

Matlab Fundamentals

Programming Lecture 1

Nicholas Dwork

1

Text Files

**Computer code is written in text files.
A text file often has the extension *.txt.**

**Computer code text files often have other extensions:
These extensions indicate the language**

- *.m - matlab file**
- *.c - c file**
- *.cpp - c plus plus file**
- *.js - javascript**
- *.html = webpage**

Note: *.doc is not a text file

2

Comments

A comment starts with the % symbol.

A comment does absolutely nothing. It's just a note from the programmer

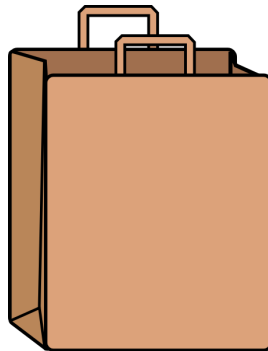
Ex:

```
% Hi fellow coder, this is a comment
```

3

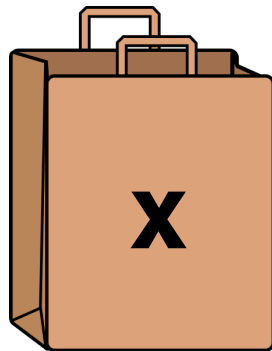
Variables

A variable is a container that you can hold values.



4

Variables have names. You can put a value in the container with the equal sign.



x = 8;

The thing on the left gets the value on the right.

5

Types of Variables

Numbers: x = 8;

Arrays: x = [2,3,5,7,9,11,13,17];

Characters: x = 'a';
x = 'x';

Strings: x = 'This is a string';
Note: a string is an array of characters;

6

We can then use that variable in expressions

```
x = 8;  
y = x + 2;
```

In the above example, y gets the value 10.

7

Arrays

Arrays: `x = [2,3,5,7,9,11,13,17];`

**An array is an ordered list of numbers.
You can get individual elements by using ().**

The value of `x(1)` is 2.

The value of `x(2)` is 3.

The value of `x(6)` is 11.

`x = 1:8;` **Makes the array [1,2,3,4,5,6,7,8]**

8

Making Arrays

`x = [2,3,5,7,9,11,13,17];` **Make the array explicitly.**

`x = 1:8;` **Makes the array [1,2,3,4,5,6,7,8]**

`x = zeros(10,1);` **Makes an array of 10 elements,
all values are zero.**

`x = ones(10,1);` **Makes an array of 10 elements,
all values are one.**

9

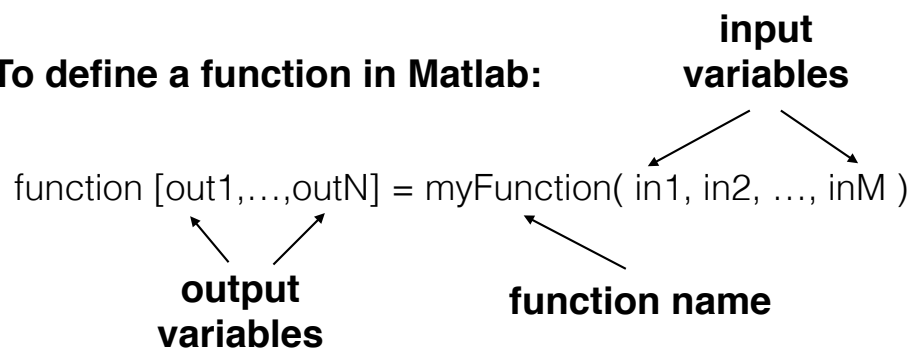
Functions

A function is a programming machine

You input stuff

You get something out

To define a function in Matlab:



10

Function Files

Each function you create will be a separate file.

11

Ex: Printing a phrase to the screen:

```
disp('Hello World!');
```

The name of the function is disp

Meant to be short for display

The input to the function is a string: 'Hello World'

This function has no output

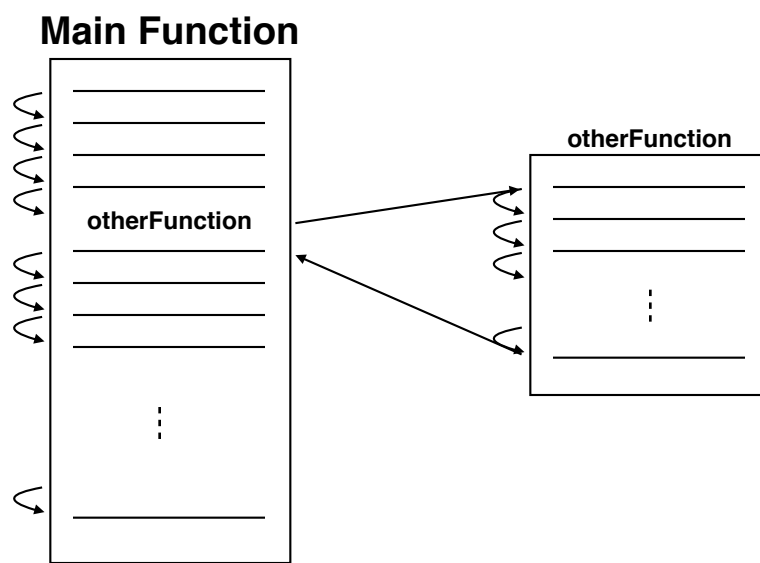
12

Calling a Function

```
x = 8;  
y = x + 2;  
disp(y);
```

13

Flow



14

Scope

A variable only exists inside the function where it is created.

```
function main()  
    x = 10;  
    y = 20;  
    z = myFunc( x, y );  
    disp(z);  
end
```

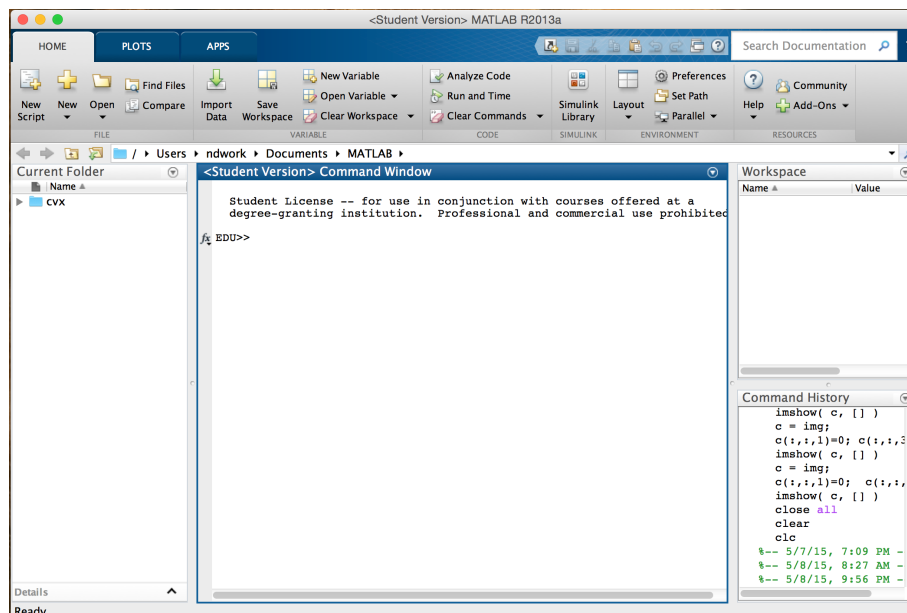
```
function out = myFunc( a, b)  
    disp(x);  
    out = a + b;  
end
```

The value displayed is 30.

The slashed line causes an error; x does not exist there.

15

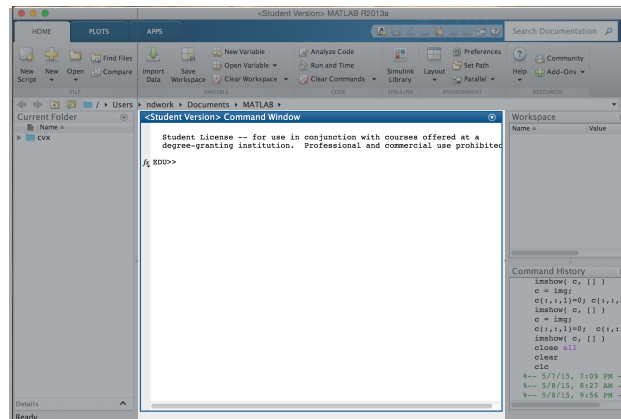
Matlab Programming Environment



16

Command Window

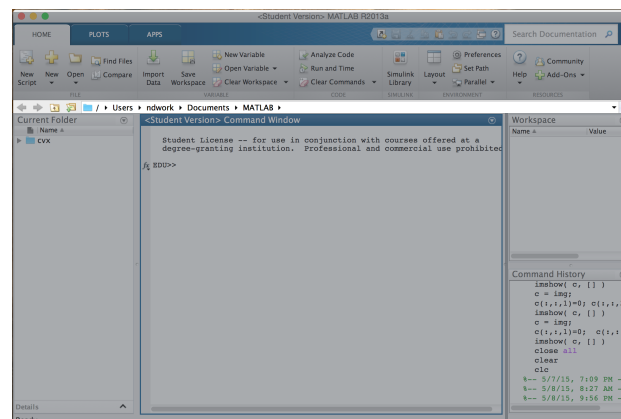
Can run single lines of code (including your own functions) into the command window.
Things that are displayed get displayed here.



17

Current Working Directory

You can use any functions in files that are located in this directory.

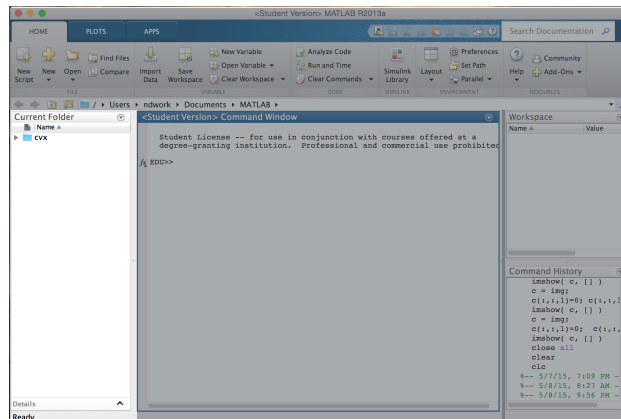


Make a new directory for any program you create.

18

Navigator Window

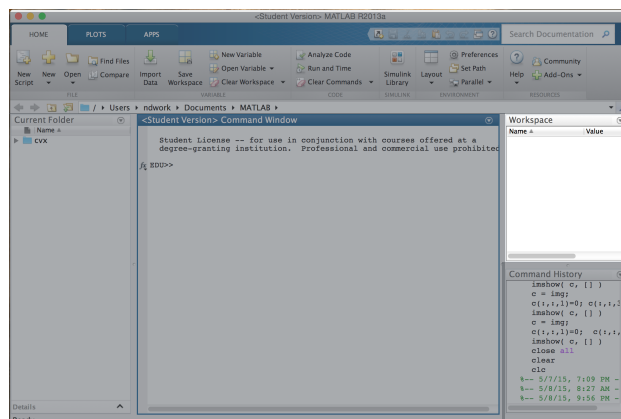
Shows files and directories.



19

Workspace

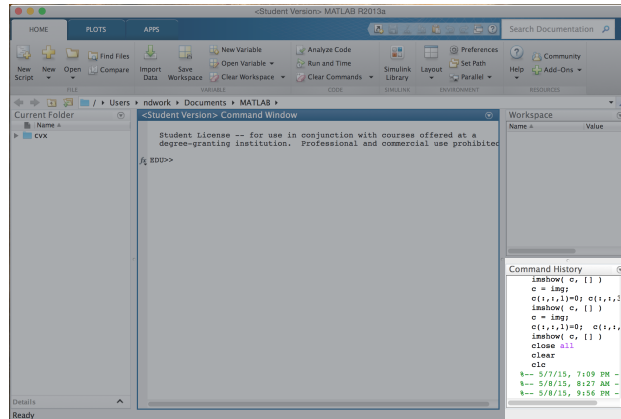
Shows variables that currently exist and their values.



20

Command History

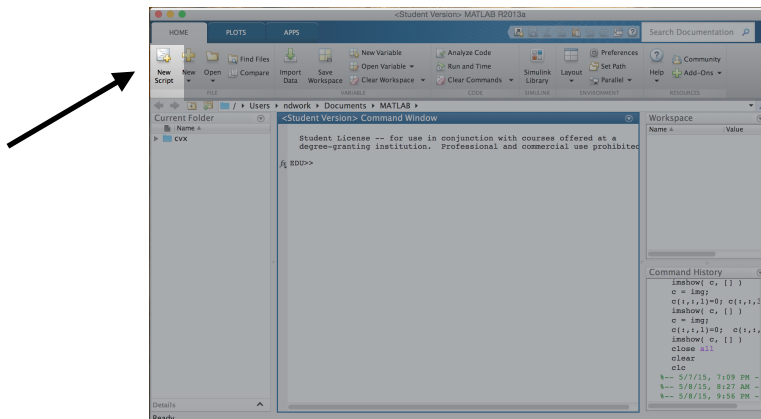
Shows recent commands that you've written into the command window.



21

New Function File

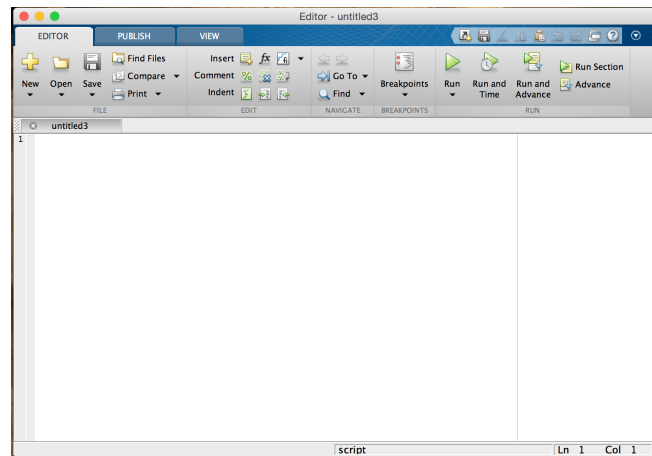
To create a file for a new function, hit the “New Script” button.



22

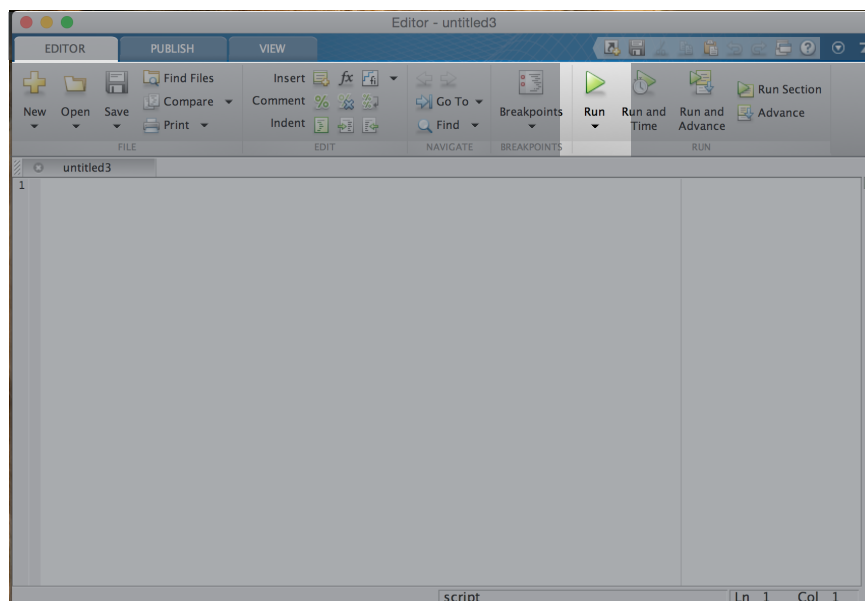
The Editor Window

**You can write new functions using the editor window.
You also debug in the editor window (discussed later).**



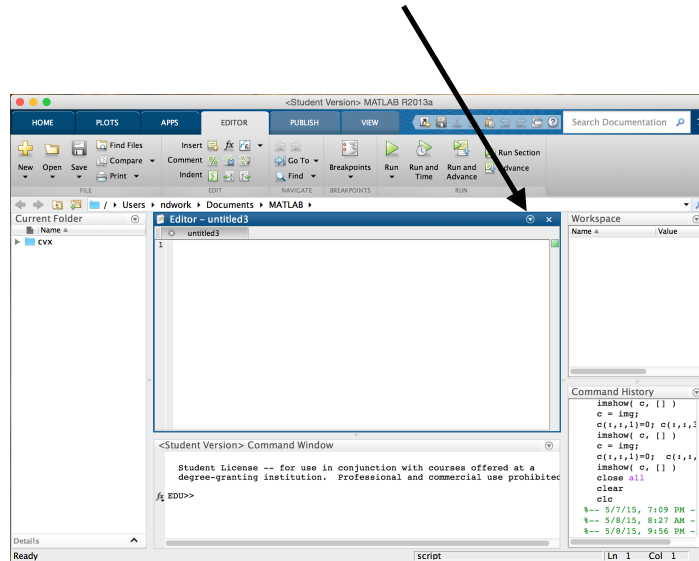
23

Hitting “Run” runs the current file shown.



24

The editor window can be combined with the Development Environment. Select “Dock” from this drop down menu.



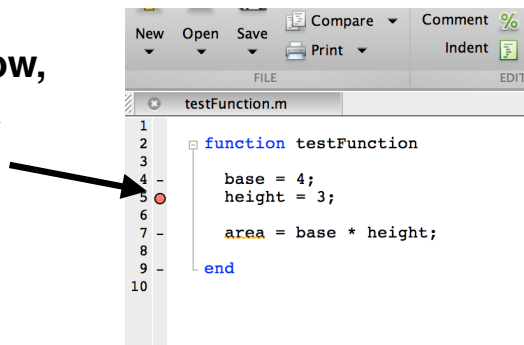
25

The Debugger

Allows the coder to stop the code in the middle and investigate.

In the editor window, click here to add a “break point”

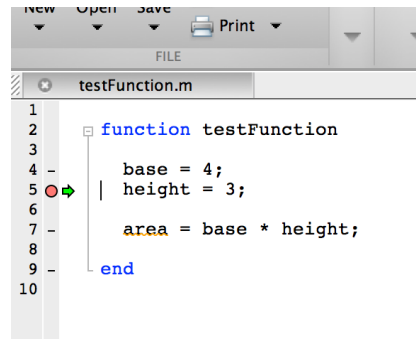
When you hit run, the program will stop at the break point.



26

The arrow indicates the next line to be executed.

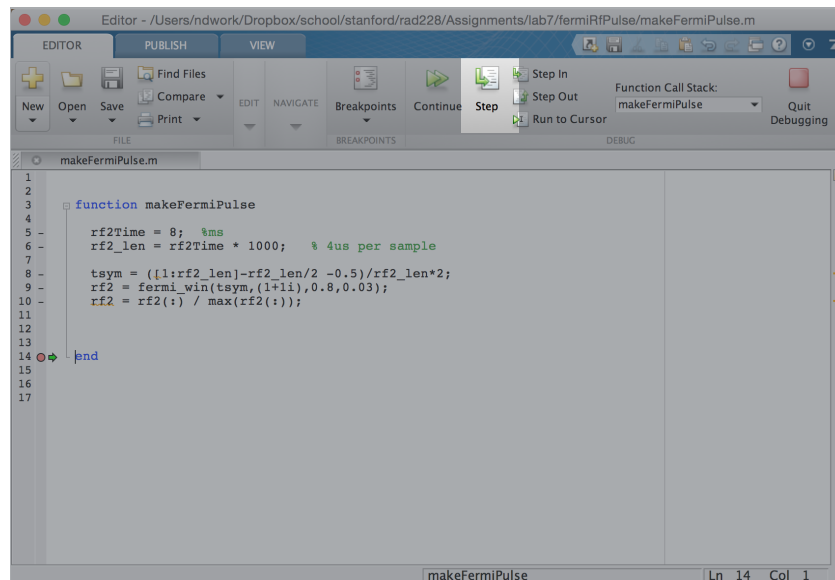
Can print variable values in the command window.



```
1  
2  
3  
4 -   base = 4;  
5 -   height = 3;  
6  
7 -   area = base * height;  
8  
9 -   end  
10
```

27

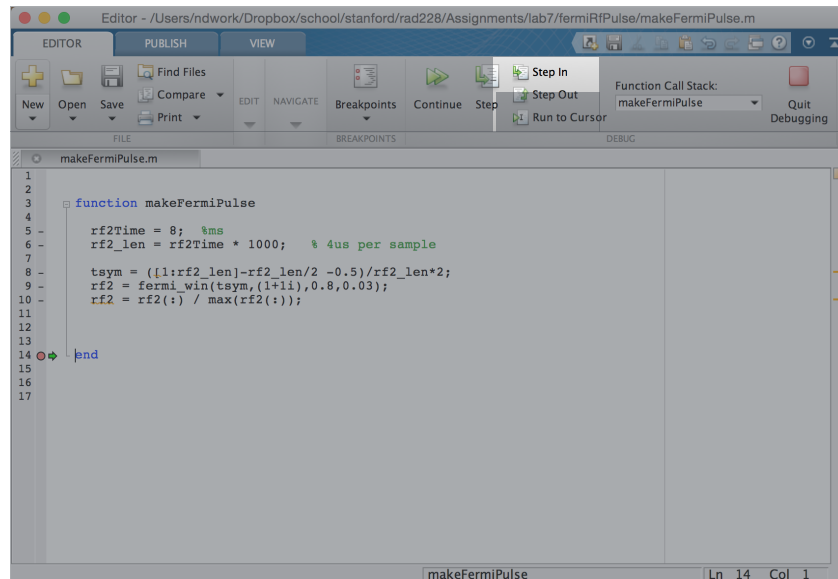
Hit “Step” to advance one line.



```
1  
2  
3  
4  
5 -   rf2Time = 8; %ms  
6 -   rf2_len = rf2Time * 1000; % 4us per sample  
7  
8 -   tsym = ([1:rf2_len]-rf2_len/2 -0.5)/rf2_len*2;  
9 -   rf2 = fermi_win(tsym,(1+1i),0.8,0.03);  
10 -   rf2 = rf2(i) / max(rf2(i));  
11  
12  
13  
14 -   end  
15  
16  
17
```

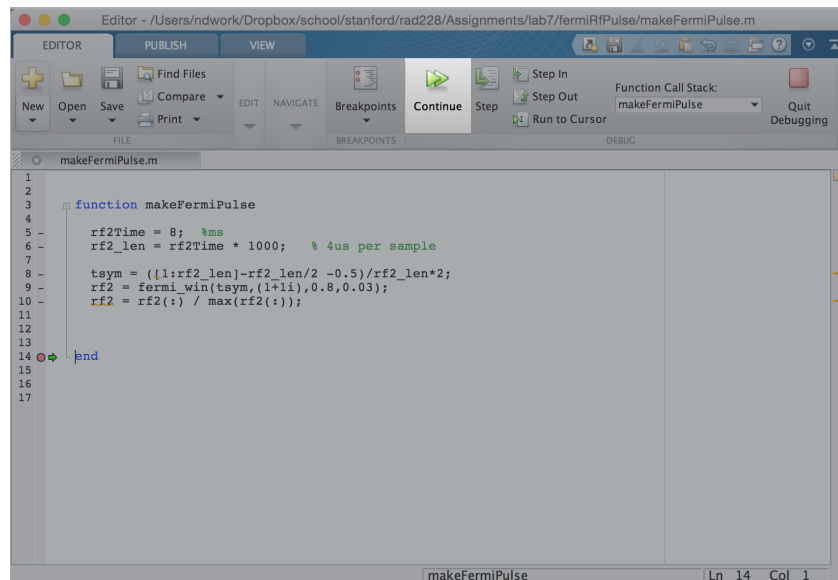
28

Hit “Step Into” if the next line is a function, and you would like to go through the function.



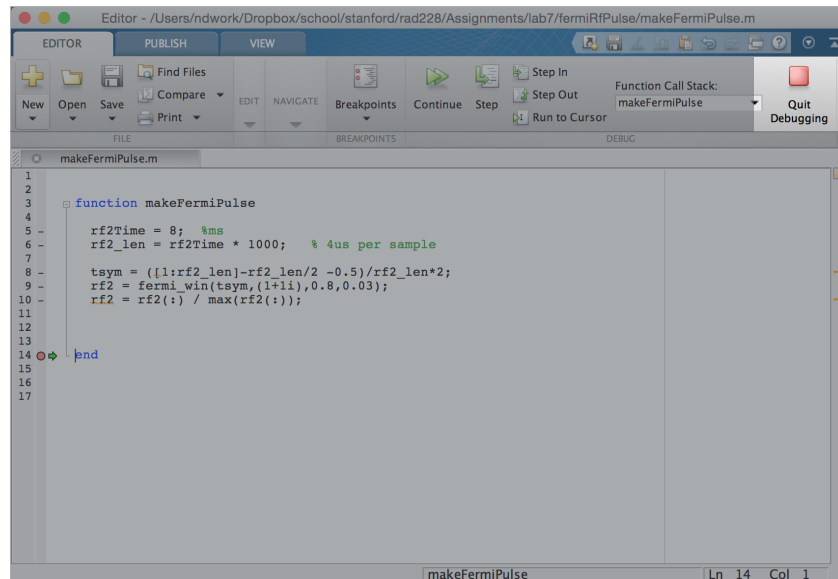
29

Hit “Continue” to run until the next breakpoint.
If there are no more breakpoints, it will run until the program completes.



30

Hit “Quit” to exit the debugger.



31

Getting Help

If you know the function name:

Type `help function_name` **on command line**
Simple text help

Type `doc function_name` **on command line**
More complete help document

Type `matlab function_name` **into Google**
All of Matlab's help documents are online

32

Getting Help

If you don't know the function name:

Use a web search engine (e.g. Google).

There's a giant vibrant community of Matlab users helping each other out through the web.

33

Comparison Operators

Return 1 if true, and 0 otherwise

== Tests to see if two expressions are equal
Ex: a == b

~= Tests to see if two expressions are not equal
Ex: a ~= b

34

Return 1 if true, and 0 otherwise

> Tests to see if the thing on the left is greater than the thing on the right.

Ex: $a > b$

>= Tests to see if the thing on the left is greater than or equal to the thing on the right.

Ex: $a \geq b$

35

Matrices

$x = [1, 2, 3; 4, 5, 6; 7, 8, 9];$

x becomes the following matrix:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Matrices are two dimensional arrays

36

Accessing Values

```
x = [ 1, 2, 3; 4, 5, 6; 7, 8, 9];
```

```
x21 = x(2,1);  
disp(x21);           % displays 5 to the screen
```

```
x2 = x(2,:);  
disp(x2);           % displays [4;5;6] to the screen
```

37

Manipulating Matrices

```
x = [ 1, 2, 3; 4, 5, 6; 7, 8, 9];  
y = [ 10, 0, 0; 0, 10, 0; 0, 0, 10];
```

```
addResult = x + y;  
subtractResult = x - y;  
multiplyResult = x * y;  
pointwiseMultiplication = x .* y;
```

38

Matrix Inversion

$$Ax = b$$

How do we solve for x? $x = A^{-1}b$

In Matlab: `x = A \ b;`

39

If ... Then ... Else

```
if a==1
    disp('a is equal to 1');
else
    disp('a is not equal to 1');
end
```

Only branch satisfying condition is executed.

40

For Loops

```
for i = first : last  
    % do something here  
end
```

For each value of i, the code inside the loop gets executed.

41

```
x = [ 1, 2, 3; 4, 5, 6; 7, 8, 9];
```

```
for i=1:3  
    disp(x(i,:))  
end
```

This code displays [1;2;3] then [4;5;6] then [7;8;9].

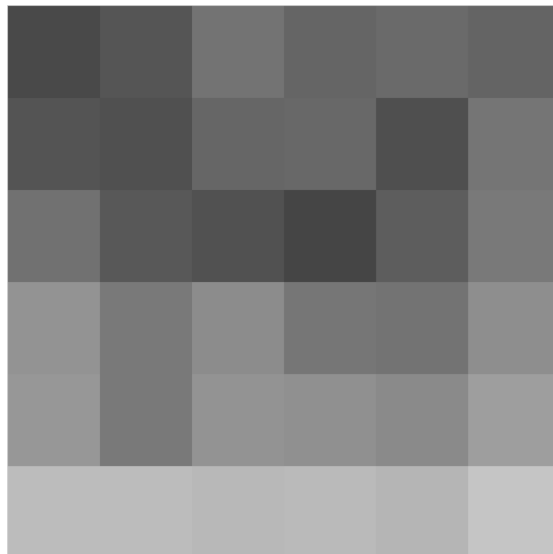
42

2D Array

57	67	96	82	87	81
66	63	83	85	62	98
94	70	64	53	75	102
129	102	121	99	96	123
133	102	129	125	119	140
174	174	170	172	166	184

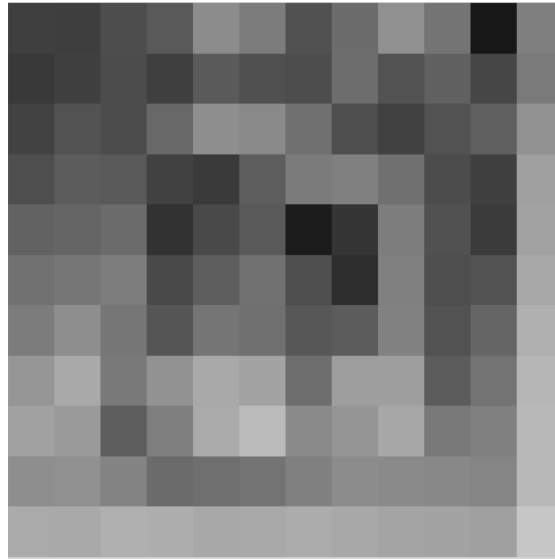
43

Rather than showing the numbers, we can show corresponding colors. 0=black, and 255=white.



44

**Here's a larger array.
0=black, and 255=white.**



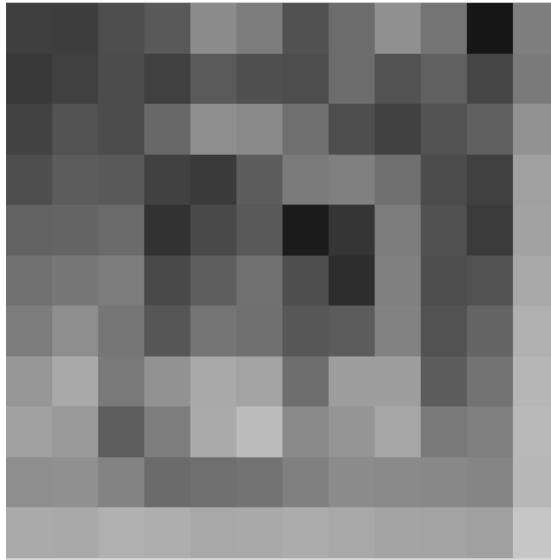
45

We can always go back to the array of numbers.

48	47	60	71	121	105	64	89	125	97	18	108
43	49	59	49	72	63	60	90	65	78	54	103
51	65	59	85	123	118	92	61	50	65	77	128
61	74	71	50	44	75	104	109	93	59	49	143
79	82	88	38	57	71	21	40	106	64	45	145
94	100	106	57	76	94	61	34	109	61	65	152
105	123	99	68	98	93	69	74	111	65	82	160
132	153	102	128	153	146	91	141	141	74	96	167
144	136	76	107	154	173	119	131	150	102	109	171
123	125	112	88	92	96	109	121	119	117	114	170
154	153	160	158	152	153	156	152	147	146	143	186

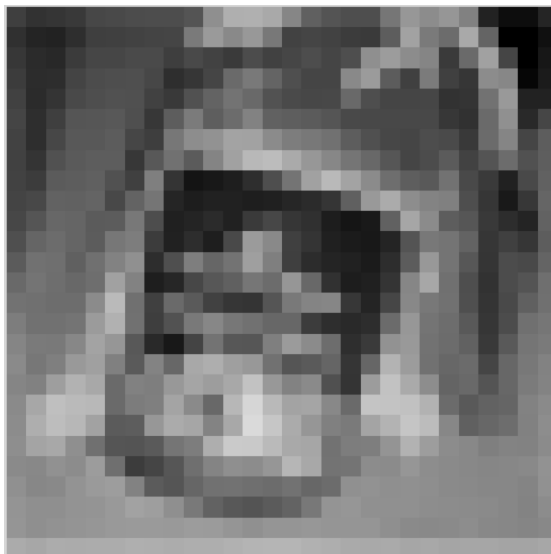
46

**Here's a larger array.
0=black, and 255=white.**



47

**Here's an even larger array. Now we have too many
numbers to display on this screen.
0=black, and 255=white.**



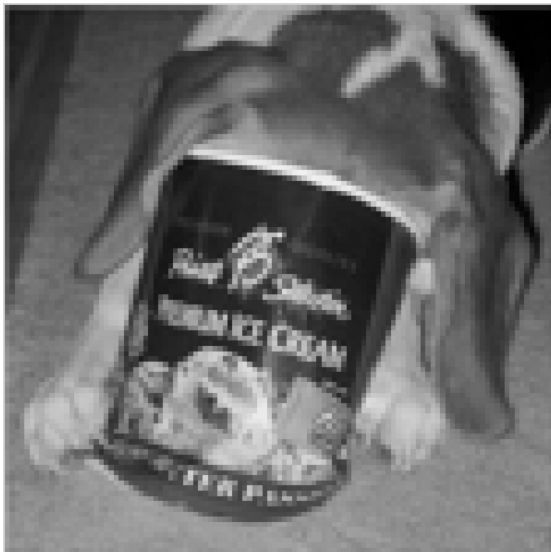
48

**And larger ...
0=black, and 255=white.**



49

**And larger
0=black, and 255=white.**



50

**Still larger. At this point, our eye can no longer discern most of the individual pixels.
0=black, and 255=white.**



51

**Largest.
0=black, and 255=white.**



52

Images

Big Conclusion: Images are just 2D arrays that are displayed in an interesting way!

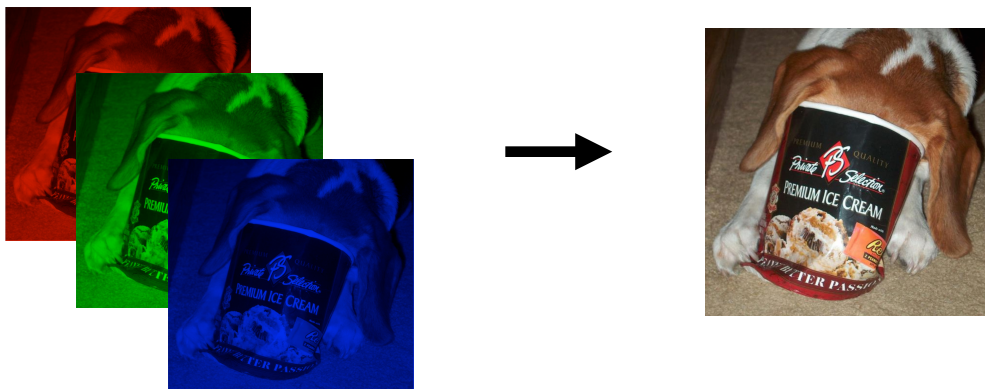
At some point, your eye can no longer distinguish the individual pixels.

In Matlab, images and matrices are exactly the same.

53

Color Image

**A color image is three different arrays.
The computer displays one of the arrays for red,
one for green, and one for blue.**



54

Image Files

***.jpg, *.png, *.gif, *.bmp are all types of image files**

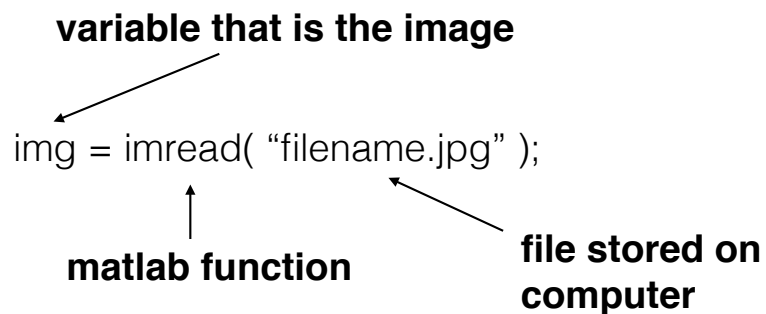
We'll discuss the differences between these file types later in the class.

55

Loading an Image

To load an image into data:

variable that is the image



```
img = imread( "filename.jpg" );
```

matlab function

file stored on computer

56

Color Image Components

<code>img(:, :, 1)</code>	% the red 2D array
<code>img(:, :, 2)</code>	% the green 2D array
<code>img(:, :, 3)</code>	% the blue 2D array

57

Making a Gray Image

The function `rgb2gray` converts a color image (3D array) into a grayscale image (2D array).

```
grayImage = rgb2gray( colorImage );
```

58

Displaying an Image

The function `imshow` **displays an image on the screen.**

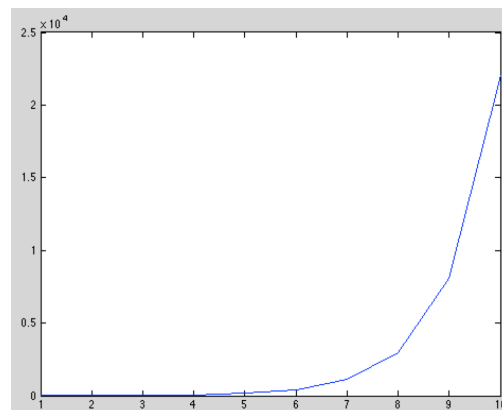
```
imshow( myImage, [] );
```

59

Plotting

```
x = 1:10;  
y = exp( x );  
plot( x, y );
```

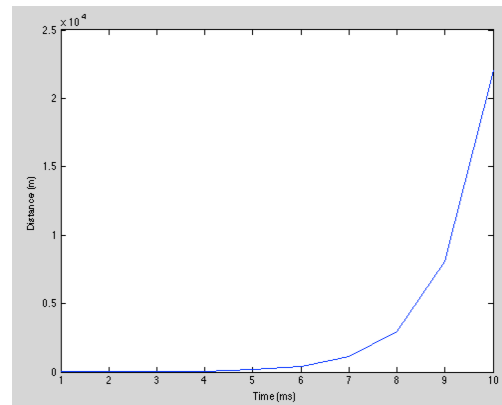
Makes the figure on the right.



60

Adding Labels

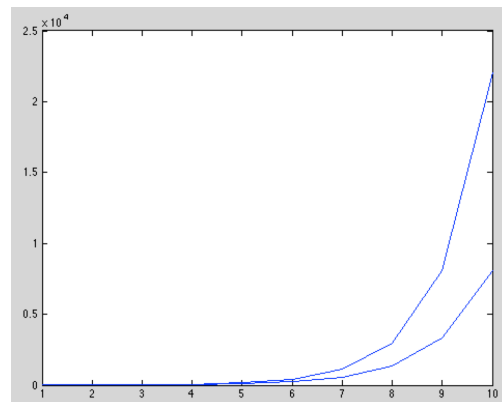
```
x = 1:10;  
y = exp( x );  
plot( x, y );  
xlabel( 'Time (ms)' );  
ylabel( 'Distance (m)' );
```



61

Plotting Multiple Functions

```
x = 1:10;  
y1 = exp( x );  
y2 = exp( 0.9*x );  
plot( x, y1 );  
hold on;  
plot( x, y2 );  
hold off;  
xlabel( 'Time (ms)' );  
ylabel( 'Distance (m)' );
```



62