Introduction to Programming for Biology Research
variation across species

the white clover weevil and three Drosophila species vary across strain and species in how much phototactic personality they have.

FlyVac
coordinates 32 modules to autonomously measure the light preference of individual flies, many times each.

LegTracker
an instrument for recording the position of all 6 of a fly’s legs, in real time, at 80Hz.

automated behavioral classification

classifications of spontaneous behavior, assigned by human investigators and machine learning algorithms.

effect of weather on behavior

predicted fly population dynamics and phototactic behavior dynamics as influenced by real world weather conditions from 2008.
Introduction to MATLAB: part I

MATLAB Basics
- The interface
- Variables/arrays/matrices
- Conditional statements
- Loops (for and while)
MATLAB: The interface

4 Default windows:

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

The command window is where one can type commands to MATLAB
The command window is where one can type commands to MATLAB.
Algorithms

- Algorithms are a sequence of step-by-step instructions for accomplishing a task
- These instructions must be:
  - Effectively computable (doable)
  - Unambiguous
- When it is finished, the algorithm must end, and must produce some kind of result
The command window is highlighted below.

This is where we input operations/commands to instruct MATLAB to perform algorithms.
Above is a command typed into the command window
Over the next few slides we’ll interpret this command
Variables. The Most Important Slides.
Variables. The Most Important Slides.

In MATLAB, data is stored in Variables

Variables are essentially named storage locations that correspond to a location in your computer's RAM where MATLAB can find your data.
Variables. The Most Important Slides.

In MATLAB, data is stored in **Variables**

**Variables** are essentially named storage locations that correspond to a location in your computer's RAM where MATLAB can find your data

```
myVar1 = 10
```
Variables. The Most Important Slides.

In MATLAB, data is stored in **Variables**

**Variables** are essentially named storage locations that correspond to a location in your computer's RAM where MATLAB can find your data.

Data is assigned to variables through the ‘equals sign’

myVar1 = 10

Data is **READ** from the **RIGHT** and **assigned** to the **left**
myFirstCommand = 'i <3 iPBR'
We’ve just entered our first command into MATLAB’s command window!

The left of the equals sign represents the variable name: `myFirstCommand`

The right side of the equals sign represents the value: ‘`i <3 iPBR`’
fx >> 'matt <3' = statement
MATLAB throws an error when we try to execute this line of code.

This is because we switched the value: ‘matt <3’ to the left of the equals sign, and the variable name to the right.

In MATLAB the variable name you’re creating or modifying will always be to the left of the equals sign and the value you’re assigning on the right.
fx >>
By typing a variable name into the command window you can recall its saved value.
myFirstCommand

myFirstCommand =

i < 3 iPBR

fx >>
MATLAB’s workspace is highlighted in the bottom left side of the screen. The variable names that the user has created are stored in this area.
<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>myFirstComm...</td>
<td>'i &lt; 3 iPBR'</td>
</tr>
</tbody>
</table>
Workspace

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

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

It’s like MATLAB’s short term memory!
Workspace

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

You add variables to the workspace by using functions, running MATLAB code, and loading saved workspaces.
Current folder (directory):
The current folder shows where MATLAB will read and write files to.

MATLAB default pathway:
/Users/matthewsmith/Documents/MATLAB
The current folder shows where MATLAB will read and write files to.

MATLAB default pathway:
/Users/matthewsmith/Documents/MATLAB

This command has created a variable named x
It is assigning a value of 7200 to the variable x
If you look in the command window, you can see that the variable `x` is now in MATLAB’s workspace.
The current folder shows where MATLAB will read and write files to.

MATLAB default pathway: `/Users/matthewsmith/Documents/MATLAB`

```
>> x = 7200

x = 
    7200

>> y = 35
```
x = 7200

>> y = 35

y = 35
This command represents basic arithmetic that can be performed on numerical values in MATLAB.

Notice that we don’t have an equals sign or a variable name assigned to this command. In the next slide we will discuss what MATLAB does in this scenario.
MATLAB has performed the command and has assigned the output to the default variable name, `ans`.

If you look in the workspace, you can see this variable `ans` and the value of 7235.
\texttt{fx} >> \texttt{x*y}
When executing a new command with no variable name MATLAB will overwrite the default variable, ans, with a new value for the last command executed.

Look in the workspace to see that ans now has a value of 252000. If you want to save an output, assign it to a unique variable name that will not be overwritten.
Variable Types

Workspace

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>randomValue</td>
<td>580343</td>
</tr>
<tr>
<td>thisIsTrue</td>
<td>'we ALL love iPBR'</td>
</tr>
</tbody>
</table>
Variable Types

**Numeric data:** integer, double, float
Variable Types

**Numeric data:** integer, double, float

Basic mathematical operators:
addition, subtraction, multiplication, etc.
Variable Types

**Numeric data:** integer, double, float

**Text data:** Sequence of characters, normally called a string. Strings are treated as arrays that contain characters.

Basic mathematical operators: addition, subtraction, multiplication, etc.

Create a string in MATLAB by using single quotations:

‘This is a string’
Variable Types

**Numeric data:** integer, double, float

**Text data:** Sequence of characters, normally called a string. Strings are treated as arrays that contain characters.

Create a string in MATLAB by using single quotations:

```
>> message = 'Hello world'
```
Variable Types

**Numeric data:** integer, double, float

**Text data:** Sequence of characters, normally called a string. Strings are treated as arrays that contain characters.

Create a string in MATLAB by using single quotations:

```
string = 'Hello world'
```
In-class exercise 1:
In-class exercise 1:

Discuss with your neighbor(s) an algorithm for converting degrees Fahrenheit to degrees Celsius.
In-class exercise 1:

Write an algorithm to convert Fahrenheit to Celsius

Execute this algorithm in MATLAB to convert 45.6°F into degrees Celsius.
In-class exercise 1:

Write an algorithm to convert Fahrenheit to Celsius

Execute this algorithm in MATLAB’s command window to convert 45.6F into degrees Celsius
Temperature conversion algorithm:

1) Select an input temperature in Fahrenheit to convert to Celsius
2) Perform the operation:
   \[ \text{celsius} = \text{input temperature} - 32 \times \frac{5}{9} \]
Temperature conversion algorithm:

1) Select an input temperature in Fahrenheit to convert to Celsius
2) Perform the operation:
   \[ \text{celsius} = \text{input temperature} - 32 \times \frac{5}{9} \]

Temperature conversion algorithm: in MATLAB

1) \text{inputTempFah} = 45.6
2) \text{outputTempCels} = \text{inputTempFah} - 32 \times \frac{5}{9}
>> inputTempFah = 45.6
outputTempCels = (inputTempFah-32)*(5/9)
Users may notice that the command window is practical for writing single line commands. It is more difficult to write long series of commands or to recall past commands.

One solution is to use MATLAB’s script editor (Above).
Writing scripts

Scripts are multiple lines of MATLAB commands and function that can be saved. You can execute a script by typing its saved name.
The temperature conversion algorithm has now been moved from the command window to the script editor window.

```
1 - inputTempFah = 12
2 - outputTempCels = (inputTempFah-32)*(5/9)
```
The following slides will show how to save the series of commands written in the script.

```
inputTempFah = 12
outputTempCels = (inputTempFah-32) * (5/9)
```
-32)*(5/9);
1. \( \text{inputTempFah} = 12 \) 
2. \( \text{outputTempCels} = \frac{(\text{inputTempFah}-32) \times 5}{9} \)
Our saved script now appears in MATLAB’s current folder, which is where MATLAB will read and write files.

```matlab
1 - inputTempFah = 12;
2 - outputTempCels = (inputTempFah - 32) * (5/9);
```
Current folder (directory):

The current folder shows where MATLAB will read and write files to.

MATLAB default pathway:
/Users/host/Documents/MATLAB
Current folder (directory):

The current folder shows where MATLAB will read and write files to.

MATLAB default pathway:
/Users/Host/Documents/MATLAB

You can tell MATLAB to include folders that aren’t in your current directory by clicking “set path”.
1. `inputTempFah = 12`
2. `outputTempCels = (inputTempFah-32)*(5/9)`
1. `inputTempFah = 12`
2. `outputTempCels = (inputTempFah-32) * (5/9)`

```matlab
inputTempFah =

12

outputTempCels =

-11.1111

fx >>
```
1 - `inputTempFah = 12`

2 - `outputTempCels = (inputTempFah - 32) * (5/9)`

```
inputTempFah =

12

outputTempCels =

-11.1111

fx >>
```
MATLAB will only read and write files to directories in its “Set Path”.

By default this is in Documents/MATLAB/

Add or remove supplementary directories by clicking on the Set Path icon (red arrow).

```matlab
inputTempFah = 12
outputTempCels = (inputTempFah-32)*(5/9)
```

```
12
ans
12
```

```
outputTempCels = -11.1111
```

fx; >>
outputTempCels =
-11.1111

fx: >>
Summary

MATLAB Layout:
- Command line
- Workspace
- Script editor
- Directory
Data storage in MATLAB
Arrays
Arrays

- When programmers (that’s you!) are dealing with large amounts of data, you can use *data structures* to store and access data.
- MATLAB (and many other languages) often use arrays (a.k.a. matrices) to store data.
  - MATLAB = MATrix LABoratory
Data storage in MATLAB

The basic unit for representing information and data is the array.

Arrays are a useful organizational tool for storing arbitrary amounts of numbers inside of a single, structured unit.
Data storage in MATLAB

60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60
Data storage in MATLAB

Dimension 1 (size = 1)

60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60

Dimension 2 (size = 17)
How to create arrays in MATLAB
How to create arrays in MATLAB

Arrays are created in MATLAB by using double brackets:

myFirstArray = [ 1,2,3,4,5,6 ]
How to create arrays in MATLAB

Arrays are created in MATLAB by using double brackets:
myFirstArray = [ 1,2,3,4,5,6 ]
How to create arrays in MATLAB

Arrays are created in MATLAB by using double brackets:

myFirstArray = [ 1,2,3,4,5,6 ]
How to create arrays in MATLAB

Arrays are created in MATLAB by using double brackets:

\[
\text{myFirstArray} = \begin{bmatrix} 1,2,3,4,5,6 \end{bmatrix}
\]

```matlab
>> myArray = [1,2,3,4,5,6]
```
How to create arrays in MATLAB

Arrays are created in MATLAB by using double brackets:

```
myFirstArray = [1,2,3,4,5,6]
```

```
x >> myArray = [1,2,3,4,5,6]
```

```
Workspace

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>flyTracks</td>
<td>38946x1 double</td>
</tr>
<tr>
<td>message</td>
<td>'Hello world'</td>
</tr>
<tr>
<td>myArray</td>
<td>[1,2,3,4,5,6]</td>
</tr>
<tr>
<td>xPos</td>
<td>38946x1 double</td>
</tr>
</tbody>
</table>
```
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]
Data storage in MATLAB

myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]

figure;plot(myArray)
Data storage in MATLAB

OMG!
But what good is storage, if you can’t get the data out?
But what good is storage, if you can’t get the data out?

<3 INDEXING
<3 INDEXING

myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]
I want this value. What position is it in?
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]

Dimension 1 (size = 1)
Dimension 2 (size = 17)

(1, 6)
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]

To index variables in MATLAB
Type the variable name, followed by parentheses with your index position inside those parentheses.
Enter the command:

myArray2 = myArray( 1, : )

This command will return every element in first row of myArray
Colon operator :

The colon has multiple important functions in MATLAB
Colon operator

The colon has multiple important functions in MATLAB

1) Selecting all elements in a dimension of an array

myArray( 1 , : ) select the first row and all columns
Enter the command:

\[
\text{myArray3} = 1 : 1 : 100
\]

This command will create a list starting at 1, counting by 1, and ending at 100.
Enter the command:

myArray3 = 1 : 2 : 100

This command will create a list starting at 1, counting by 2, and ending at 100
Colon operator

The colon has multiple important functions in MATLAB

1) Selecting all elements in a dimension of an array
   myArray(1,:)  # go to the first row, select all rows

2) Creating lists of numbers
   myArray2 = 1:1:100  # make a list of numbers from 1 to 100 and count by 1s
Parentheses versus brackets in MATLAB
## Parentheses versus brackets in MATLAB

<table>
<thead>
<tr>
<th>Parentheses ( )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parentheses are used for:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brackets [ ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brackets are used to:</td>
</tr>
</tbody>
</table>
Parentheses versus brackets in MATLAB

### Parentheses ( )

Parentheses are used for:

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexing into an array</td>
<td>x(1:3)</td>
</tr>
<tr>
<td>Defining order of operations</td>
<td>(3+4)^2</td>
</tr>
<tr>
<td>Function inputs</td>
<td>mean(x)</td>
</tr>
</tbody>
</table>

### Brackets [ ]

Brackets are used to:
## Parentheses ()

Parentheses are used for:

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexing into an array</td>
<td><code>x(1:3)</code></td>
</tr>
<tr>
<td>Defining order of operations</td>
<td><code>(3+4)^2</code></td>
</tr>
<tr>
<td>Function inputs</td>
<td><code>mean(x)</code></td>
</tr>
</tbody>
</table>

## Brackets []

Brackets are used to:

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create an array or matrix</td>
<td><code>x = [1 2; 3 4]</code></td>
</tr>
<tr>
<td>Delete (excise) elements</td>
<td><code>x(x &lt; 0) = []</code></td>
</tr>
<tr>
<td>Group function outputs</td>
<td><code>[value index] = max(x)</code></td>
</tr>
</tbody>
</table>
Enter the command:

myArray4 = 1 : 2 : 100;

The semicolon (;) at the end of a command will suppress the displayed output. MATLAB will still execute the command even though the output isn’t displayed.
Semicolon operator

The semicolon has multiple important functions in MATLAB

1) Suppress a command’s displayed output
Semicolon operator

; 

The semicolon has multiple important functions in MATLAB

1) Suppress a command’s displayed output

2) Creating matrices
How to create **matrices** in MATLAB
How to create **matrices** in MATLAB

In MATLAB you can create a new row inside an array using the semi-colon 

;
How to create *matrices* in MATLAB

In MATLAB you can create a new row inside an array using the semi-colon

```
myMatrix =
[100,101,102,103;104,105,106,107;108,109,110,111]
```
How to create matrices in MATLAB

In MATLAB you can create a new row inside an array using the semi-colon

\[
\text{myMatrix} = \\
\begin{bmatrix}
100,101,102,103;104,105,106,107;108,109,110,111
\end{bmatrix}
\]
How to create matrices in MATLAB

In MATLAB you can create a new row inside an array using the semi-colon.

myMatrix =
[100,101,102,103;104,105,106,107;108,109,110,111]
Enter the command:

```matlab
figure; surf(myMatrix)
```
Cell-Arrays
Cell-Arrays

Two data sets of different sizes, but we’d like to store them in one variable

<table>
<thead>
<tr>
<th>Tiny image</th>
<th>Metadata on tiny image</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8147 0.2785 0.9572</td>
<td>0.7922 0.6557 0.8491 0.6787 0.7431 0.6555 0.7060</td>
</tr>
<tr>
<td>0.9058 0.5469 0.4854</td>
<td>0.9595 0.0357 0.9340 0.7577 0.3922 0.1712 0.0318</td>
</tr>
<tr>
<td>0.1270 0.9575 0.8003</td>
<td></td>
</tr>
<tr>
<td>0.9134 0.9649 0.1419</td>
<td></td>
</tr>
<tr>
<td>0.6324 0.1576 0.4218</td>
<td></td>
</tr>
<tr>
<td>0.0975 0.9706 0.9157</td>
<td></td>
</tr>
</tbody>
</table>
Cell-Arrays

Two data sets of different sizes, but we’d like to store them in one variable

BUT HOW?
Cell-Arrays

These are data types with indexed containers called “cells”. The user can instruct MATLAB to store large amounts of data in a specific cell.
Cell-Arrays

These are data types with indexed containers called “cells”. The user can instruct MATLAB to store large amounts of data in a specific cell.

Almost like an excel spreadsheet

Within each cell you can store numerical or string data of any size
Cell-Arrays

These are data types with indexed containers called “cells”. The user can instruct MATLAB to store large amounts of data in a specific cell.

```
>> myCell{1} = myMatrix
```
Cell-Arrays

These are data types with indexed containers called “cells”. The user can instruct MATLAB to store large amounts of data in a specific cell.

```
mystat {1} = myMatrix;
```
Types of Operations

• Conditional operations: instructions are carried out only if certain conditions are met

Learning to Program:
1. Check your calendar
2. IF it is Monday or Wednesday, attend iPBR class
3. Tackle coding assignments
4. Become master programmer
Logical (boolean):

A variable that has two values: 
true or false
**Logical** (boolean):

A variable that has two values: **true** or **false**

MATLAB interprets this as either: 1 or 0
**Logical (boolean):**

Operators for logical comparison:

<table>
<thead>
<tr>
<th>Logical operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>is less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>is greater than</td>
</tr>
<tr>
<td>&gt;=</td>
<td>is greater than or equal to</td>
</tr>
<tr>
<td>&lt;=</td>
<td>is less than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>is equal to</td>
</tr>
<tr>
<td>~</td>
<td>not equal to</td>
</tr>
<tr>
<td>&amp;</td>
<td>and</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conditional statements:

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an if statement.

Will I go to class today?
Conditional statements:

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an `if` statement.

Will I go to class today?
```java
if freeFood == true
    MattAttend = true
```

This conditional statement starts with `if`, followed by a logical
Conditional statements:

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an `if` statement.

Will I go to class today?

```java
if freeFood == true
    MattAttend = true
elseif interestingSpeaker == true | Georgia_Attend == true
    MattAttend = true
```
Conditional statements:
Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an `if` statement.

Will I go to class today?
```java
if freeFood == true
    MattAttend = true
elseif interestingSpeaker == true | Georgia_Attend == true
    MattAttend = true
```

The second conditional statement uses `ELSEIF` If the first logical evaluates false, then evaluate this `ELSEIF` statement. Perform the operation under the `ELSEIF` clause, if the logical evaluates true.
Conditional statements:

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an if statement.

Will I go to class today?
if freeFood == true
    MattAttend = true
elseif interestingSpeaker == true | Georgia_Attend == true
    MattAttend = true
else
    MattAttend = false
Conditional statements:

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an `if` statement.

Will I go to class today?

```java
if freeFood == true
    MattAttend = true
```

```java
elseif interestingSpeaker == true | Georgia_Attend == true
    MattAttend = true
```

```java
else
    MattAttend = false
```

The last conditional statement uses `else`. If all ELSE fails (every other statement evaluates false), then perform the commands below `else`.
Conditional statements:

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an if statement.

Will I go to class today?
if freeFood == true
  MattAttend = true
elseif interestingSpeaker == true | Georgia_Attend == true
  MattAttend = true
else
  MattAttend = false
end
Conditional statements:

Conditional statements enable you to select at run time which block of code to execute. The simplest conditional statement is an if statement.

Will I go to class today?

```matlab
if freeFood == true
    MattAttend = true
elseif interestingSpeaker == true | Georgia_Attend == true
    MattAttend = true
else
    MattAttend = false
end
```

Tell MATLAB to end a conditional statement by typing **end**
Conditional statements:

myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]
Conditional statements:

myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]

if sum(myArray) > 1000
    print('The neuron has spiked."
elseif sum(myArray) < -500
    print('The neuron has been inhibited."
else
    print('There is no significant change."
end
Conditional statements:

```matlab
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ];

if sum(myArray) > 1000
    disp('The neuron has spiked.');
elseif sum(myArray) < -500
    disp('The neuron has been inhibited.');</nelse
    disp('There is no significant change.');</nend
```

The output of this series of commands will be:
The neuron has spiked

MATLAB evaluates the first conditional statement and finds the statement to be true so it evaluates the command beneath it.
Types of Operations

- Iterative operations: instructions are carried out repeatedly

Learning to Program:
1. Attend iPBR lectures and review sessions
2. Tackle coding assignments
3. REPEAT steps 1 and 2 until August 2
4. Become master programmer
How can I repeat the same lines of code?
Scenario: You’re looking for warm places to travel. You look online at all the trendy and hip spots around the world, but all the recorded temperature data is in Celsius.

You have a friend that can do the temperature conversions in their head extremely quickly. You must instruct your friend to convert the temperatures listed below.

WHAT DO YOU SAY?

12, 6, 130, 273, -34

“Hello, friend. Could you use your conversion equation FOR the temperatures: 12, 6, 130, 273, and -34
Loop Control

Used to repeatedly execute a block of code

Two loop control operators:
Loop Control

Used to repeatedly execute a block of code

Two loop control operators:

For

While
Loop Control: *for*

A *for*-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.
A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

In MATLAB, all for-loops start with:
for VariableName = [series of numbers]
Loop Control: for

A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

In MATLAB, all for-loops start with:

```matlab
for VariableName = [series of numbers]
    MATLAB commands you want to execute and repeat
```

32)*(5/9)
Loop Control: *for*

A *for*-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

In MATLAB, all *for*-loops start with:

```matlab
for VariableName = [series of numbers]
    MATLAB commands you want to execute and repeat
end
```

```
32)*(5/9)
```

```matlab
end
```
Loop Control: `for`

A `for`-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

In MATLAB, all `for`-loops start with:
```
for VariableName = [series of numbers]
    MATLAB commands you want to execute and repeat
end
```

Write MATLAB code to iterate your temp conversion algorithm over the five numerical values:
```
12, 6, 130, 273, -34
```

My temp conversion algorithm:
```
inputTempFah = 12
outputTempCels = (inputTempFah-32)*(5/9)
```
Loop Control: *for*

A *for*-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

In MATLAB, all *for*-loops start with:

```matlab
for  inputTempFah = [series of numbers]
   MATLAB commands you want to execute and repeat
end
```

My temp conversion algorithm:
```matlab
inputTempFah = 12
outputTempCels = (inputTempFah-32)*(5/9)
```
Loop Control: for

A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

In MATLAB, all for-loops start with:

```
for inputTempFah = [12, 6 130, 273, -34]
    MATLAB commands you want to execute and repeat
End
```

My temp conversion algorithm:
```matlab
inputTempFah = 12
outputTempCels = (inputTempFah-32)*(5/9)
```
Loop Control: **for**

A **for**-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

In MATLAB, all **for**-loops start with:
```
for inputTempFah = [12, 6 130, 273, -34]
    outputTempCels = (inputTempFah-32)*(5/9)
end
```

My temp conversion algorithm:
```
inputTempFah = 12
outputTempCels = (inputTempFah-32)*(5/9)
```
A for-loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly.

In MATLAB, all for-loops start with:

```matlab
for inputTempFah = [12, 6 130, 273, -34]
    outputTempCels = (inputTempFah-32)*(5/9)
end
```

In the scenario (slide 143), I instruct my friend to convert a set of values by using the word for. “for the number 12, 6..etc”

To the left is represents the same idea in MATLAB code. MATLAB will execute the commands inside the for-loop with the value of the inputTempFah changing for each iteration of the loop.
Coming Up Next

• Congrats on finishing part 1 of intro to MATLAB
• Next week we will continue with part 2:
  – While loops
  – Executing/writing functions
  – Basic plotting
• Problem set 2 will be released tomorrow morning
• Review session on Monday at 7pm (243NW)