x = [1, 5, 10, 15, 30] \]

`sum(x) = 61`

Specify what variable to perform this operation on

Tell MATLAB what function to call by typing its name
MATLAB commands/
functions

figure - creates window for plotting
close - closes figure windows
exit - closes MATLAB
length - gives the length of largest dimension
size - outputs length of column and rows
tic - starts timer
toc - outputs elapsed time since “tic” command
zeros - creates array of zeros
ones - creates array of ones
sum - adds all elements in a vector or matrix
mean - finds average of all elements in a vector or matrix
std - finds standard deviation of all elements in a vector or matrix
MATLAB commands/functions

figure - creates window for plotting
close - closes figure windows
exit - closes MATLAB
length - gives the length of largest dimension
size - outputs length of column and rows
\texttt{tic} - starts timer
\texttt{toc} - outputs elapsed time since “tic” command
zeros - creates array of zeros
ones - creates array of ones
sum - adds all elements in a vector or matrix
mean - finds average of all elements in a vector or matrix
std - finds standard deviation of all elements in a vector or matrix
help command
help command
help command
Mathworks is full of incredibly helpful resources, including video tutorials, steps for function execution, Q&A, and custom functions.
Writing a function:

Benefits of a function:

- Can easily quickly recall operations (complex or simple) by coding them into a function.
- Functions can perform tasks more efficiently. This is partly because functions use a local workspace instead of the global workspace (used by the command window and scripts).
Writing a function:

Benefits of a function:

- Can easily and quickly recall operations (complex or simple) by coding them into a function.
- Functions can perform tasks more efficiently. This is partly because functions use a local workspace instead of the global workspace (used by the command window and scripts).

Example:

```matlab
>> matrix = [6,5,4;3,2,1]
```
Writing a function:

Benefits of a function:

• Can easily quickly recall operations (complex or simple) by coding them into a function.

• Functions can perform tasks more efficiently. This is partly because functions use a local workspace instead of the global workspace (used by the command window and scripts).

```
>> matrix = [6,5,4;3,2,1]

matrix =

    6     5     4
    3     2     1
```

```
[m1,m2,m3,m4] = size(X) returns m1 = 2, m2 = 3, m3 = 4, m4 = 1
[m,n] = size(X) returns m = 2, n = 12
m3 = size(X,3) returns m3 = 3
```
Writing a function:

- Benefits of a function:
  - Can easily quickly recall operations (complex or simple) by coding them into a function.
  - Functions can perform tasks more efficiently. This is partly because functions use a local workspace instead of the global workspace (used by the command window and scripts).
Wri$ng'a'func$on:'

Benefits'of'a'func$on:'

• Can easy/quickly recall operations (complex or simple) by coding them into a function."

• Functions can perform tasks more efficiently. – This is partly because functions use a local workspace instead of the global workspace (used by the command window and scripts)."
Writing MATLAB functions

A user can write custom functions specific for their needs
Writing a function:

You must specify what variables to input into the function. The function cannot automatically draw from the global workspace. Functions use a local workspace, they cannot directly call from the global workspace. Therefore, you must specify what variables are input and what variables are output from the function.

You call your function by the name it is saved as. 

ex: test('x')
function [output_args] = untitled2(input_args)

% UNTITLED2 Summary of this function goes here

% Detailed explanation goes here

end
The user specifies that this is a function by typing “function” in the first line.
The user specifies that this is a function by typing “function” in the first line.

The name by which MATLAB will save your function
The user specifies that this is a function by typing “function” in the first line.

The name by which MATLAB will save your function inputs.

Define your function inputs.

The name by which MATLAB will save your function.
The user specifies that this is a function by typing “function” in the first line. Define your function inputs. Define your function outputs. The name by which MATLAB will save your function.
Practice writing functions:

Write a function:

- It will have one input (a numerical value)

- Inside your function, multiply the input variable by 5

- Return this new value as your output
Practice writing functions:

Write a function to find the largest value in a vector or matrix
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

How would you do this without a computer?

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]
Practice writing functions:

I would start with the first element in the vector. This would be temporarily defined as the initial largest element. I would then proceed through all elements and compare them to the temporary largest element. If it the current element is larger than the temporary largest, it will replace it. After proceeding through the entire vector only the absolute largest element will be stored.

\[x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144]\]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1,45,3,200,18,305,23,78,144] \]

largest value = [ 0 ]
Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]

is \[ 1 > \text{largest value} \]

largest value = \[ 0 \]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1,45,3,200,18,305,23,78,144] \]

is \( 1 > \) largest value = true

largest value = [0]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]

largest value = [ 1 ]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]

largest value = [ 1 ]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]

is \( 45 > \) largest value

largest value = [ 1 ]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[
x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144]
\]

is \(45 > \text{largest value} = \text{true}\)

largest value = [1]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]

largest value = [45]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]

is \( 3 > \text{largest value} \)

largest value = [45]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]

is \( 3 \) > largest value = false

largest value = [ 45 ]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]

continue this until you go through the entire vector

largest value = [305]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

\[ x_1 = [1, 45, 3, 200, 18, 305, 23, 78, 144] \]
Practice writing functions:

Write a function to find the largest value in a vector or matrix

largest = 0
for i = 1:length(x1)
    if x1( i ) > largest
        largest = x1( i )
    end
end
Practice writing functions:

```
function [ output ] = largestValue( input )
    % This function will return the largest value of the input
    end
```
Practice writing functions:

```matlab
function [output] = largestValue(input)
% This function will return the largest value of the input
output = 0;
for i = 1:length(input)
    if input(i) > output
        output = input(i)
    end
end
end
```
Practice writing functions:

```matlab
function [ output ] = largestValue( input )
% This function will return the largest value of the input
output = 0;
for i = 1:length(input)
    if input(i) > output
        output = input(i)
    end
end
end
```

```matlab
>> x1 = [1,45,3,200,18,305,23,78,144];
>> large = largestValue(x1)
```
Practice writing functions:

```matlab
function [ output ] = largestValue( input )
    % This function will return the largest value of the input
    output = 0;
    for i = 1:length(input)
        if input(i) > output
            output = input(i);
        end
    end
end
```

large = 305
2-D plot command

plot( y )
2-D plot command

function name

plot( y )

input: data to be plotted
2-D plot command

function name

plot( y )
2-D plot command

\[ \text{plot}(y) \]

function name

\text{plot}(y)

\text{>> y} = [0,1,2,4,8];
\text{>> plot(y)}
2-D plot command

function name

plot(x, y)
2-D plot command

function name

>> x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
>> y = [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100];
>> plot(x, y)
Random number generator

rand()
Random number generator

function name

rand( )
Random number generator

function name

rand( 4 )

input: size of output variable
MATLAB assumes
4 rows and 4 columns
Random number generator

function name

rand( 2,3 )

input: size of random matrix (rows, columns)
Make a dice simulator
Write a function:

- Two inputs (how many sides of the die, how many rolls)
- Inside your function, use rand to generate random rolls of the x-sided die
- Return an array filled with the generated rolls
Fin.

Congrats!
- No review session Monday
- Next lecture (Wed) is Data analysis
- The final problem set will be posted tomorrow