



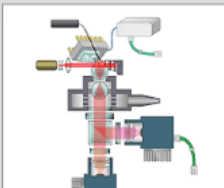
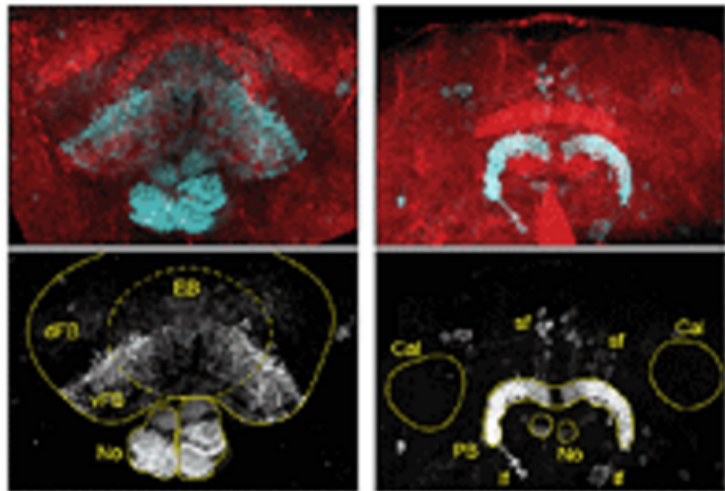
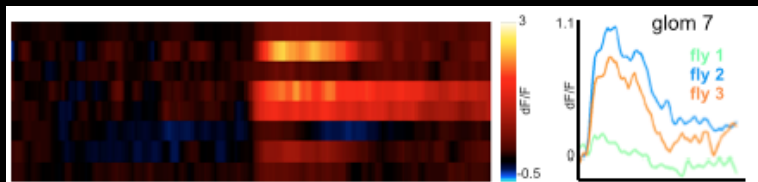
# Introduction to Programming for Biology Research



Department of Organismic and Evolutionary Biology

Center for Brain Science

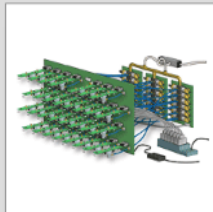
Harvard University



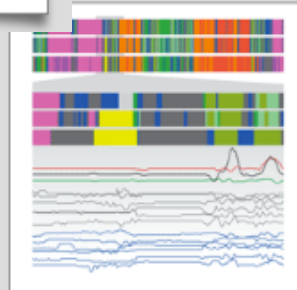
**LegTracker**  
an instrument for recording the position of all 6 of a fly's legs, in real time, at 80Hz



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



**automated behavioral classification**  
classifications of spontaneous behavior, assigned by human investigators and machine learning algorithms



**variation across species**

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

**effect of weather on behavior**

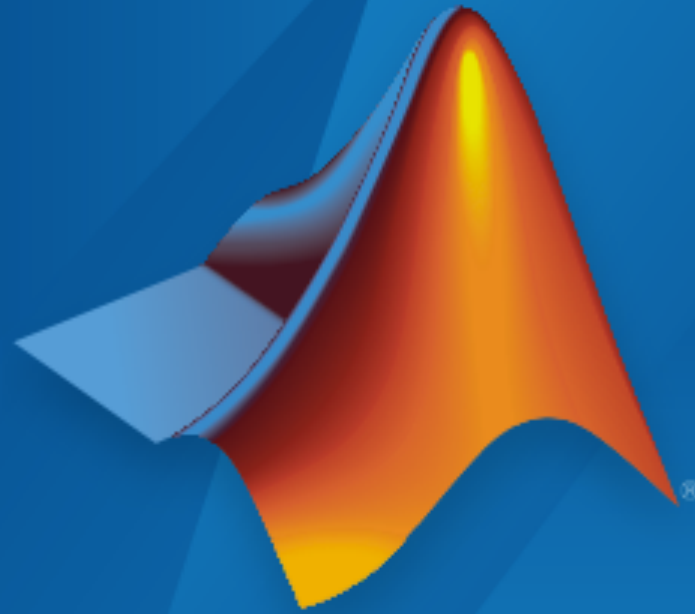
predicted fly population dynamics and phototactic behavior dynamics as influenced by real world weather conditions from 2008

# Introduction to MATLAB: part I

## MATLAB Basics

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





MATLAB®

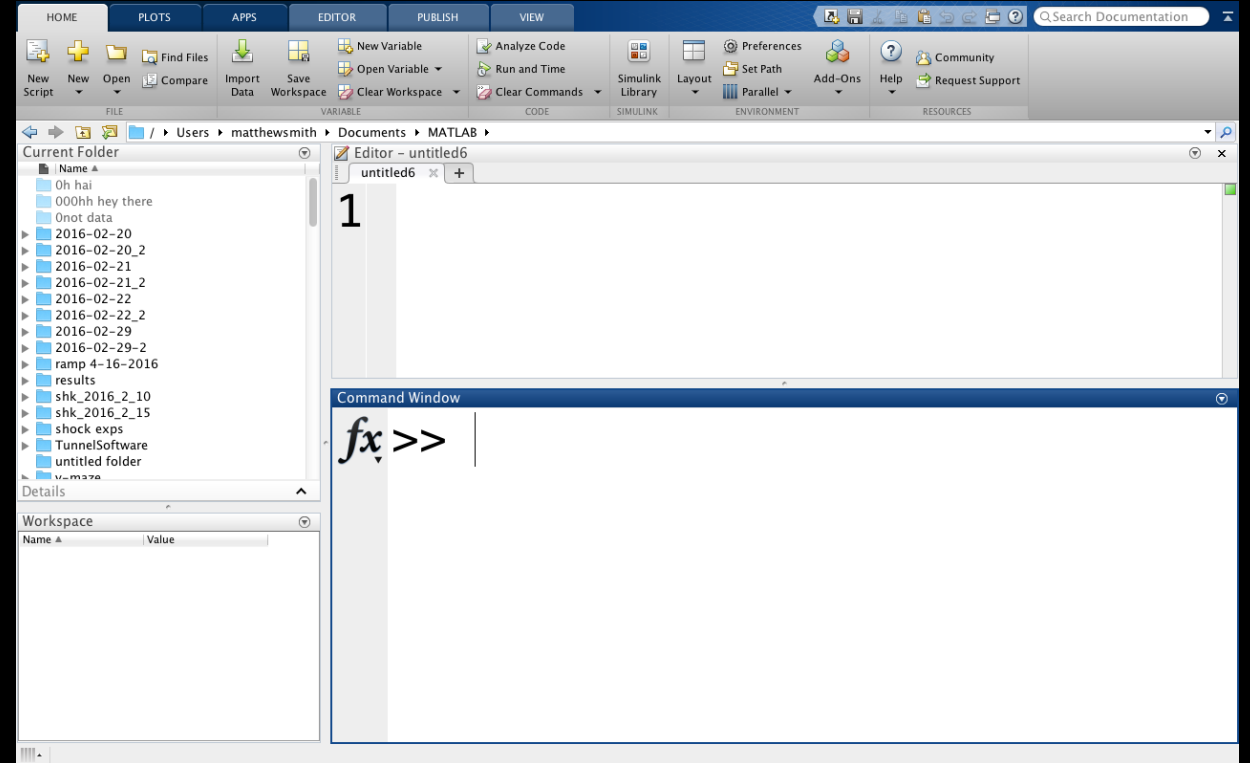
 MathWorks®

R2015a

# MATLAB: The interface

## 4 Default windows:

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



# Command window

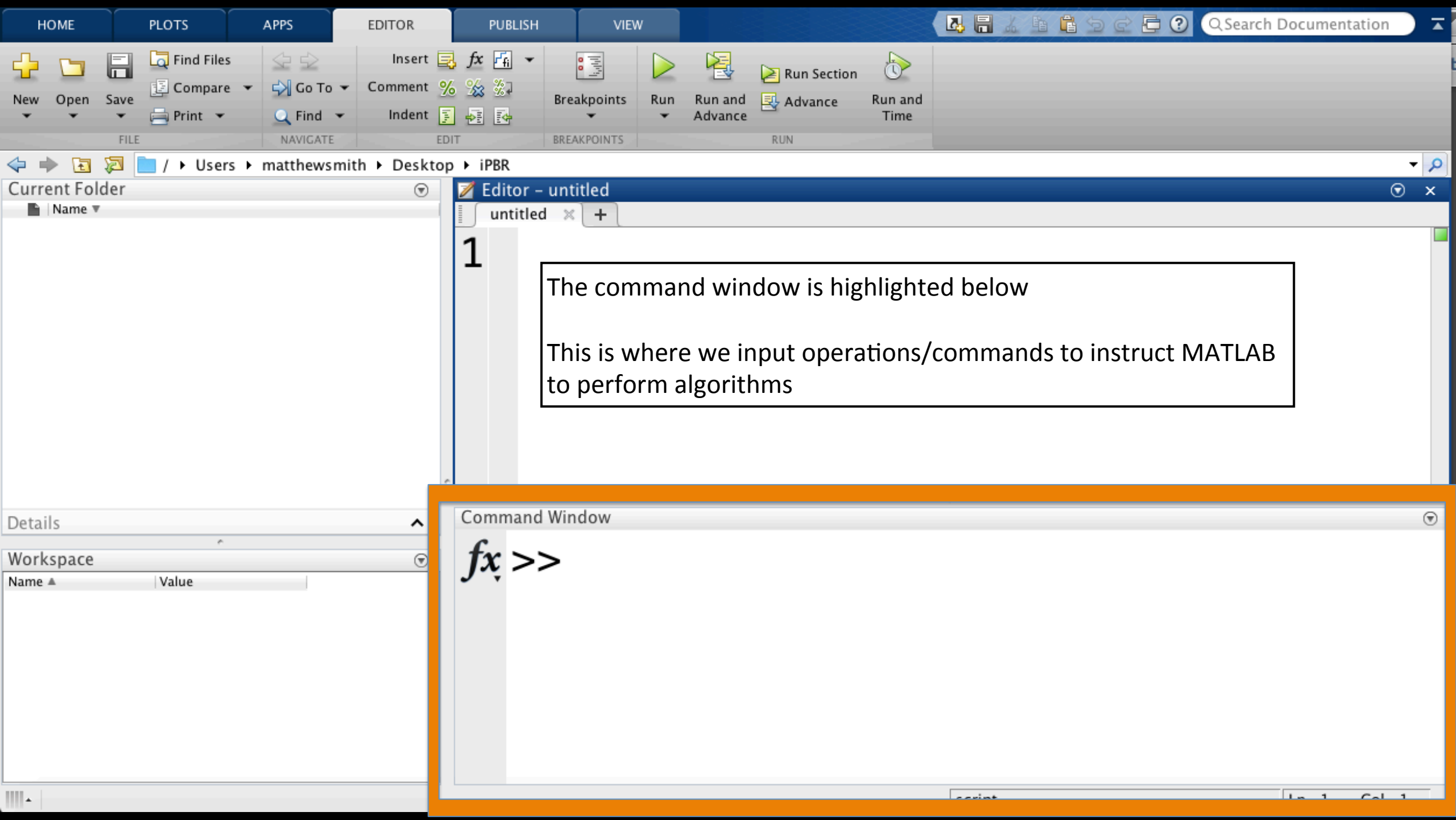
The command window is where one can type commands to  
MATLAB

# Algorithms

# 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





1

The command window is highlighted below

This is where we input operations/commands to instruct MATLAB to perform algorithms

Command Window

*fx* >>

The image shows a software development environment with a blue-themed interface. At the top, there are tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below these are various toolbars with icons for file operations (New, Open, Save, Find Files, Compare, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), breakpoints, and running code (Run, Run and Advance, Run Section, Run and Time). A search bar for documentation is in the top right corner.

The main workspace is divided into several panes. On the left, there is a 'Current Folder' pane showing a file tree for 'Users > matthewsmith > Desktop > iPBR'. Below it is a 'Details' pane and a 'Workspace' pane with columns for 'Name' and 'Value'. The central pane is an 'Editor - untitled2' window showing a single line of code on line 1: `1`. Below the editor is a 'Command Window' pane with a blue header. It contains the command: `fx >> myFirstCommand = 'i <3 iPBR'`. The `fx` and `>>` are in black, and the rest of the command is in purple.

A text box with a black border is overlaid on the Command Window, containing the text: "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 computers 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 computers 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 computers 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

File Edit Breakpoints Run

New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

Editor - untitled2

```
1
```

Command Window

```
fx >> myFirstCommand = 'i <3 iPBR'
```

Details

Workspace

Name	Value
------	-------

The image shows the MATLAB IDE interface. At the top, there are tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below these are various toolbars for file operations (New, Open, Save, Find Files, Compare, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), breakpoints, and running code (Run, Run and Advance, Run Section, Run and Time). The current folder is set to 'Users > matthewsmith > Desktop > iPBR'. The editor window shows a file named 'untitled2' with a single line of code: `1`. The Command Window displays the command: `>> myFirstCommand = 'i <3 iPBR'`. The workspace window shows a variable named 'myFirstComm...' with the value 'i <3 iPBR'. A text box on the right explains the command syntax.

```
>> myFirstCommand = 'i <3 iPBR'
```

myFirstCommand =

i <3 iPBR

*fx* >>

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'



HOME PLOTS APPS

Q Search Documentation

New Script New Open Compare Import Data Save Workspace New Variable Open Variable Clear Workspace Analyze Code Run and Time Clear Commands Simulink Library Layout Parallel Preferences Set Path Add-Ons Help Community Request Support

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

- Name
- iPBRtempCalc.m

Details ^

Workspace

Name ▲	Value
--------	-------

Command Window

```
fx >> 'matt <3' = statement
```



Current Folder

- Name
- ipBRtempCalc.m

Details

Workspace

Name	Value
------	-------

Command Window

```
>> 'matt <3' = statement
      'matt <3' = statement
            ↑
Error: The expression to the left of the
equals sign is not a valid target for an
assignment.
```

*fx* >>

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.

File Edit Breakpoints Run

New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

```
Editor - untitled2  
untitled2 x +  
1
```

Command Window

```
fx >>
```

Details ^

Workspace

Name	Value
myFirstComm...	'i <3 iPBR'

File Edit Breakpoints Run

New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

```
Editor - untitled2  
untitled2 x +  
1
```

Command Window

```
fx >> myFirstCommand
```

By typing a variable name into the command window you can recall it's saved value

Details

Workspace

Name	Value
myFirstComm...	'i <3 iPBR'

File Edit Breakpoints Run

New Open Save Find Files Compare Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

Details

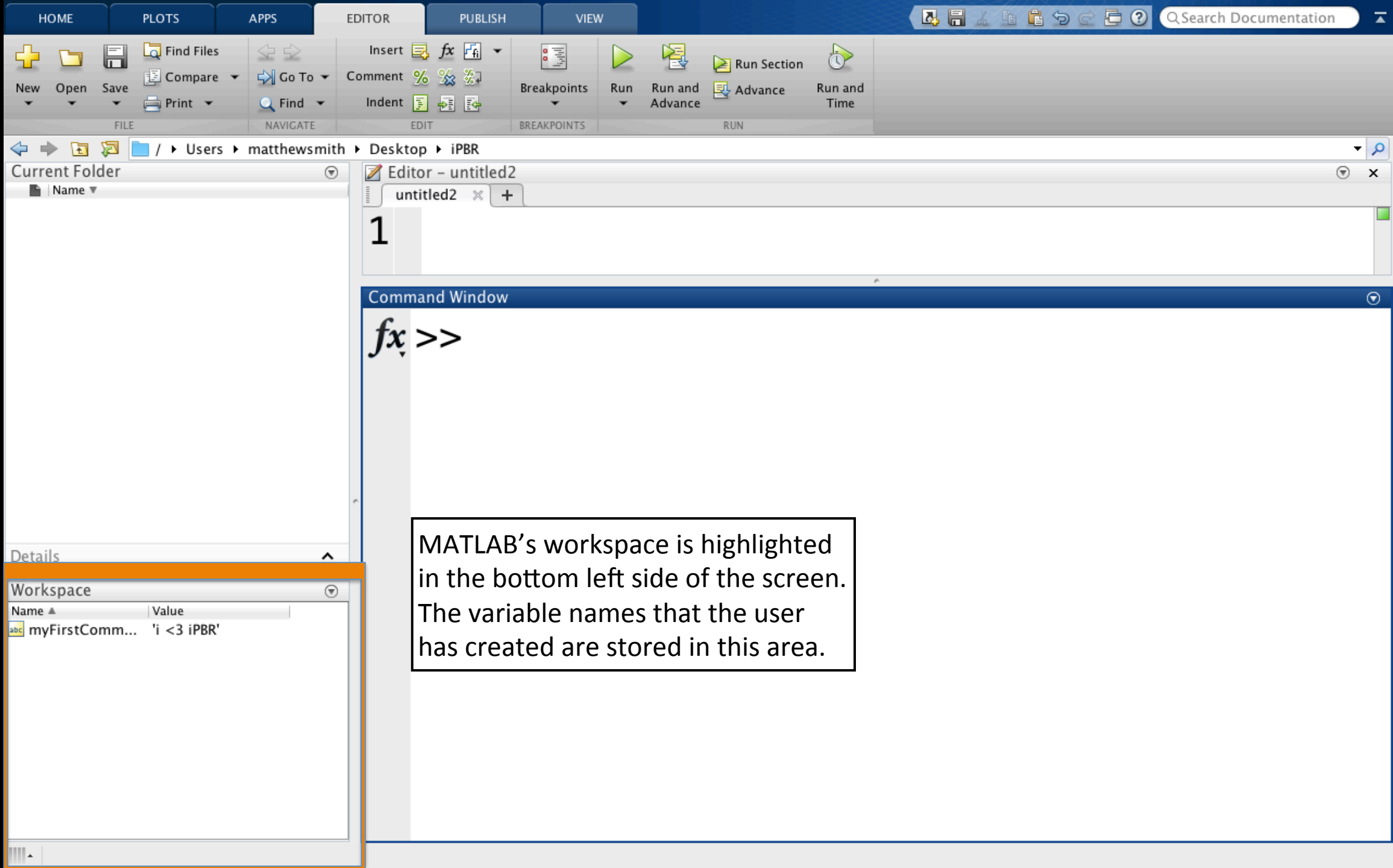
Workspace

Name	Value
myFirstComm...	'i <3 iPBR'


```
Editor - untitled2  
untitled2  
1
```

Command Window

```
>> 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.

Workspace	
Name ▲	Value
 myFirstComm...	'i <3 iPBR'

# 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.

File Edit Breakpoints Run

New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

```
Editor - untitled2  
untitled2 x +  
1
```

Command Window

```
fx >>
```

Details

Workspace

Name	Value
myFirstComm...	'i <3 iPBR'

File Edit Breakpoints Run

New Open Save Find Files Compare Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

```
Editor - untitled2  
untitled2 x +  
1
```

Command Window

```
fx >> x = 7200
```

This command has created a variable named x  
It is assigning a value of 7200 to the variable x

Details

Workspace

Name	Value
myFirstComm...	'i <3 iPBR'

FILE NAVIGATE EDIT BREAKPOINTS RUN

New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

Details

```
Editor - untitled2  
untitled2 x +  
1
```

Command Window

```
>> x = 7200  
  
x =  
  
7200
```

*fx* >> If you look in the command window, you can see that the variable x is now in MATLAB's workspace

Workspace

Name	Value
myFirstComm...	'i <3 iPBR'
x	7200

File navigation: New, Open, Save, Find Files, Compare, Print, Go To, Find

Edit: Insert, Comment, Indent, Breakpoints

Run: Run, Run and Advance, Run Section, Advance, Run and Time

Navigation: / > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

Details

Workspace

Name	Value
myFirstComm...	'i <3 iPBR'
x	7200

```
Editor - untitled2  
untitled2 x +  
1
```

Command Window

```
>> x = 7200  
  
x =  
  
7200  
  
fx >> y = 35
```

File Edit Breakpoints Run

New Open Save Find Files Compare Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name

Details

Workspace

Name	Value
myFirstComm...	'i <3 iPBR'
x	7200
y	35

Editor - untitled2

untitled2

1

Command Window

```
x =  
  
7200  
  
>> y = 35  
  
y =  
  
35  
  
fx >>
```

The image shows the MATLAB IDE interface. At the top, there are tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below these are various toolbars for file operations, navigation, editing, breakpoints, and running code. The main workspace is divided into several panes:

- Current Folder:** Shows the path `/ > Users > matthewsmith > Desktop > iPBR`.
- Editor - untitled2:** Contains a single line of code: `1`.
- Command Window:** Displays the command `fx >> x + y`.
- Workspace:** A table showing the current state of variables in the workspace.

A text box with a black border is overlaid on the Command Window, containing the following text:

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

Name	Value
myFirstComm...	'i <3 iPBR'
x	7200
y	35



HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

Users > matthewsmith > Desktop > iPBR

Current Folder

Name

Details

Workspace

Name	Value
ans	7235
myFirstComm...	'i <3 iPBR'
x	7200
y	35

```
1
```

Command Window

```
>> x + y  
  
ans =  
  
7235
```

*fx* >> 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

File navigation and editing tools: New, Open, Save, Find Files, Compare, Print, Go To, Find, Insert, Comment, Indent, Breakpoints, Run, Run and Advance, Run Section, Advance, Run and Time.

Navigation bar: / > Users > matthewsmith > Desktop > iPBR

Current Folder view showing a file explorer interface with a search bar and a list of files.

Details

Workspace

Name	Value
ans	7235
myFirstComm...	'i <3 iPBR'
x	7200
y	35

```
Editor - untitled2  
untitled2  
1
```

Command Window

```
fx >>
```

File Edit Breakpoints Run

New Open Save Find Files Compare Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name

Details

Workspace

Name	Value
ans	7235
myFirstComm...	'i <3 iPBR'
x	7200
y	35

Editor - untitled2

```
1
```

Command Window

```
fx >> x*y
```

The image shows the MATLAB IDE interface. At the top, there are tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. Below these are various toolbars for file operations, navigation, editing, breakpoints, and running code. The main workspace is divided into several panes:

- Current Folder:** Shows the path `/ > Users > matthewsmith > Desktop > iPBR`.
- Editor - untitled2:** Contains a single line of code: `1`.
- Command Window:** Shows the execution of the command `>> x*y`, resulting in `ans = 252000`.
- Workspace:** A table showing the current state of variables in the workspace.

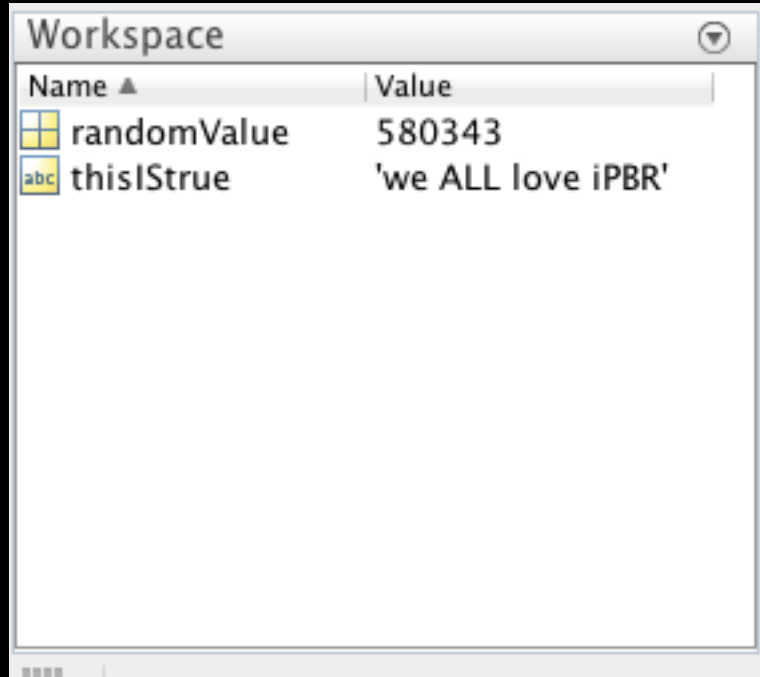
Name	Value
ans	252000
myFirstComm...	'i <3 iPBR'
x	7200
y	35

*fx* >>

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

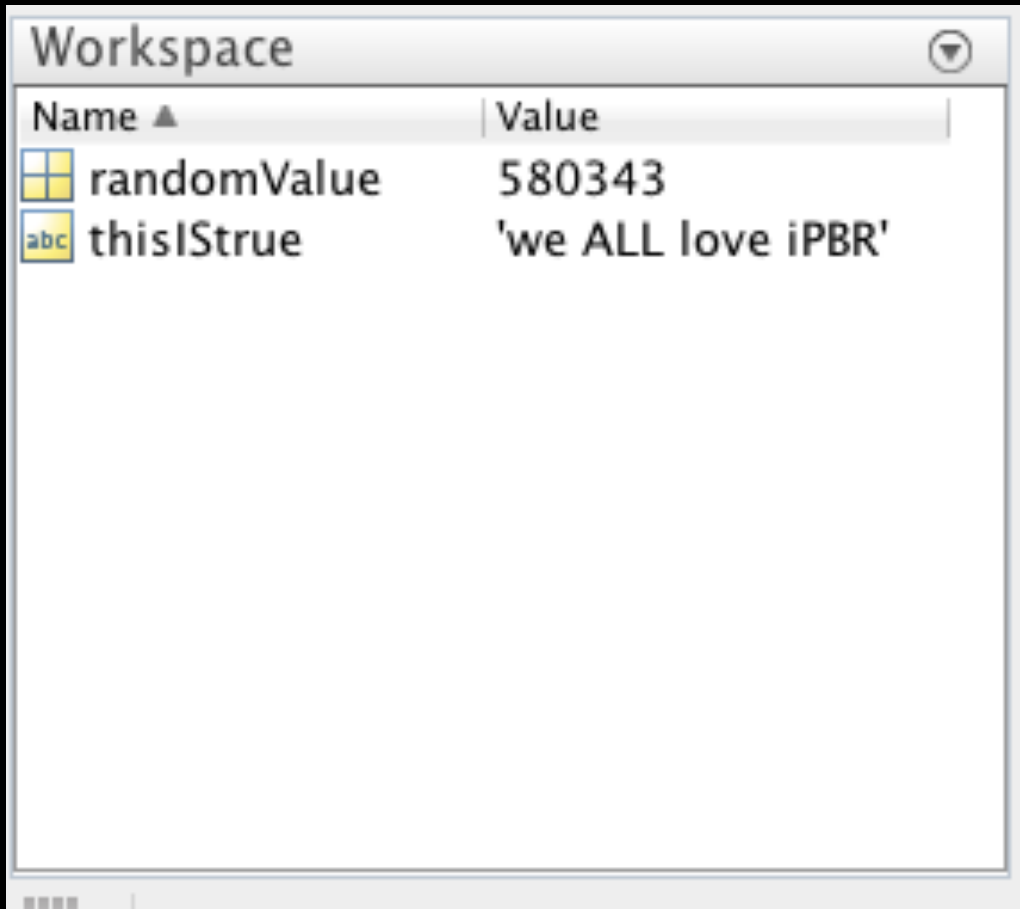


A screenshot of a workspace window titled "Workspace". It displays a table with two columns: "Name" and "Value". The table contains two rows of data. The first row shows a variable named "randomValue" with a value of "580343". The second row shows a variable named "thisIsTrue" with a value of "'we ALL love iPBR'".

Name ▲	Value
randomValue	580343
thisIsTrue	'we ALL love iPBR'

# Variable Types

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

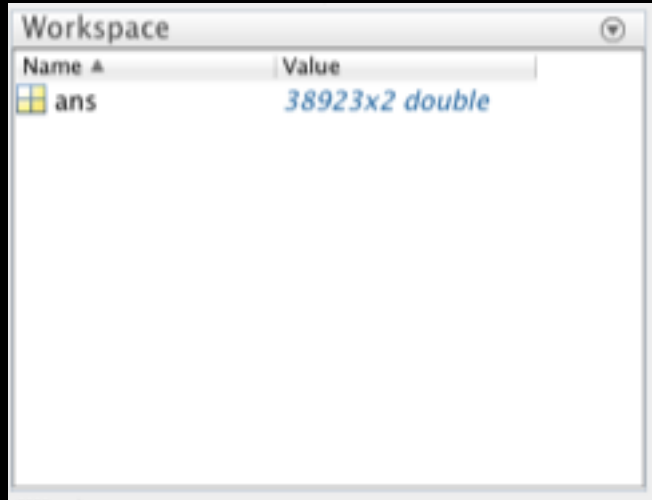


The image shows a 'Workspace' window with a table of variables. The table has two columns: 'Name' and 'Value'. The first row shows a variable named 'randomValue' with a value of 580343. The second row shows a variable named 'thisIstrue' with a value of 'we ALL love iPBR'. Each row has a small icon to its left: a yellow square with a white grid for 'randomValue' and a yellow square with the letters 'abc' for 'thisIstrue'.

Name ▲	Value
randomValue	580343
thisIstrue	'we ALL love iPBR'

# Variable Types

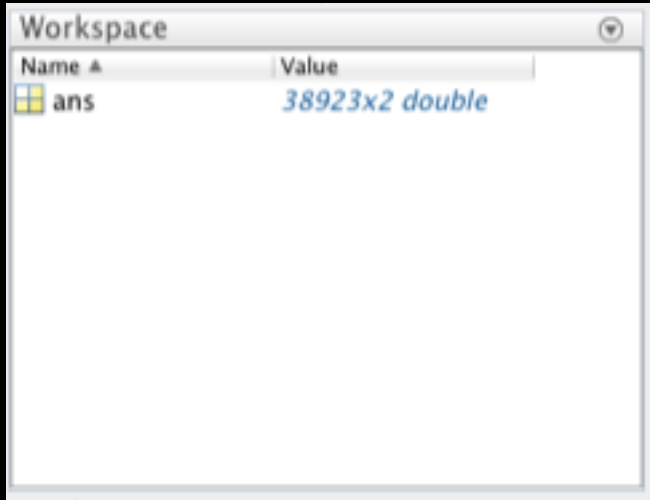
**Numeric data:** integer, double, float



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

# Variable Types

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



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

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

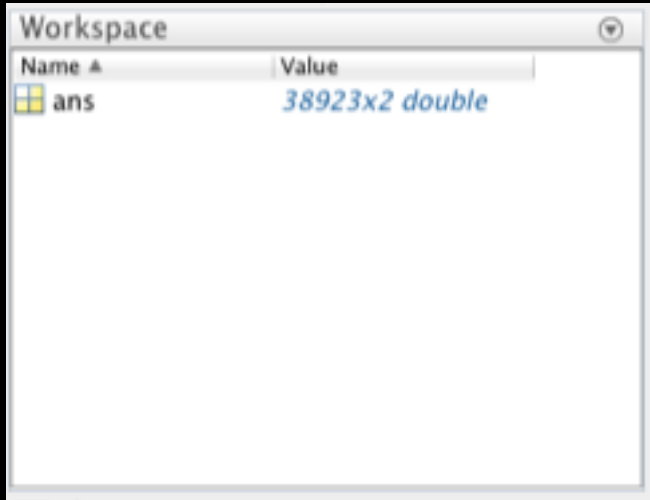
Create a string in MATLAB by using single quotations:

**'This is a string'**



# Variable Types

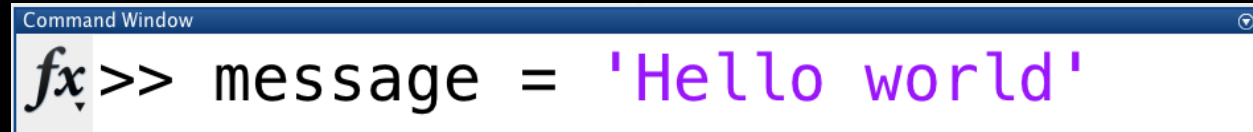
**Numeric data:** integer, double, float



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

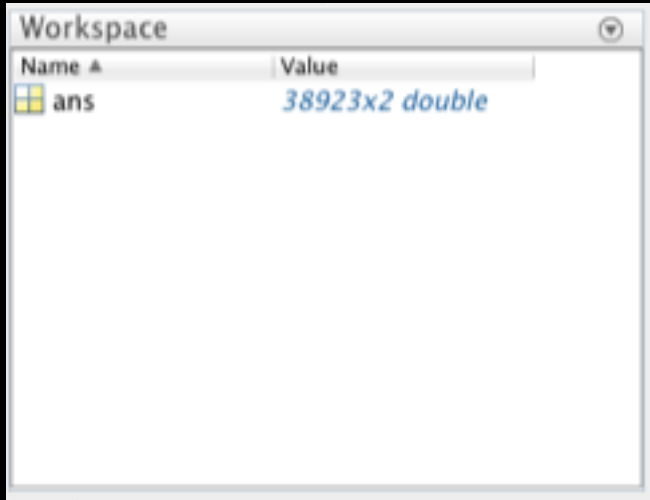
**Text data:** Sequence of characters, normally called a string. Strings are treated as arrays that contain characters.

Create a string in MATLAB by using single quotations:

A screenshot of the MATLAB Command Window. The window has a title bar that says "Command Window". The command prompt shows the command `message = 'Hello world'` being entered. The prompt is `fx>>`.

# Variable Types

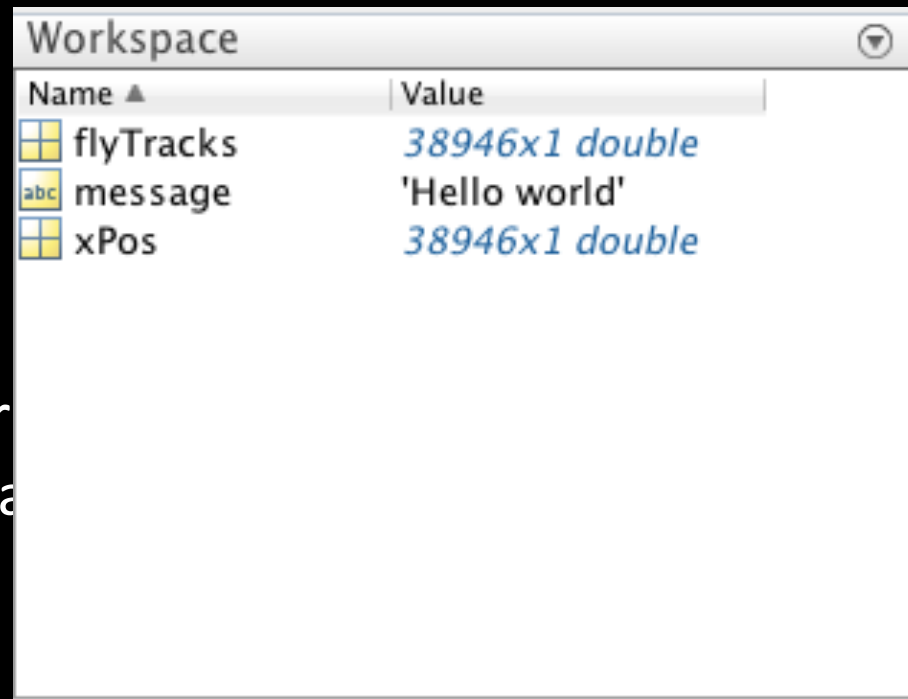
**Numeric data:** integer, double, float



Name ▲	Value
ans	38923x2 double

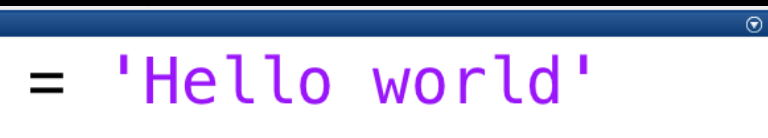
Basic mathematical operators:  
addition, subtraction, multiplication

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



Name ▲	Value
flyTracks	38946x1 double
message	'Hello world'
xPos	38946x1 double

Represent a string in MATLAB by single quotations:



```
= '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

## 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:  
$$\text{celsius} = \text{input temperature} - 32 * (5/9)$$



## Temperature conversion algorithm:

- 1) Select an input temperature in Fahrenheit to convert to Celsius
- 2) Perform the operation:  
$$\text{celsius} = \text{input temperature} - 32 * (5/9)$$

## Temperature conversion algorithm: in MATLAB

- 1) `inputTempFah = 45.6`
- 2) `outputTempCels = inputTempFah - 32 * (5/9)`

File Edit Breakpoints Run

New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder

Name
------

Editor - untitled3

iPBRtempCalc.m x handle\_.m x untitled3 x +

iPBRtempCalc.m (Script)

Workspace

Name	Value
matrix	3x4 double
neuron	1x17 double
t	4x4 double

```
fx >> inputTempFah = 45.6  
outputTempCels = (inputTempFah-32)*(5/9)
```

The screenshot displays the MATLAB software interface. At the top, there are tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, showing a toolbar with various icons for file operations (New, Open, Save, Find Files, Compare, Print), navigation (Go To, Find), editing (Insert, Comment, Indent), and execution (Run, Run and Advance, Run Section, Advance, Run and Time). A red arrow points to the 'Insert' icon in the toolbar. Below the toolbar is a file browser showing the current folder path: / > Users > matthewsmith > Desktop > iPBR. The main editor window shows a single line of code: `1`. Below the editor is the Command Window, which contains the prompt `fx >>`. A red box highlights the Command Window area, containing the following text:

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)

On the left side of the interface, there is a 'Workspace' window showing a table of variables:

Name	Value
ans	252000
myFirstComm...	'i <3 iPBR'
x	7200
y	35

# 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.

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Comment Indent Breakpoints Run Run and Advance Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

Users > matthewsmith > Desktop > iPBR

Current Folder  
Name  
iPBRtempCalc.m\*

```
Editor - /Users/matthewsmith/Desktop/iPBR/iPBRtempCalc.m*  
iPBRtempCalc.m* x +  
1 — inputTempFah = 12  
2 — outputTempCels = (inputTempFah-32)*(5/9)
```

Details ^

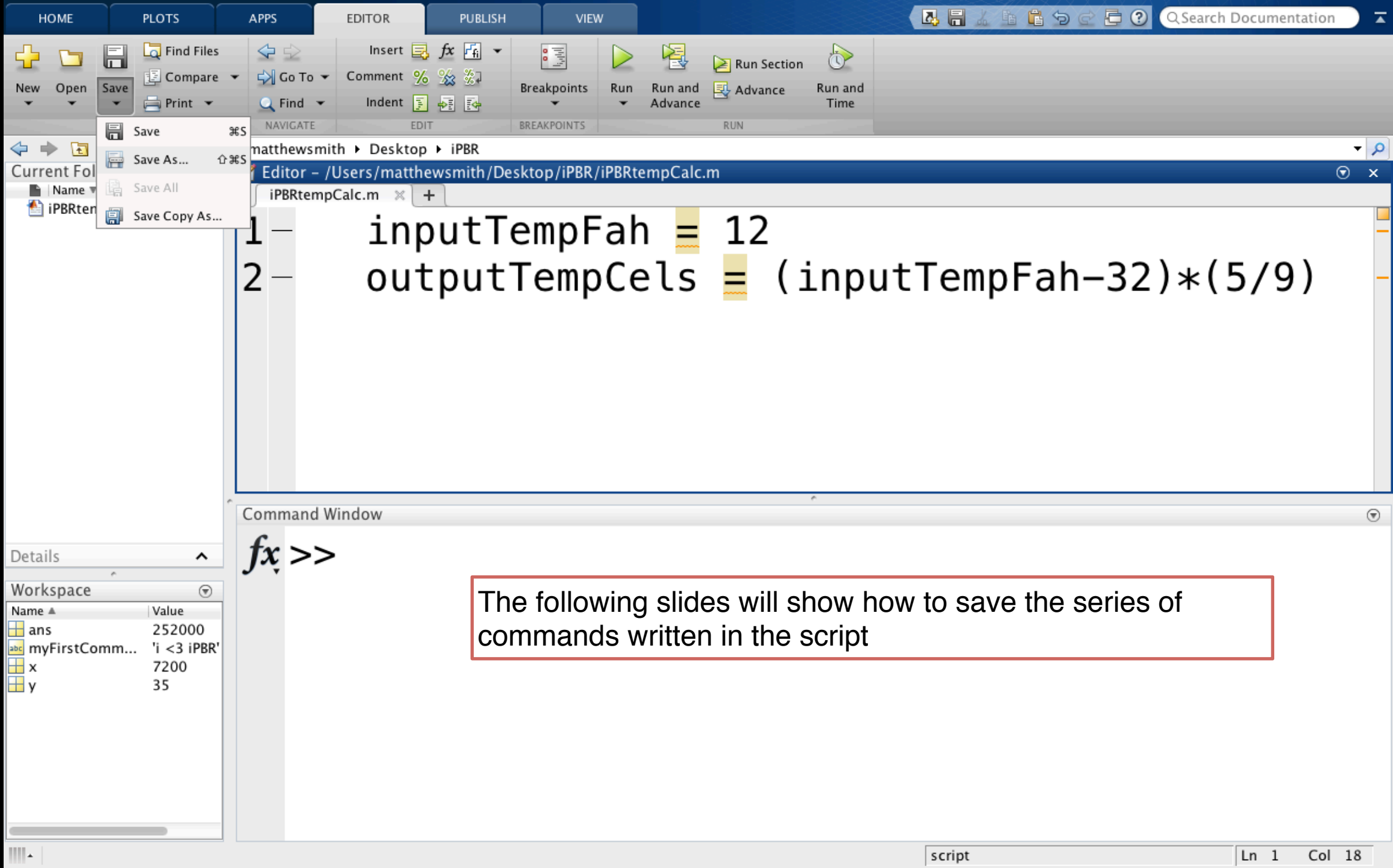
Workspace

Name	Value
ans	252000
myFirstComm...	'i <3 iPBR'
x	7200
y	35

Command Window

*fx* >>

The temperature conversion algorithm has now been moved from the command window to the script editor window.



HOME PLOTS APPS

New Open Save Compare Print Find

FILE NAVIGAT

Current Folder / Users > matthews

Editor untitle

1  
2

Command *fx* >

Details ^

Workspace

Name	Value
ans	252000
myFirstComm...	'i <3 iPBR'
x	7200
y	35

Select File for Save As

Save As: iPBRtempCalc

Tags:

iPBR Search

Favorites

- Dropbox
- iCloud Drive
- Applications
- Desktop
- Documents
- Downloads

Devices

Shared

- All...

Tags

- Red
- Orange
- Yellow
- Green
- Blue
- Purple
- Gray

Name Date Modified

Format: MATLAB Code files (\*.m)

Hide extension New Folder Cancel Save

Search Documentation

-32)\*(5/9);

script Ln 2 Col 42

File Edit Breakpoints Run

New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

/ > Users > matthewsmith > Desktop > iPBR

Current Folder  
Name  
iPBRtempCalc.m

```
Editor - /Users/matthewsmith/Desktop/iPBR/iPBRtempCalc.m  
iPBRtempCalc.m  
1 — inputTempFah = 12  
2 — outputTempCels = (inputTempFah-32)*(5/9)
```

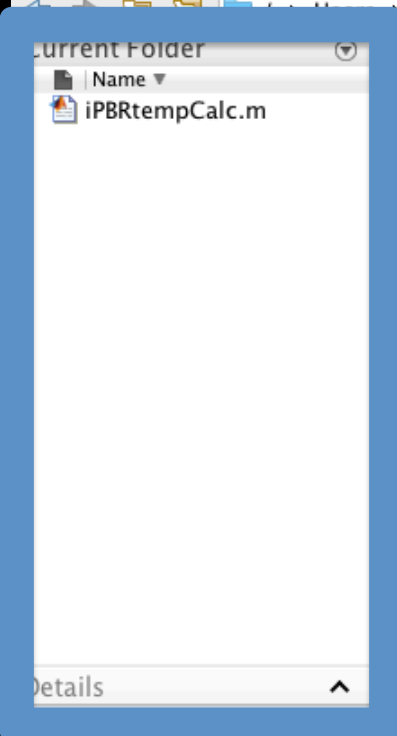
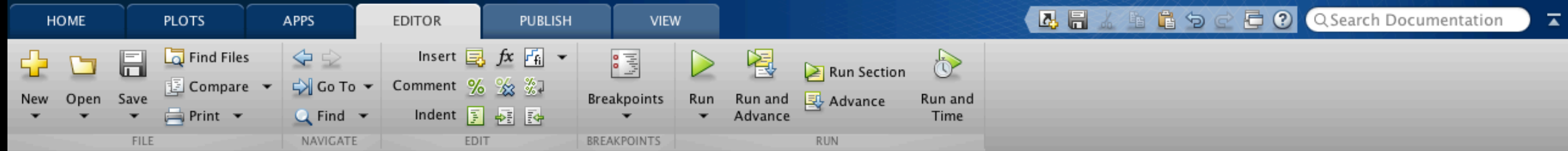
Details ^

Workspace

Name	Value
ans	252000
myFirstComm...	'i <3 iPBR'
x	7200
y	35

Command Window  
*fx* >>





```
matthewsmith > Desktop > iPBR
Editor - /Users/matthewsmith/Desktop/iPBR/iPBRtempCalc.m
iPBRtempCalc.m x +
1 — inputTempFah = 12
2 — outputTempCels = (inputTempFah-32)*(5/9)
```

Command Window  
*fx* >>

Our saved script now appears in MATLAB's current folder, which is where MATLAB will read and write files

workspace

Name	Value
ans	252000
myFirstComm...	'i <3 iPBR'
x	7200
y	35

# 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".

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Comment Indent Breakpoints Run Run and Advance Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

Users > matthewsmith > Desktop > iPBR

Current Folder

- Name
- iPBRtempCalc.m

```
Editor - /Users/matthewsmith/Desktop/iPBR/iPBRtempCalc.m  
iPBRtempCalc.m x +  
1 — inputTempFah = 12  
2 — outputTempCels = (inputTempFah-32)*(5/9)
```

Details ^

Workspace

Name	Value
ans	252000
myFirstComm...	'i <3 iPBR'
x	7200
y	35

Command Window

```
fx >> iPBRtempCalc
```

Type the script name into the command window and press enter to recall the series of commands saved in your script

File Edit Breakpoints Run

New Open Save Find Files Compare Print Go To Find Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

Users > matthewsmith > Desktop > iPBR

Current Folder

- ipBRtempCalc.m

Details

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

Workspace

Name	Value
ans	252000
inputTempFah	12
myFirstComm...	'i <3 iPBR'
outputTempCels	-11.1111
x	7200
y	35

Command Window

```
inputTempFah =
12
outputTempCels =
-11.1111
fx >>
```

Current Folder

- Name
- iPBRtempCalc.m

Details

```
Editor - /Users/matthewsmith/Desktop/iPBR/iPBRtempCalc.m  
iPBRtempCalc.m x +  
1 - inputTempFah = 12  
2 - outputTempCels = (inputTempFah-32)*(5/9)
```

Command Window

```
inputTempFah =  
  
12  
  
outputTempCels =  
  
-11.1111  
  
fx >>
```

Workspace

Name	Value
ans	252000
inputTempFah	12
myFirstComm...	'i <3 iPBR'
outputTempCels	-11.1111
x	7200
y	35

HOME PLOTS APPS EDITOR PUBLISH VIEW

New Script New Open Compare Import Data Save Workspace New Variable Open Variable Clear Workspace Analyze Code Run and Time Clear Commands Simulink Library Layout Preferences Set Path Parallel Add-Ons Help Community Request Support

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Users > matthewsmith > Desktop > iPBR

Current Folder

Name
iPBRtempCalc.m

Details

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

Workspace

Name	Value
ans	252000
inputTempFah	12
myFirstComm...	'i <3 iPBR'
outputTempCels	-11.1111
x	7200
y	35

Command Window

```
inputTempFah =
    12

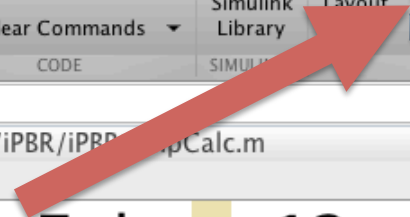
outputTempCels =
   -11.1111

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)



New Script New Open Compare Import Data

New Variable Open Variable Analyze Code Run and Time Preferences Set Path

FILE / > Users > mat

Current Folder  
Name  
ipBRtempCalc.m

1  
2

Com

Details ^

Workspace

Name	Value
ans	252000
inputTempFah	12
myFirstComm...	'i <3 iPBR'
outputTempCels	-11.1111
x	7200
y	35

Set Path

All changes take effect immediately.

MATLAB search path:

- /Users/matthewsmith/Documents/MATLAB
- /Users/matthewsmith/Desktop/Experiments
- /Users/matthewsmith/Desktop/Experiments/08162016\_odorLrn
- /Users/matthewsmith/Desktop/Experiments/08162016\_odorLrn/d1
- /Users/matthewsmith/Desktop/Experiments/2016-06-29
- /Users/matthewsmith/Desktop/Experiments/2016-06-29/oe
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers/D1
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers/D1/cnt
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers/D1/htp
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers/D2
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers/D2/cnt
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers/D2/htp
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers/exp1
- /Users/matthewsmith/Desktop/Experiments/20161101\_htp\_pers/exp2
- /Users/matthewsmith/Desktop/Experiments/20161114\_persist
- /Users/matthewsmith/Desktop/Experiments/20161114\_persist/cnt1
- /Users/matthewsmith/Desktop/Experiments/20161114\_persist/cnt2

Buttons: Add Folder..., Add with Subfolders..., Move to Top, Move Up, Move Down, Move to Bottom, Remove, Save, Close, Revert, Default

outputTempCels =

-11.1111

fx >>

Search Documentation

2)\*(5/9)

^



# 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

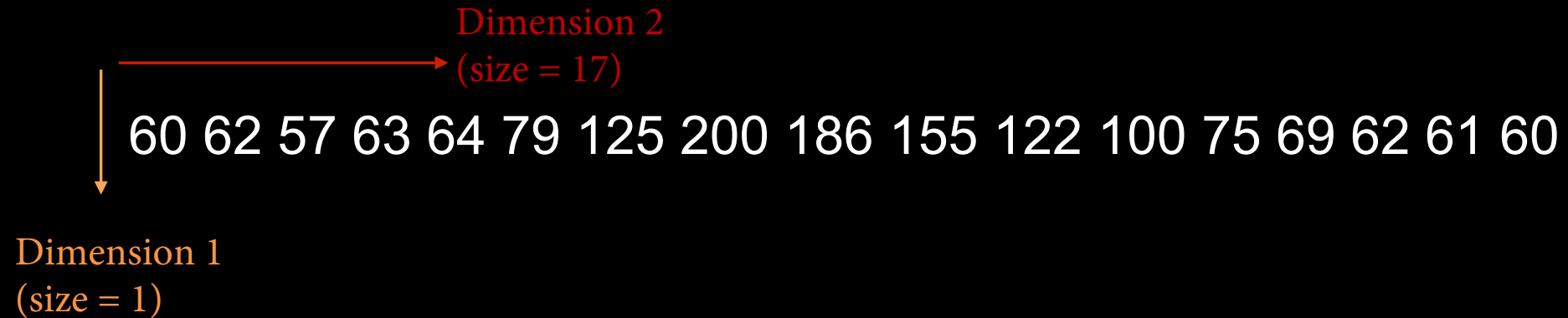
The basic unit for representing information and data is the array

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



# 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 ]
```

**brackets -> [ ]**

# How to create arrays in MATLAB

Arrays are created in MATLAB by using double brackets:

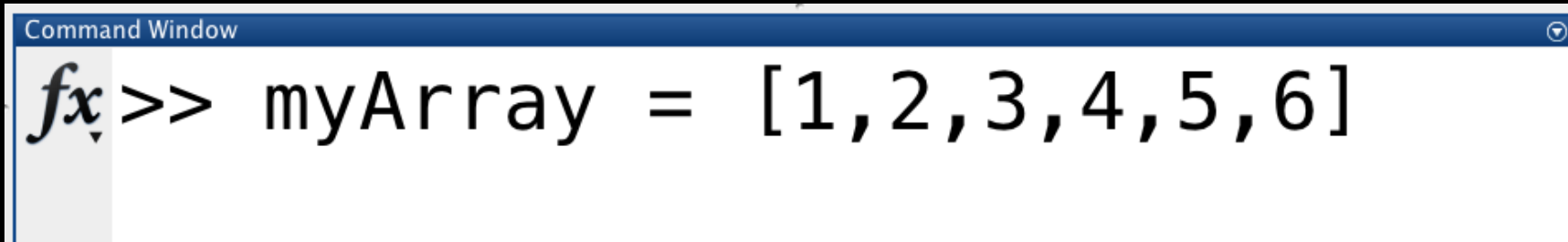
```
myFirstArray = [ 1,2,3,4,5,6 ]
```



# How to create arrays in MATLAB

Arrays are created in MATLAB by using double brackets:

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

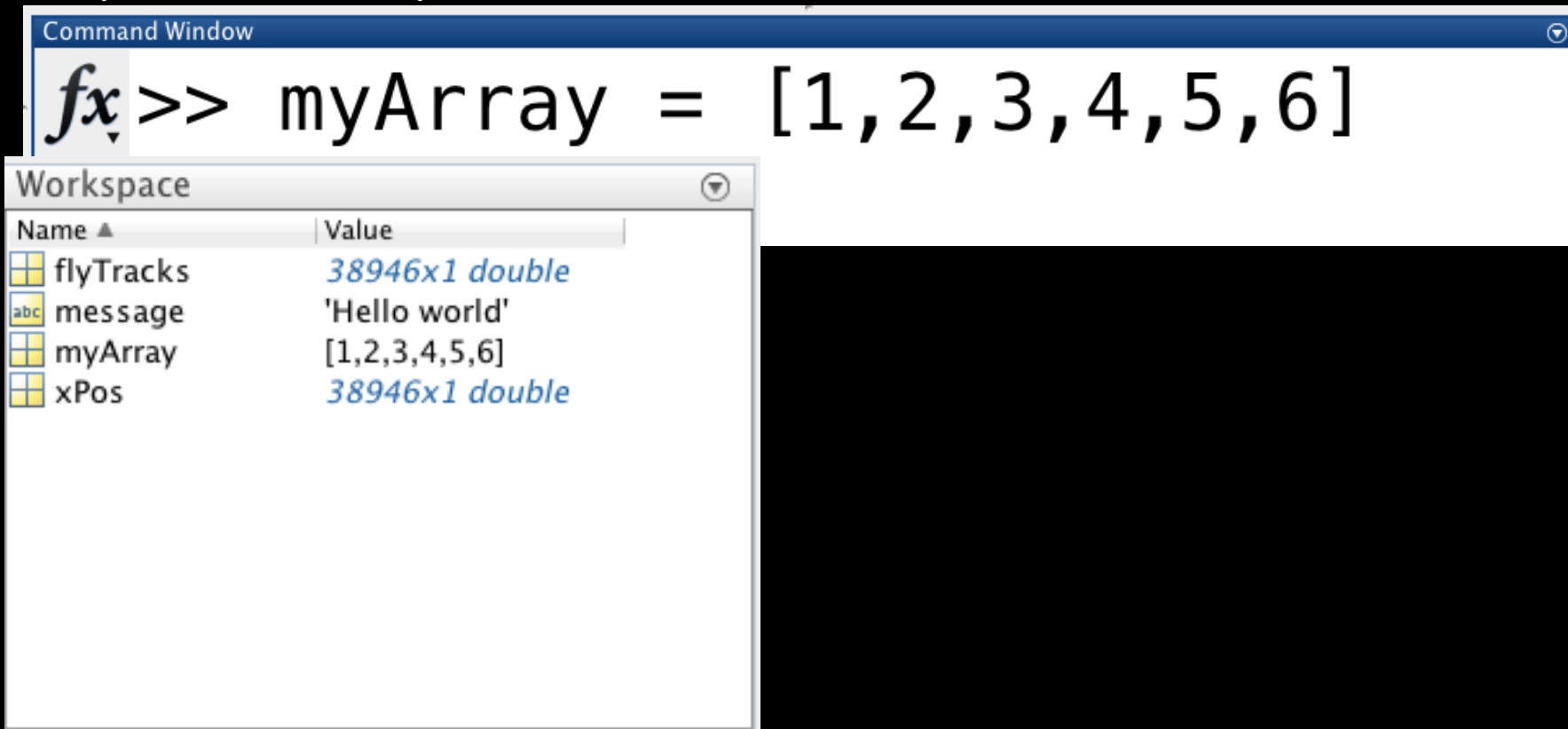
A screenshot of the MATLAB Command Window. The window title is "Command Window". The prompt is "fx" followed by the command "myArray = [1,2,3,4,5,6]".

```
Command Window  
fx >> 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 ]`



The screenshot shows the MATLAB Command Window and Workspace. The Command Window displays the command `myArray = [1,2,3,4,5,6]` being entered. The Workspace window shows the variable `myArray` with the value `[1,2,3,4,5,6]`.

Name	Value
flyTracks	38946x1 double
message	'Hello world'
myArray	[1,2,3,4,5,6]
xPos	38946x1 double

# Data storage in MATLAB

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

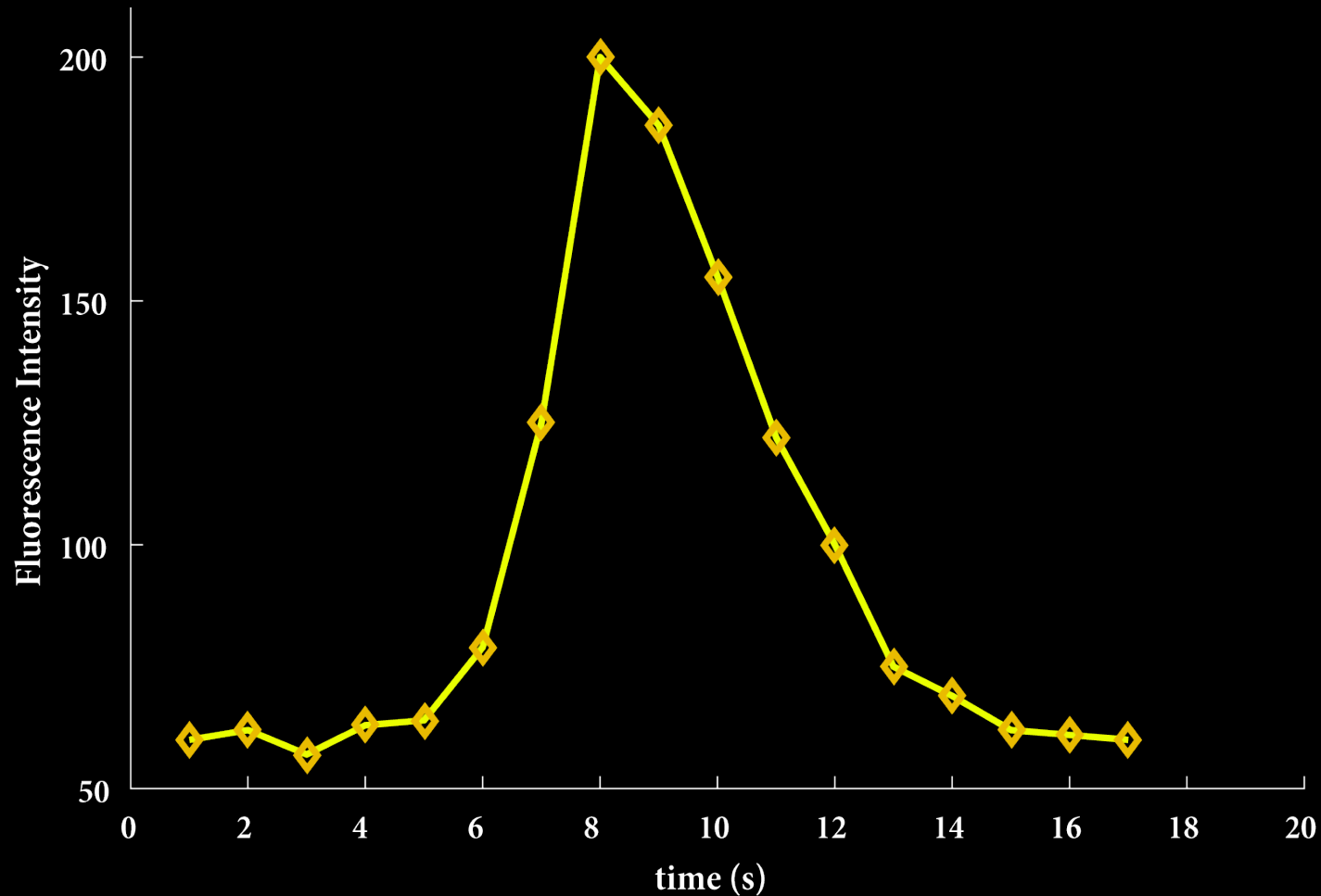
# Data storage in MATLAB

```
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

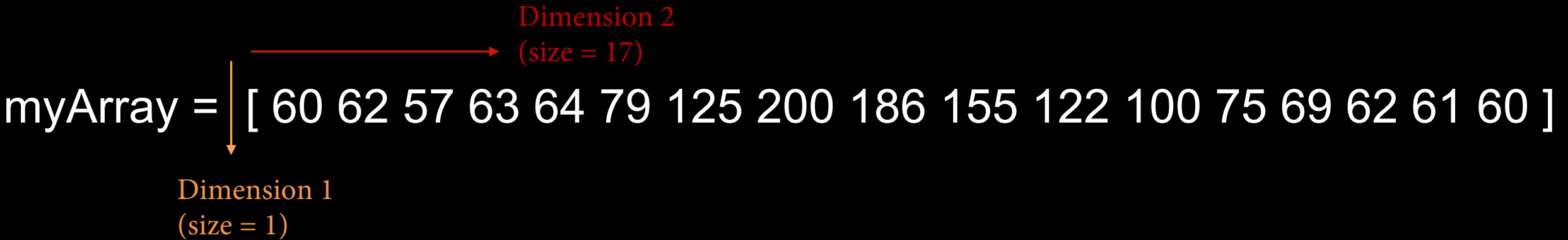
```
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]
```

# <3 INDEXING

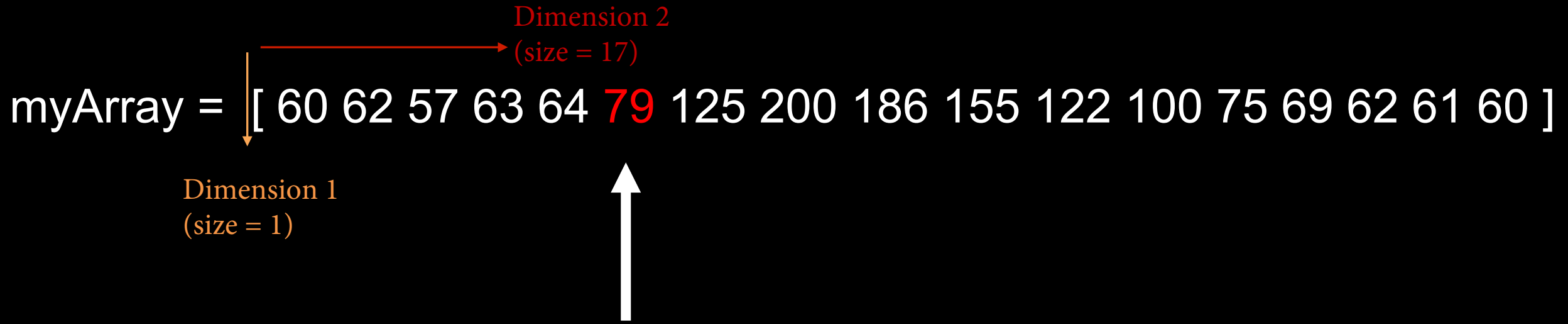
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]

Dimension 1  
(size = 1)

Dimension 2  
(size = 17)



# <3 INDEXING



I want this value. What position is it in?

# <3 INDEXING

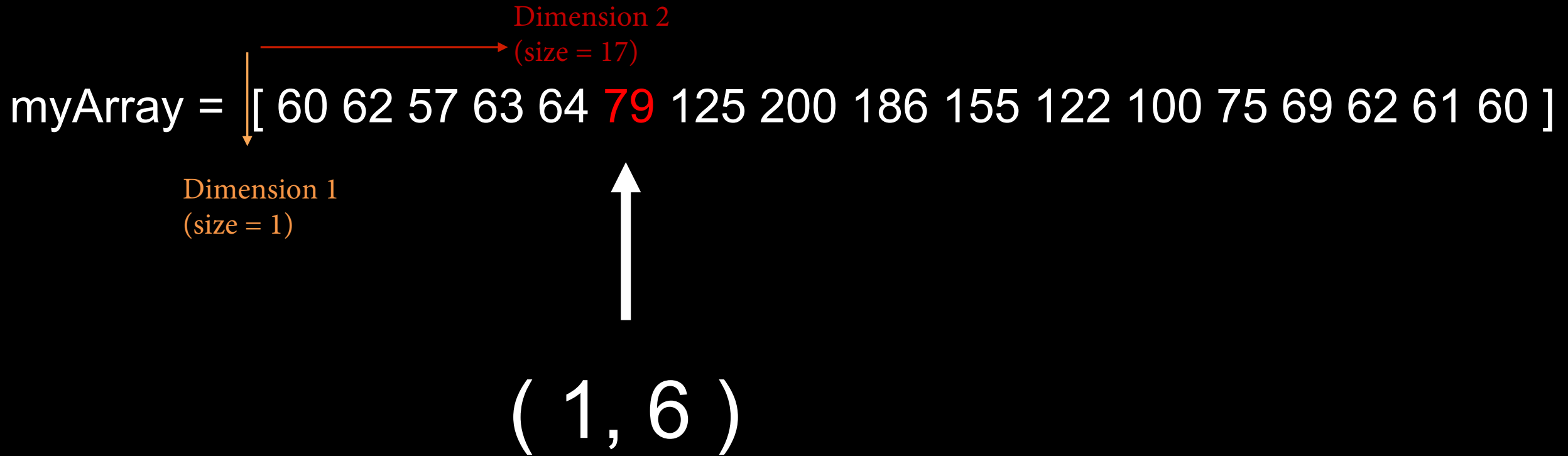
myArray = [ 60 62 57 63 64 79 125 200 186 155 122 100 75 69 62 61 60 ]

Dimension 2  
(size = 17)

Dimension 1  
(size = 1)

( row, column )

# <3 INDEXING



# <3 INDEXING

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)

myArray( 1, 6 )

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:

```
myArray3 = 1 : 1 : 100
```

start                          end  
interval

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

Enter the command:

```
myArray3 = 1 : 2 : 100
```

                  start                  end  
                  interval

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

## **Parentheses ( )**

---

Parentheses are used for:

## **Brackets [ ]**

---

Brackets are used to:



# Parentheses versus brackets in MATLAB

## Parentheses ( )

Parentheses are used for:

Indexing into an array	<code>x(1:3)</code>
Defining order of operations	<code>(3+4)^2</code>
Function inputs	<code>mean(x)</code>

## Brackets [ ]

Brackets are used to:

# Parentheses versus brackets in MATLAB

## Parentheses ( )

Parentheses are used for:

Indexing into an array	<code>x(1:3)</code>
Defining order of operations	<code>(3+4)^2</code>
Function inputs	<code>mean(x)</code>

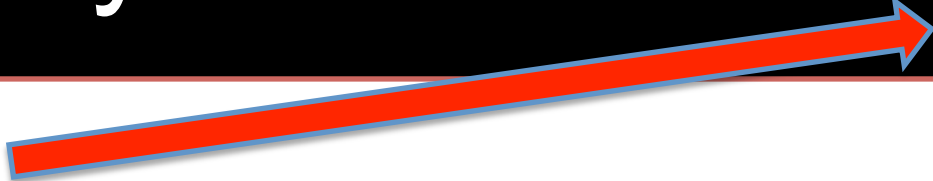
## Brackets [ ]

Brackets are used to:

Create an array or matrix	<code>x = [1 2; 3 4]</code>
Delete (excise) elements	<code>x(x &lt; 0) = []</code>
Group function outputs	<code>[value index] = max(x)</code>

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

;

myMatrix =

[100,101,102,103;104,105,106,107;108,109,110,111]

Command Window

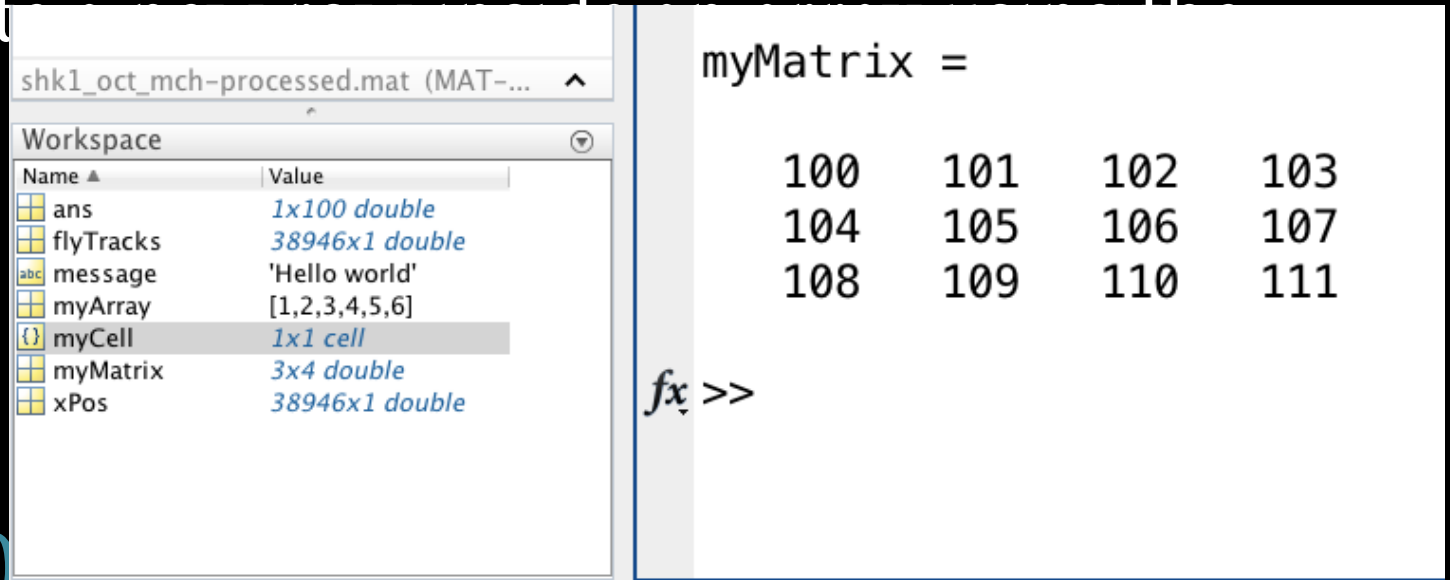
```
fx >> myMatrix = [100,101,102,103;104,105,106,107;108,109,110,111]
```

# How to create matrices in MATLAB

In MATLAB you can create a matrix using a semi-colon

myMatrix =

[100,101,102,103;104,105,106,107;108,109,110,111]



The screenshot shows the MATLAB workspace and command window. The workspace window displays the following variables:

Name	Value
ans	1x100 double
flyTracks	38946x1 double
message	'Hello world'
myArray	[1,2,3,4,5,6]
myCell	1x1 cell
myMatrix	3x4 double
xPos	38946x1 double

The command window shows the following code and output:

```
fx >> myMatrix =  
    100    101    102    103  
    104    105    106    107  
    108    109    110    111
```

Command Window

```
fx >> myMatrix = [100,101,102,103;104,105,106,107;108,109,110,111]
```

Enter the command:

```
figure;surf(myMatrix)
```

# Cell-Arrays

# Cell-Arrays

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

Tiny image

0.8147	0.2785	0.9572
0.9058	0.5469	0.4854
0.1270	0.9575	0.8003
0.9134	0.9649	0.1419
0.6324	0.1576	0.4218
0.0975	0.9706	0.9157

Metadata on tiny image

0.7922	0.6557	0.8491	0.6787	0.7431	0.6555	0.7060
0.9595	0.0357	0.9340	0.7577	0.3922	0.1712	0.0318

# Cell-Arrays

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

0.8147	0.2785	0.9572
0.9058	0.5469	0.4854
0.1270	0.9575	0.8003
0.9134	0.9649	0.1419
0.6324	0.1576	0.4218
0.0975	0.9706	0.9157

0.7922	0.6557	0.8491	0.6787	0.7431	0.6555	0.7060
0.9595	0.0357	0.9340	0.7577	0.3922	0.1712	0.0318

**BUT HOW?**

# Cell-Arrays

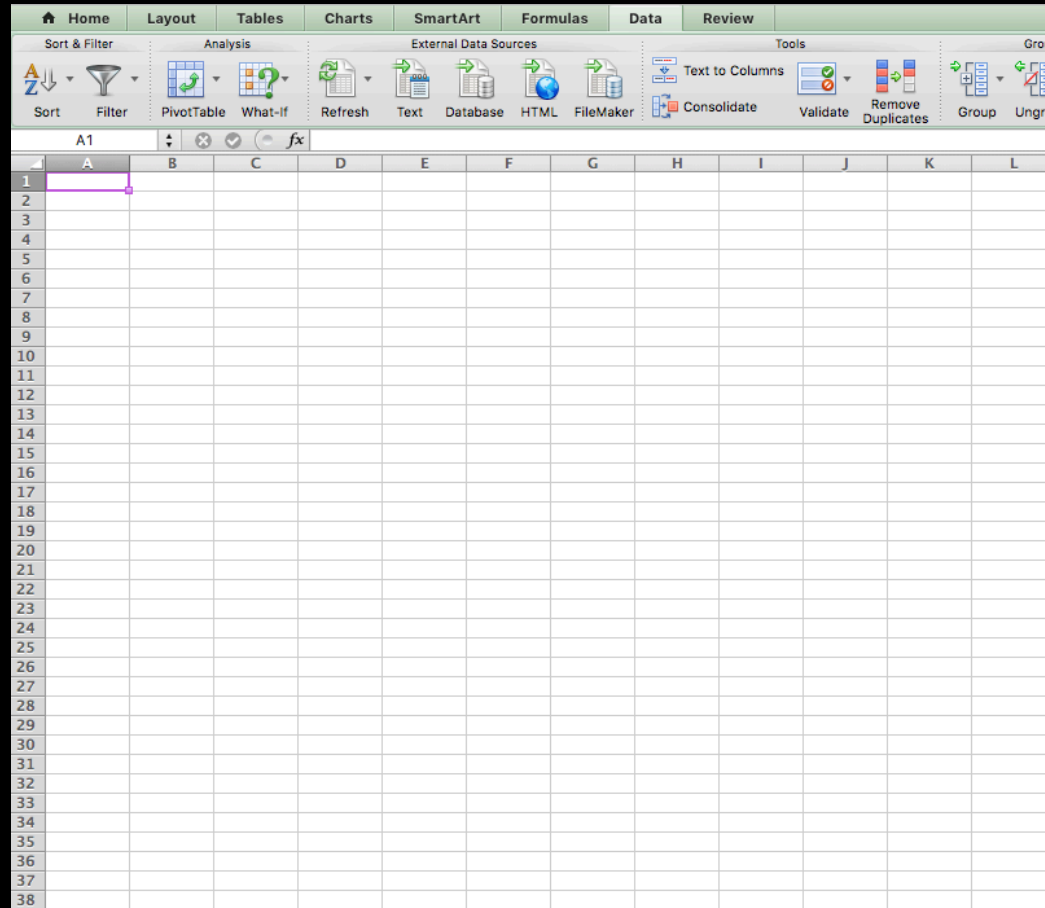
These are data types with indexed containers called “cells”. The user can instruct MATLAB to store large amounts of data in a specific cell.

# 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.

Curly braces



```
Command Window  
fx>> 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.

Curly braces



```
fx >> myCell{1} = myMa
```

Name ▲	Value
ans	1x100 double
flyTracks	38946x1 double
message	'Hello world'
myArray	[1,2,3,4,5,6]
myCell	1x1 cell
myMatrix	3x4 double
xPos	38946x1 double

# 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:

Logical operator:		Meaning:
<	-----	is less than
>	-----	is greater than
>=	-----	is greater than or equal to
<=	-----	is less than or equal to
==	-----	is equal to
~=	-----	not equal to
&	-----	and
	-----	or

## 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?

```
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?

```
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?

```
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?

```
if freeFood == true
```

```
    MattAttend = true
```

```
elseif interestingSpeaker == true | Georgia_Attend == true
```

```
    MattAttend = true
```

```
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?

```
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:

```
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.')
else
        disp('There is no significant change.')
end
```

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?

# 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.

# 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]



# 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

# 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
```

# 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
```

My temp conversion algorithm:

```
inputTempFah = 12
```

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

Write MATLAB code to iterate your temp conversion algorithm over the five numerical values:

12, 6, 130, 273, -34

# 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= [series of numbers]
```

```
    MATLAB commands you want to execute and repeat
```

```
end
```

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:

```
for inputTempFah= [12, 6 130, 273, -34]
```

MATLAB commands you want to execute and repeat

```
End
```

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:

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

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:

```
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)