UNIVERSITY
OF WOLLONGONG
IN DUBAI

Lecture 6 Week 7

Mohamad Nassereddine

Lecture Contents

- Input and Output
- Graphics and Visualization

Importing and Exporting data

- Until now, all operations have been performed on data entered either manually, or generated automatically
- However, most Engineering projects retain their data in files that can be opened, read, edited and saved.
- Common office tools for data manipulation include Microsoft Excel (spreadsheets) and Access (databases), or Adobe Photoshop for images.
  - However, the capability of these programs is largely user-interface driven.
  - MATLAB provides a full programming and development environment.

Accessing files using MATLAB

- MATLAB provides some built-in functions for accessing and manipulating data in files
  - Files can be read or written to
  - Multiple common file formats available
    - eg. `*.mat`, **`*.jpg`, `*.xlsx`, `*.csv`, `*.txt`**
- For files that are not in a standard format, MATLAB also provides low-level File I/O (input/output) functions that allow:
  - Reading a line at a time
  - Specifying what format the data is
  - … and much, much more

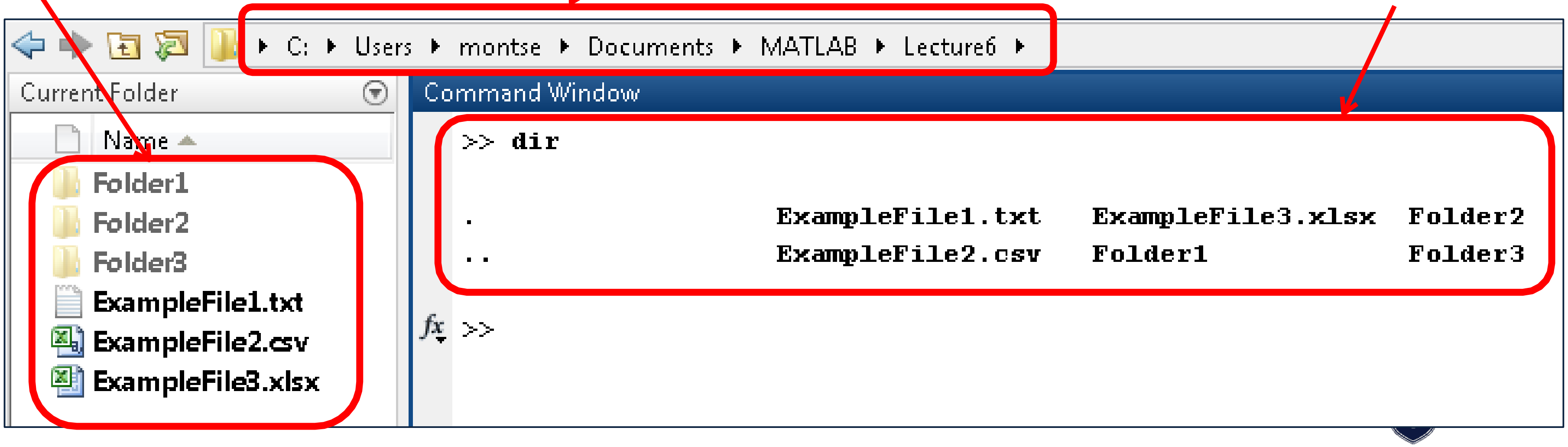
File listings

- When opening files in MATLAB, we require a way to determine
  - which file we want (eg. filename),
  - how to interpret it (eg. known extension format),
  - and where it is located (directory structure information).

Current Folder Location

Current Folder contents

Current Folder contents using command line



C: ▸ Users ▸ montse ▸ Documents ▸ MATLAB ▸ Lecture6 ▸

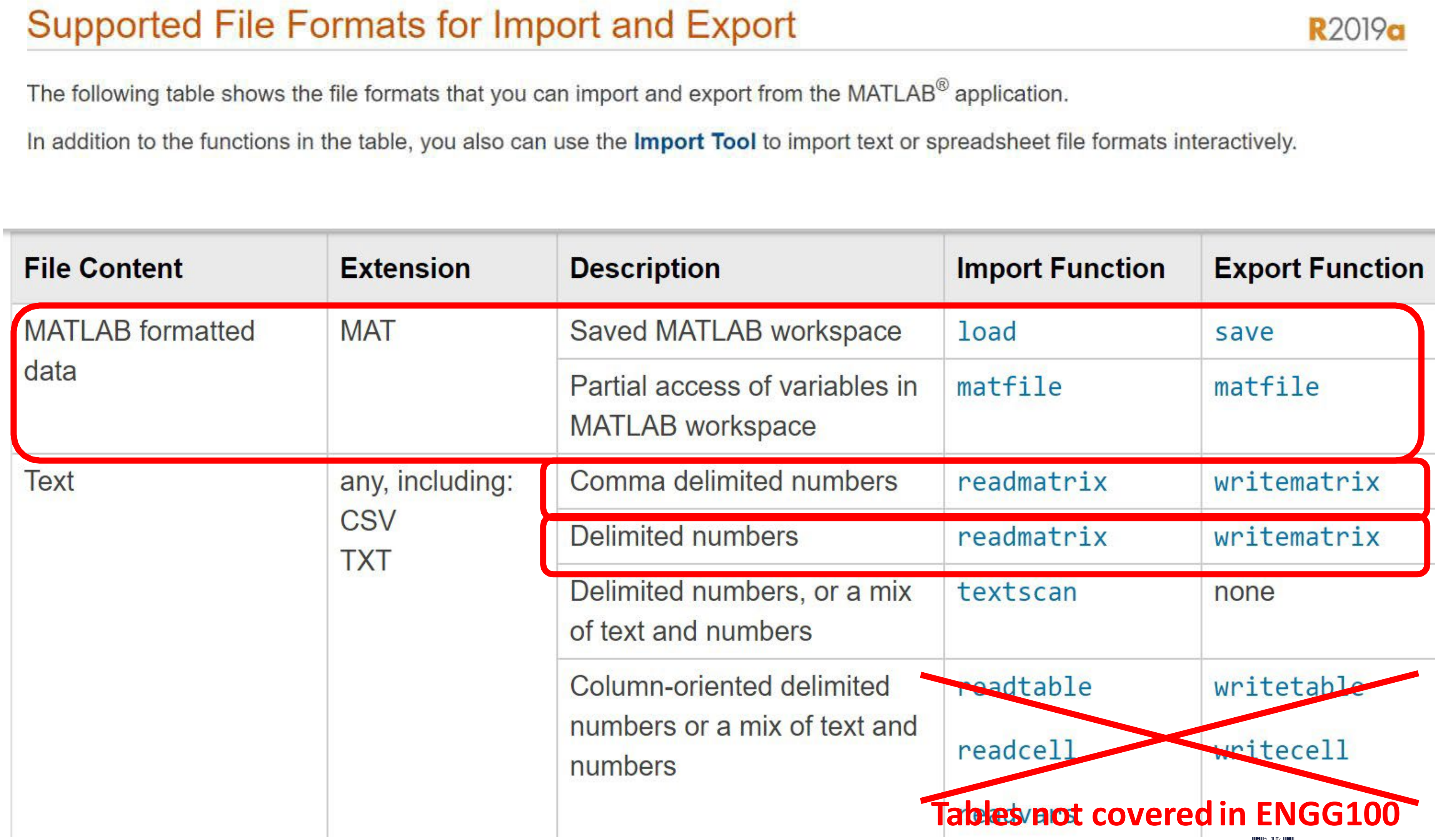
- Folder1
- Folder2
- Folder3
- ExampleFile1.txt
- ExampleFile2.csv
- ExampleFile3.xlsx

```
>> dir

.                 ExampleFile1.txt   ExampleFile3.xlsx   Folder2
..                ExampleFile2.csv   Folder1             Folder3

fx >>
```

File types and uses

- Many different file types exist
  - (*.mat) MATLAB files containing
  - (*.jpg, *.png, …) image files
  - (*.xlsx, *.xls, …) Microsoft Excel spreadsheets
  - (*.txt) Text files
  - (*.csv) Comma-Separated Values files
    - Can be opened in notepad like textfiles
    - Can also be opened in MS Excel as a (limited) spreadsheet

http://au.mathworks.com/help/matlab/import_export/supported-file-formats.html



Supported File Formats for Import and Export

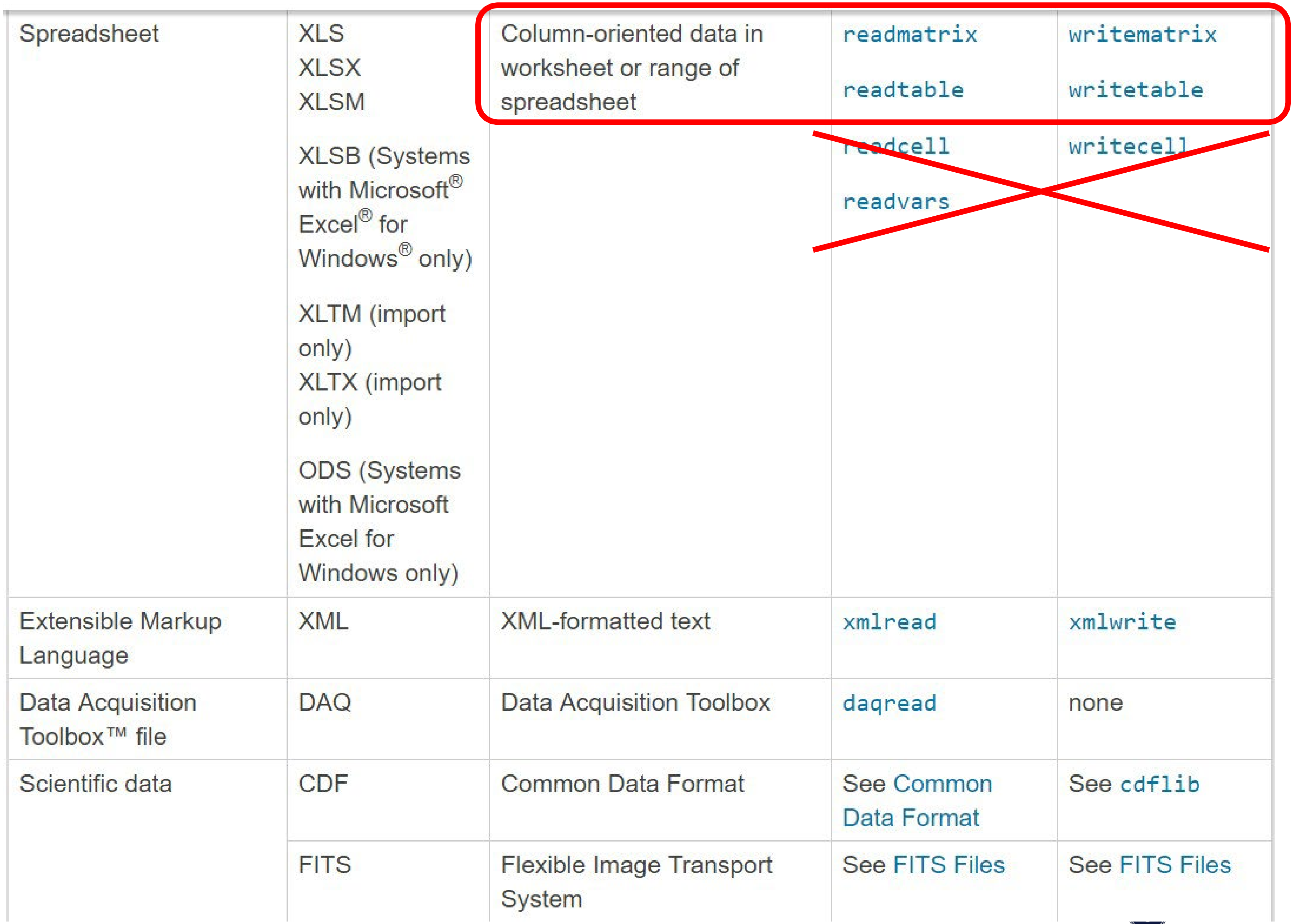
R2019a

The following table shows the file formats that you can import and export from the MATLAB® application.

In addition to the functions in the table, you also can use the **Import Tool** to import text or spreadsheet file formats interactively.

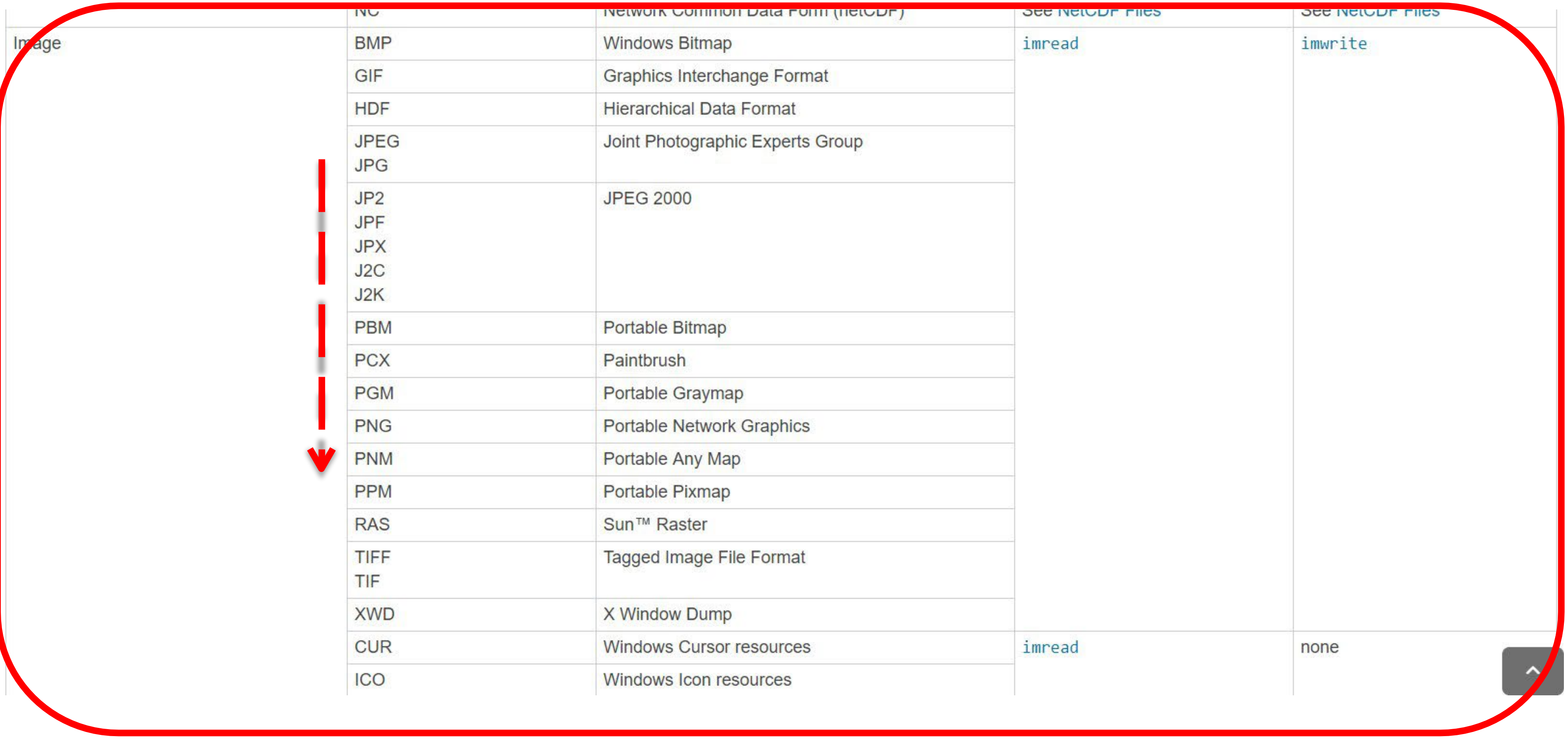
File Content	Extension	Description	Import Function	Export Function
MATLAB formatted data	MAT	Saved MATLAB workspace	load	save
		Partial access of variables in MATLAB workspace	matfile	matfile
Text	any, including: CSV TXT	Comma delimited numbers	readmatrix	writematrix
		Delimited numbers	readmatrix	writematrix
		Delimited numbers, or a mix of text and numbers	textscan	none
		Column-oriented delimited numbers or a mix of text and numbers	~~readtable~~ ~~readcell~~ ~~readvars~~	~~writetable~~ ~~writecell~~

Tables not covered in ENGG100

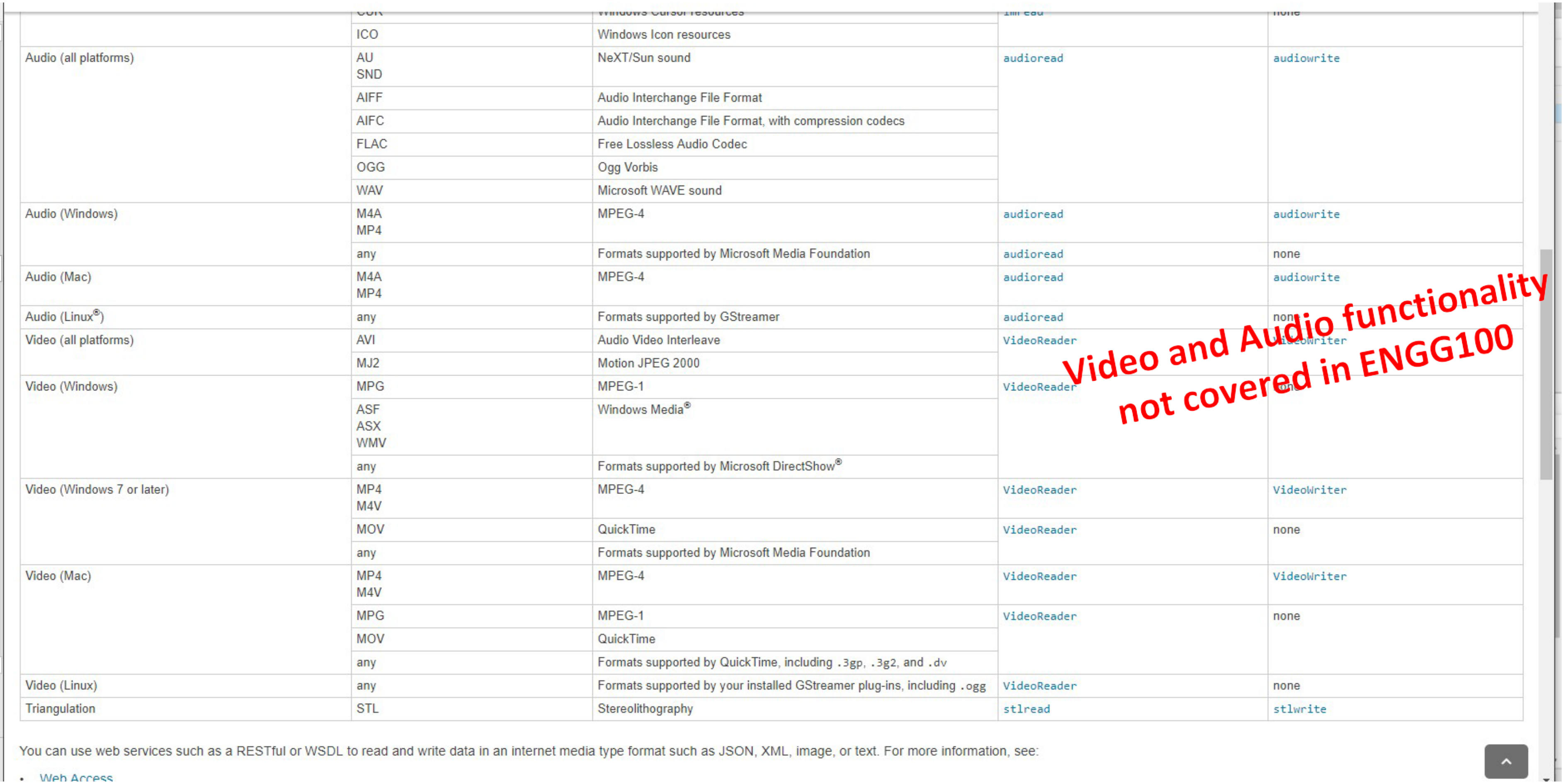


Spreadsheet	XLS
	XLSX
	XLSM
	XLSB (Systems with Microsoft® Excel® for Windows® only)
	XLTM (import only)
	XLTX (import only)
	ODS (Systems with Microsoft Excel for Windows only)
		Column-oriented data in worksheet or range of spreadsheet	`readmatrix`
			`readtable`
			~~`readcell`~~
			~~`readvars`~~
				`writematrix`
				`writetable`
				~~`writecell`~~

Extensible Markup Language	XML	XML-formatted text	`xmlread`	`xmlwrite`
Data Acquisition Toolbox™ file	DAQ	Data Acquisition Toolbox	`daqread`	none
Scientific data	CDF	Common Data Format	See Common Data Format	See `cdflib`
	FITS	Flexible Image Transport System	See FITS Files	See FITS Files



	NC	Network Common Data Form (netCDF)	See NetCDF Files	See NetCDF Files
Image	BMP	Windows Bitmap	imread	imwrite
	GIF	Graphics Interchange Format		
	HDF	Hierarchical Data Format		
	JPEG JPG	Joint Photographic Experts Group		
	JP2 JPF JPX J2C J2K	JPEG 2000		
	PBM	Portable Bitmap		
	PCX	Paintbrush		
	PGM	Portable Graymap		
	PNG	Portable Network Graphics		
	PNM	Portable Any Map		
	PPM	Portable Pixmap		
	RAS	Sun™ Raster		
	TIFF TIF	Tagged Image File Format		
	XWD	X Window Dump		
	CUR	Windows Cursor resources	imread	none
	ICO	Windows Icon resources		



	CUR	Windows Cursor resources	imread	none
	ICO	Windows Icon resources		
Audio (all platforms)	AU SND	NeXT/Sun sound	audioread	audiowrite
	AIFF	Audio Interchange File Format		
	AIFC	Audio Interchange File Format, with compression codecs		
	FLAC	Free Lossless Audio Codec		
	OGG	Ogg Vorbis		
	WAV	Microsoft WAVE sound		
Audio (Windows)	M4A MP4	MPEG-4	audioread	audiowrite
	any	Formats supported by Microsoft Media Foundation	audioread	none
Audio (Mac)	M4A MP4	MPEG-4	audioread	audiowrite
Audio (Linux®)	any	Formats supported by GStreamer	audioread	none
Video (all platforms)	AVI	Audio Video Interleave	VideoReader	VideoWriter
	MJ2	Motion JPEG 2000		
Video (Windows)	MPG	MPEG-1	VideoReader	none
	ASF ASX WMV	Windows Media®		
	any	Formats supported by Microsoft DirectShow®		
Video (Windows 7 or later)	MP4 M4V	MPEG-4	VideoReader	VideoWriter
	MOV	QuickTime	VideoReader	none
	any	Formats supported by Microsoft Media Foundation		
Video (Mac)	MP4 M4V	MPEG-4	VideoReader	VideoWriter
	MPG	MPEG-1	VideoReader	none
	MOV	QuickTime		
	any	Formats supported by QuickTime, including .3gp, .3g2, and .dv		
Video (Linux)	any	Formats supported by your installed GStreamer plug-ins, including .ogg	VideoReader	none
Triangulation	STL	Stereolithography	stlread	stlwrite

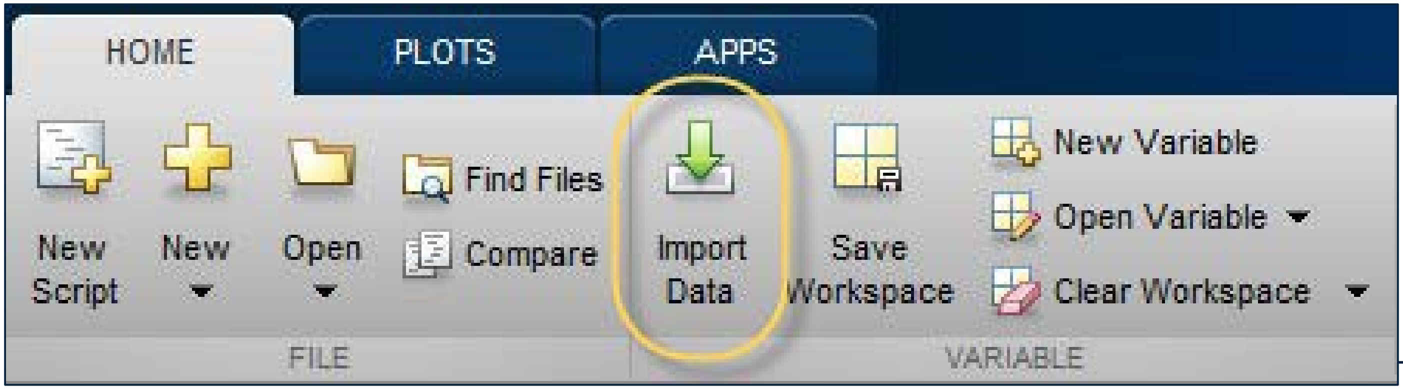
You can use web services such as a RESTful or WSDL to read and write data in an internet media type format such as JSON, XML, image, or text. For more information, see:

- Web Access

Video and Audio functionality not covered in ENGG100

Reading and Writing to Files

- Common file types can be accessed using built-in MATLAB functions
  - Process entire files at once
  - Little knowledge required about the format
  - Import either by selection box (below), or programmatically
- Files that are formatted differently require low-level I/O (input/output) functions
  - Requires in-depth knowledge of the file format
  - Access file data line-by-line or even character-by-character

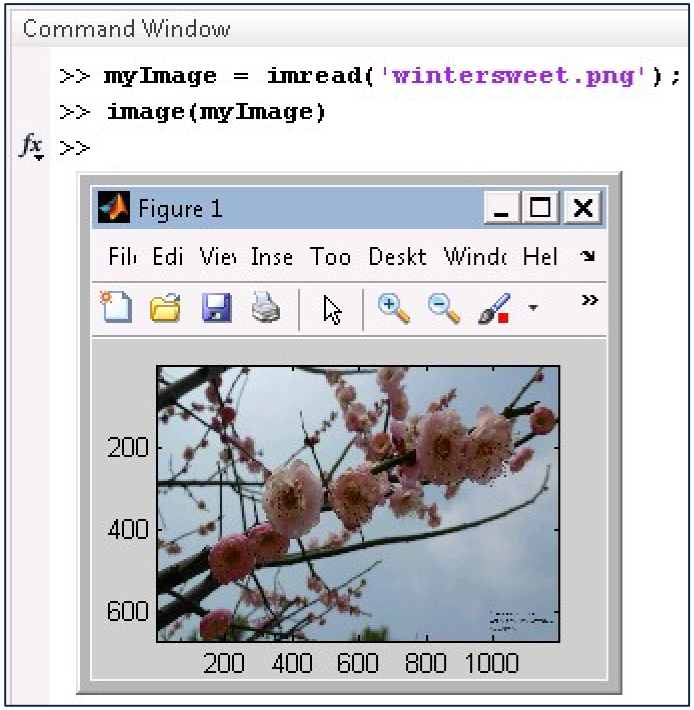


Loading and saving a ***.mat** file

- Loading and saving variables into/from a **`*.mat`** file
  - Allows for easy loading and saving of your current workspace and variables
  - You can return to the current workspace at a later stage, or on another computer

Command	Purpose
load	Load variables from file into workspace
save	Saves workspace variables in a file.
matfile	Access and change variables directly in MAT-files, without loading into memory

Importing/exporting Images

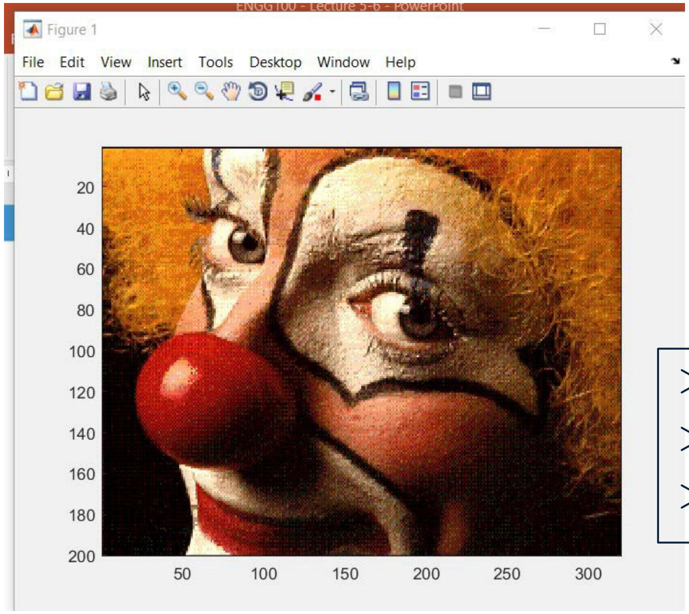


```
>> myImage = imread('wintersweet.png');
>> image(myImage)
>>
```

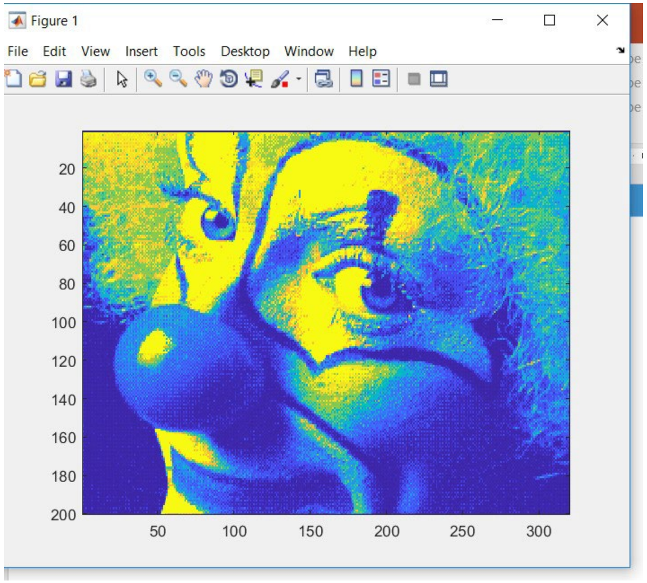
Command	Purpose
imshow	Display image
image	Display image from array
imagesc	Scale data and display image object
imread	Read image from graphics file
imwrite	Write image to graphics file
imfinfo	Information about graphics file
imformats	Manage image file format registry
frame2im	Return image data associated with movie frame
im2frame	Convert image to movie frame
im2java	Convert image to Java image

Index image function

A number of sample images are built into MATLAB


ENGG 100 - Lecture 5-6 - PowerPoint
Figure 1
File Edit View Insert Tools Desktop Window Help
20
40
60
80
100
120
140
160
180
200
50
100
150
200
250
300

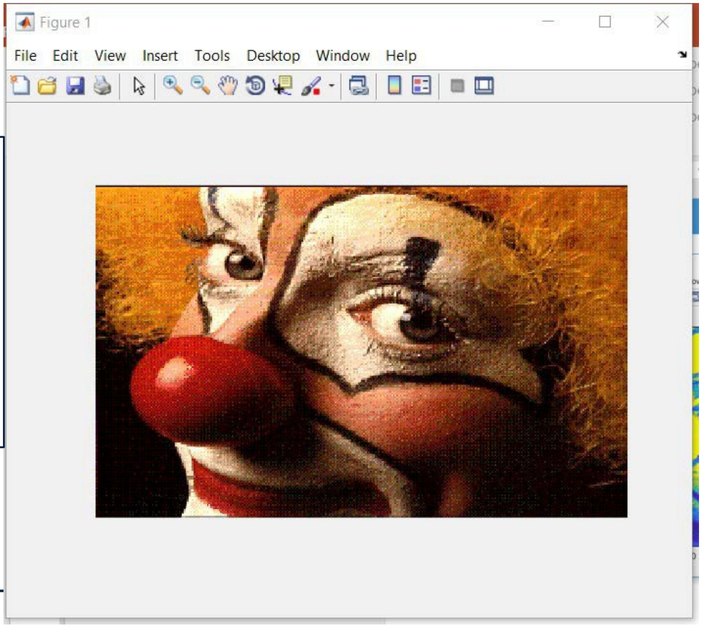
```
>> load clown
>> image(X)
```



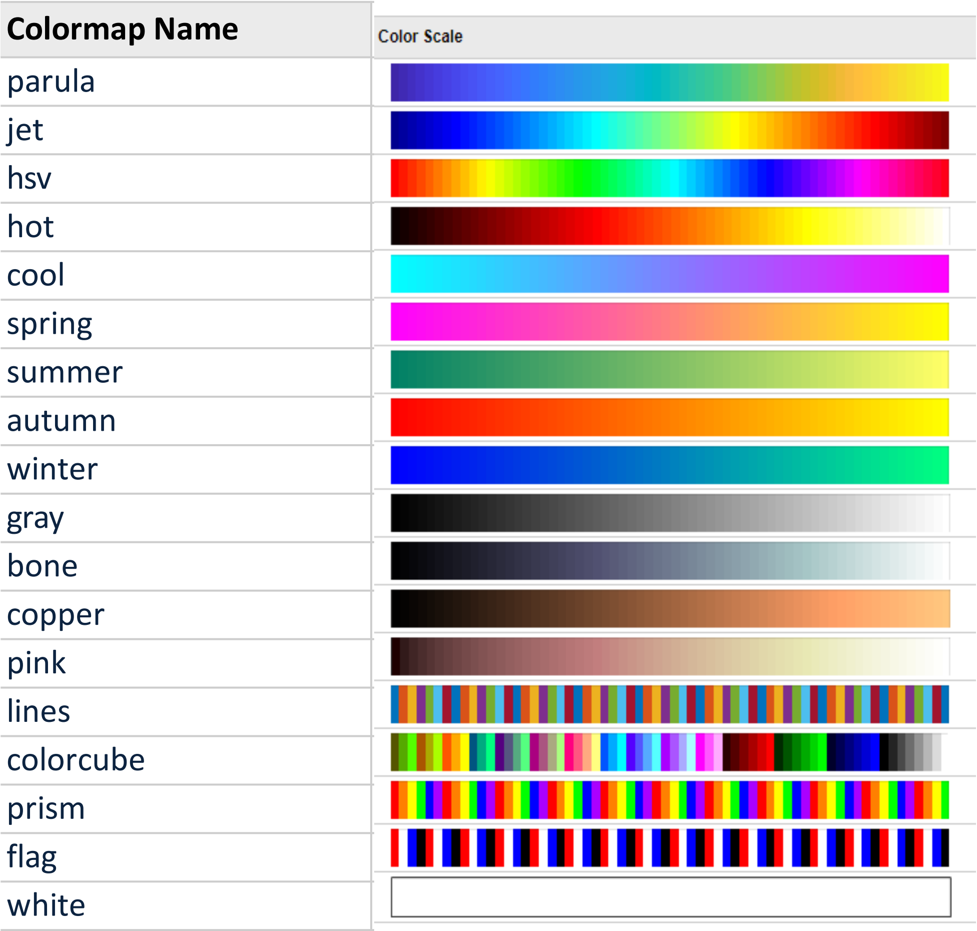
```
>> load clown
>> image(X)
>> colormap(map)
```

A colormap is matrix of values between 0 and 1 that define the colors for graphics objects such as surface, image, and patch objects.

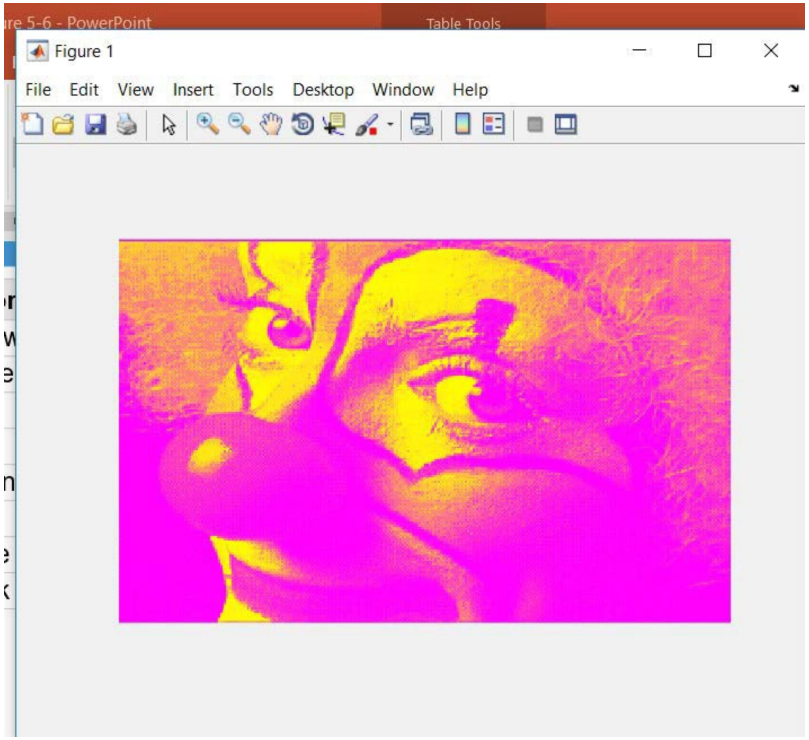
```
load clown
image(X)
colormap(map)
axis image
axis off
```



Color	RGB Triplet
yellow	[1 1 0]
magenta	[1 0 1]
cyan	[0 1 1]
red	[1 0 0]
green	[0 1 0]
blue	[0 0 1]
white	[1 1 1]
black	[0 0 0]



Colormap Name	Color Scale
parula	
jet	
hsv	
hot	
cool	
spring	
summer	
autumn	
winter	
gray	
bone	
copper	
pink	
lines	
colorcube	
prism	
flag	
white	



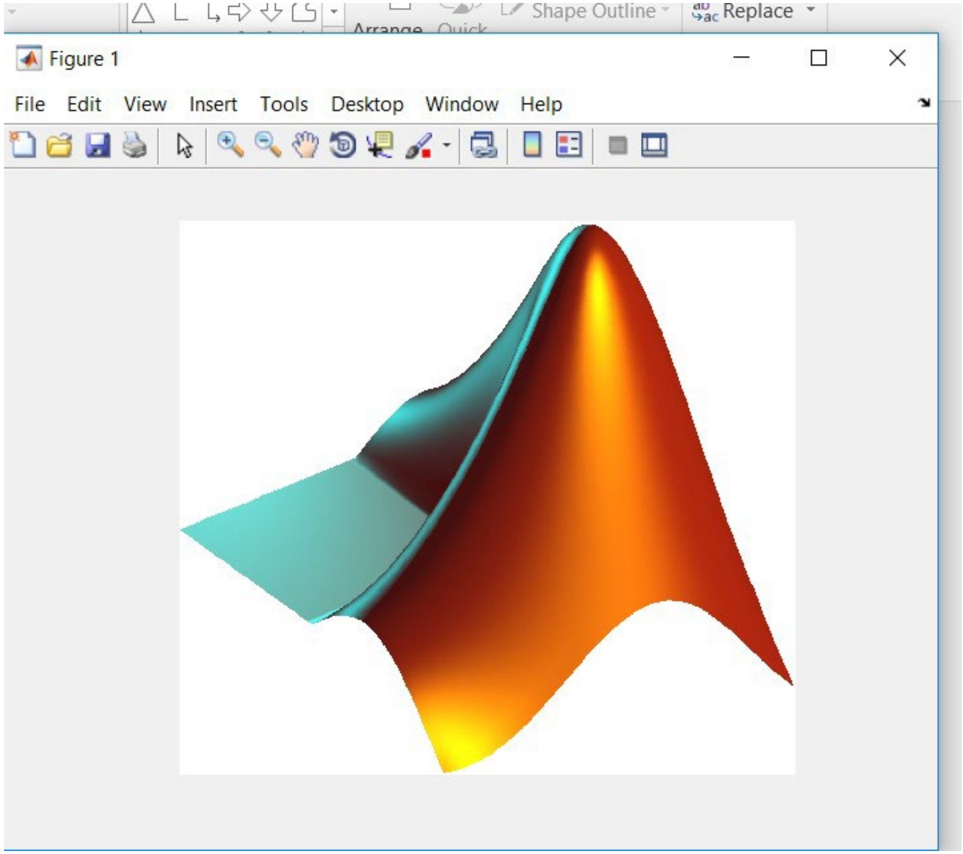
```
load clown
image(X)
colormap(spring)
axis image
axis off
```

UNIVERSITY
OF WOLLONGONG
IN DUBAI

True Color (RGB) Images

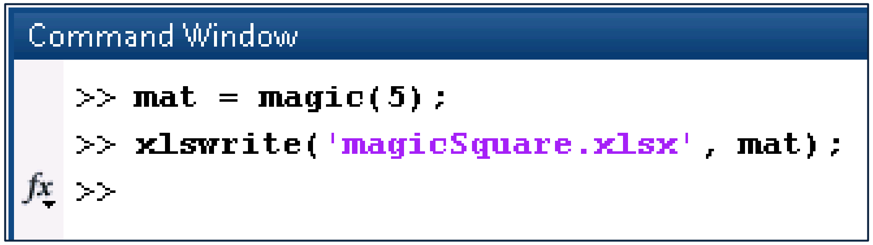
Stored as 3D matrix *m x n x 3*

```
X=imread('Matlab_Logo.png');
image(X)
axis image
axis off
```



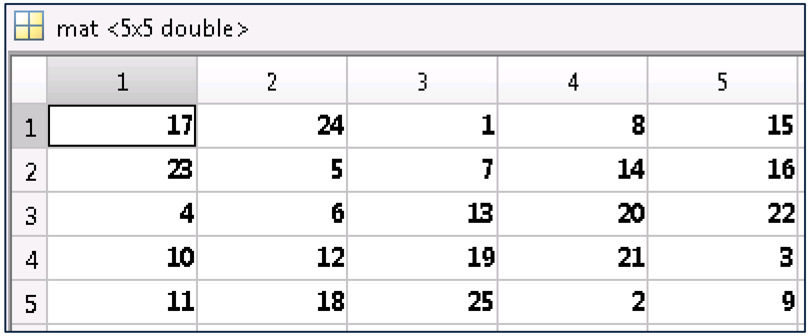
We don’t need to load a colormap because the colour-intensity information is included in the matrix

Importing/Exporting Spreadsheets



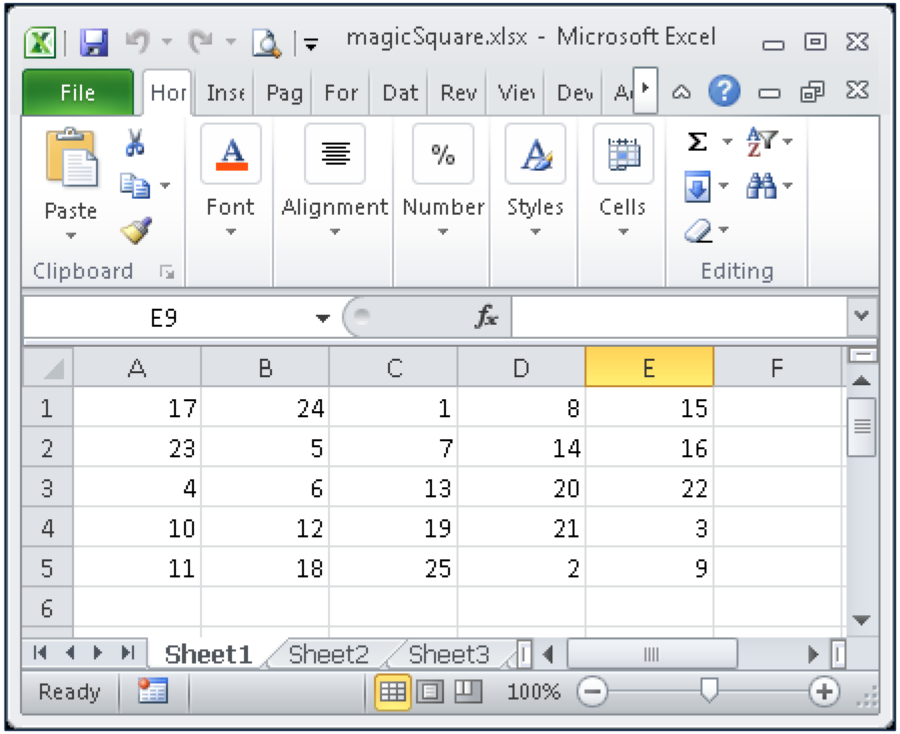
Command Window

```
>> mat = magic(5);
>> xlswrite('magicSquare.xlsx', mat);
>>
```



mat <5x5 double>

	1	2	3	4	5
1	17	24	1	8	15
2	23	5	7	14	16
3	4	6	13	20	22
4	10	12	19	21	3
5	11	18	25	2	9



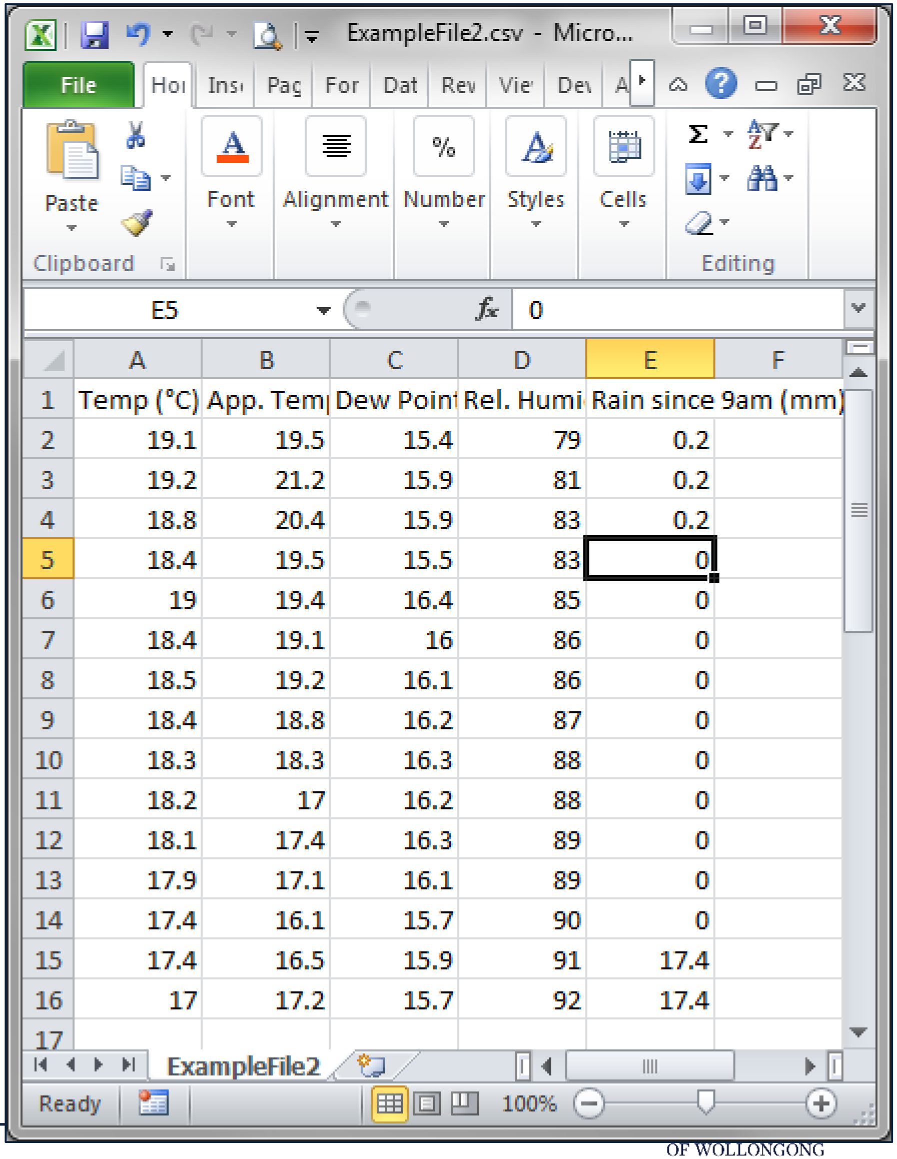
magicSquare.xlsx - Microsoft Excel

	A	B	C	D	E	F
1	17	24	1	8	15	
2	23	5	7	14	16	
3	4	6	13	20	22	
4	10	12	19	21	3	
5	11	18	25	2	9	
6						

Command	Purpose
xlsfinfo	Determine if file contains Microsoft Excel spreadsheet
xlsread	Read Microsoft Excel spreadsheet file
xlswrite	Write Microsoft Excel spreadsheet file

Importing/Exporting Text and Comma-Separated Values files

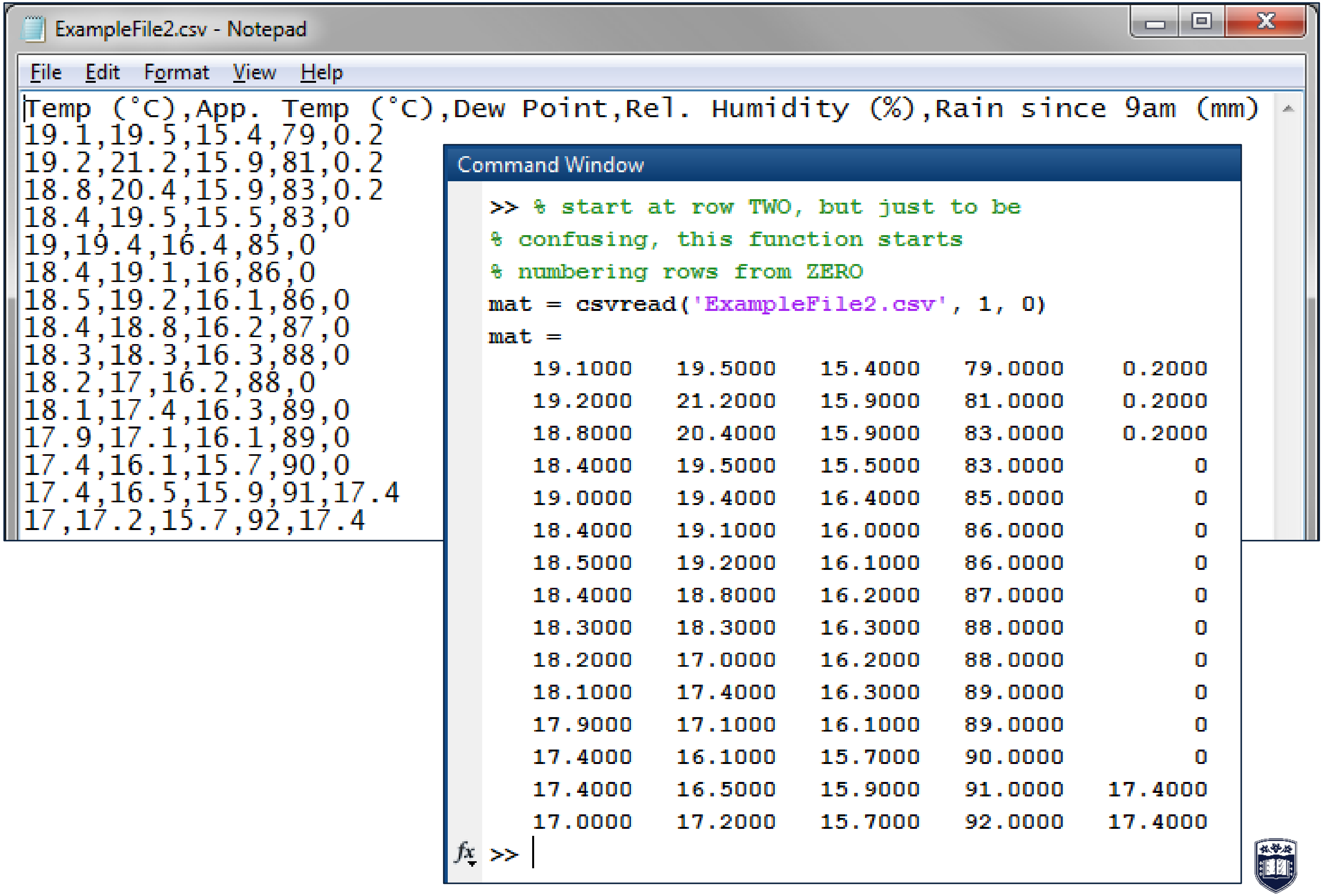
Command	Purpose
csvread	Read comma-separated value (CSV) file
csvwrite	Write comma-separated value file
readmatrix	creates an array by reading column-oriented data from a file
dlmwrite	Write matrix to ASCII-delimited file
textscan	Read formatted data from text file or string
type	Display contents of file



E5 = 0

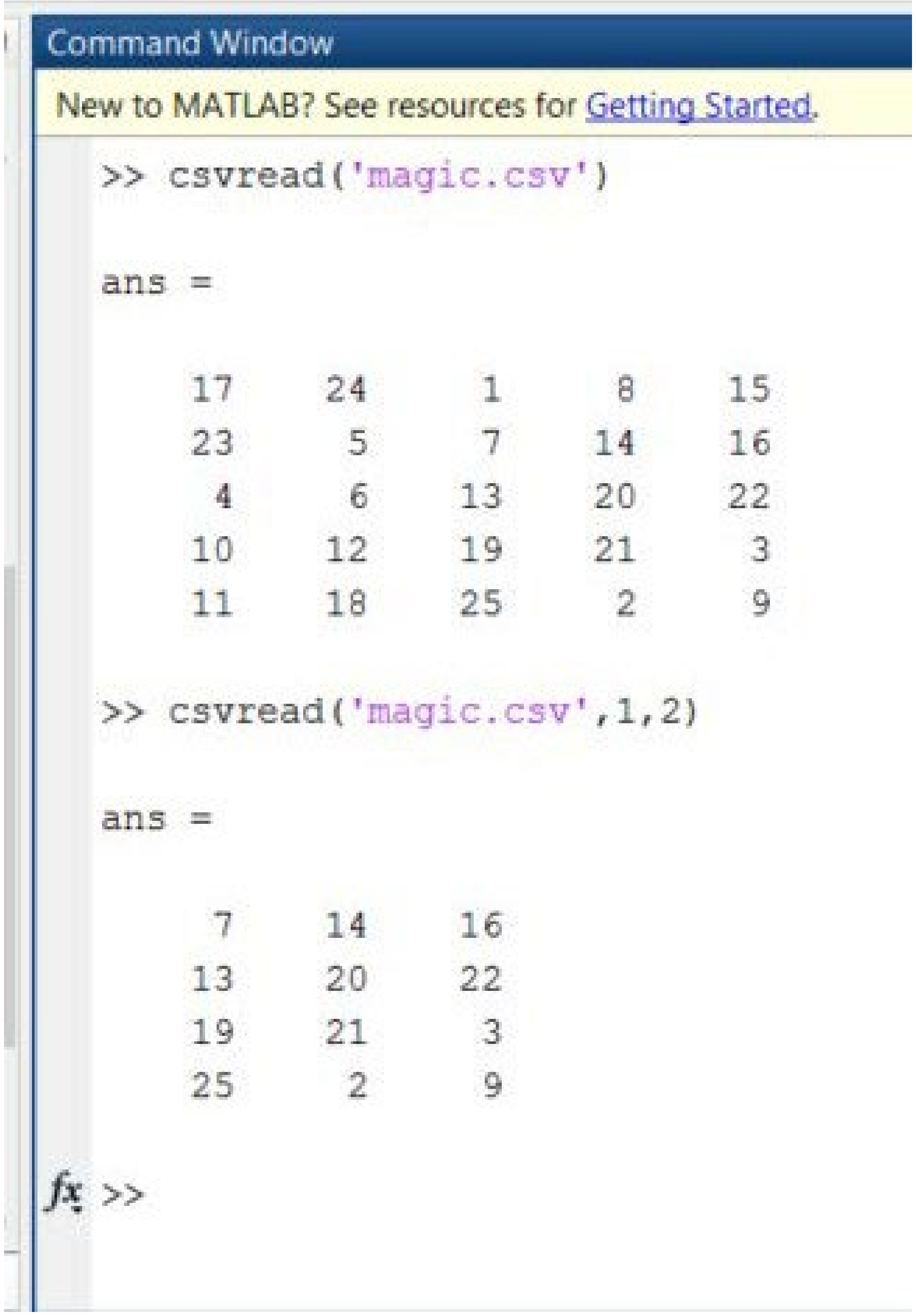
	A	B	C	D	E	F
1	Temp (°C)	App. Tem	Dew Point	Rel. Humi	Rain since 9am (mm)	
2	19.1	19.5	15.4	79	0.2	
3	19.2	21.2	15.9	81	0.2	
4	18.8	20.4	15.9	83	0.2	
5	18.4	19.5	15.5	83	0	
6	19	19.4	16.4	85	0	
7	18.4	19.1	16	86	0	
8	18.5	19.2	16.1	86	0	
9	18.4	18.8	16.2	87	0	
10	18.3	18.3	16.3	88	0	
11	18.2	17	16.2	88	0	
12	18.1	17.4	16.3	89	0	
13	17.9	17.1	16.1	89	0	
14	17.4	16.1	15.7	90	0	
15	17.4	16.5	15.9	91	17.4	
16	17	17.2	15.7	92	17.4	
17						

Comma-Separated Values files



```
Temp (°C),App. Temp (°C),Dew Point,Rel. Humidity (%),Rain since 9am (mm)
19.1,19.5,15.4,79,0.2
19.2,21.2,15.9,81,0.2
18.8,20.4,15.9,83,0.2
18.4,19.5,15.5,83,0
19,19.4,16.4,85,0
18.4,19.1,16,86,0
18.5,19.2,16.1,86,0
18.4,18.8,16.2,87,0
18.3,18.3,16.3,88,0
18.2,17,16.2,88,0
18.1,17.4,16.3,89,0
17.9,17.1,16.1,89,0
17.4,16.1,15.7,90,0
17.4,16.5,15.9,91,17.4
17,17.2,15.7,92,17.4
```

```
>> % start at row TWO, but just to be
% confusing, this function starts
% numbering rows from ZERO
mat = csvread('ExampleFile2.csv', 1, 0)
mat =
    19.1000   19.5000   15.4000   79.0000    0.2000
    19.2000   21.2000   15.9000   81.0000    0.2000
    18.8000   20.4000   15.9000   83.0000    0.2000
    18.4000   19.5000   15.5000   83.0000         0
    19.0000   19.4000   16.4000   85.0000         0
    18.4000   19.1000   16.0000   86.0000         0
    18.5000   19.2000   16.1000   86.0000         0
    18.4000   18.8000   16.2000   87.0000         0
    18.3000   18.3000   16.3000   88.0000         0
    18.2000   17.0000   16.2000   88.0000         0
    18.1000   17.4000   16.3000   89.0000         0
    17.9000   17.1000   16.1000   89.0000         0
    17.4000   16.1000   15.7000   90.0000         0
    17.4000   16.5000   15.9000   91.0000   17.4000
    17.0000   17.2000   15.7000   92.0000   17.4000
fx >>
```



```
>> csvread('magic.csv')

ans =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

>> csvread('magic.csv',1,2)

ans =

     7    14    16
    13    20    22
    19    21     3
    25     2     9

>>
```

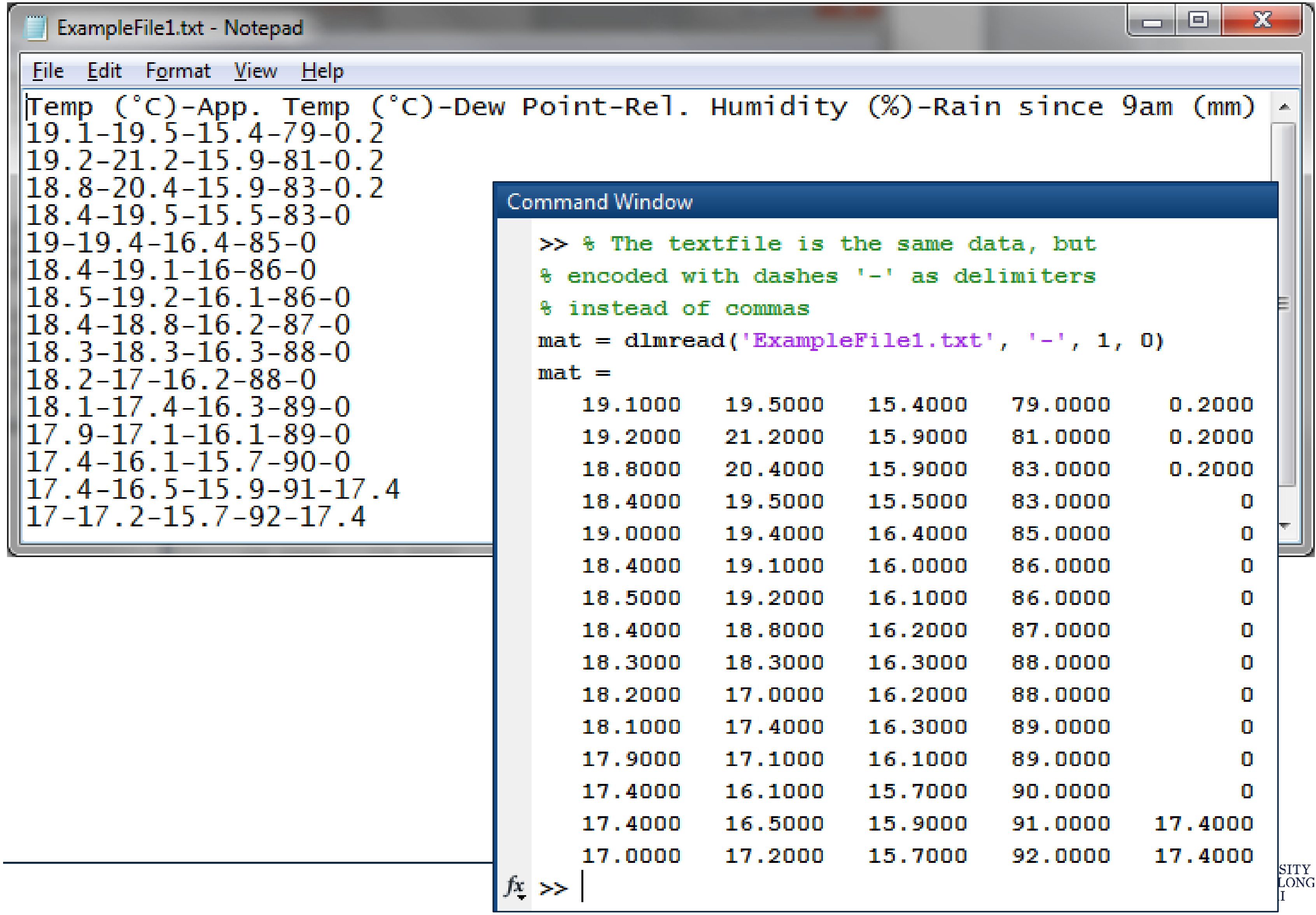
M = csvread(filename,R1,C1)

reads data from the file starting at row offset R1 and column offset C1.

For example, the offsets R1=0, C1=0 specify the first value in the file.

csvread('magic.csv,1,2) reads the data from the 2nd row and 3rd column

Delimited Text Files



```
Temp (°C)-App. Temp (°C)-Dew Point-Rel. Humidity (%)-Rain since 9am (mm)
19.1-19.5-15.4-79-0.2
19.2-21.2-15.9-81-0.2
18.8-20.4-15.9-83-0.2
18.4-19.5-15.5-83-0
19-19.4-16.4-85-0
18.4-19.1-16-86-0
18.5-19.2-16.1-86-0
18.4-18.8-16.2-87-0
18.3-18.3-16.3-88-0
18.2-17-16.2-88-0
18.1-17.4-16.3-89-0
17.9-17.1-16.1-89-0
17.4-16.1-15.7-90-0
17.4-16.5-15.9-91-17.4
17-17.2-15.7-92-17.4
```

```
>> % The textfile is the same data, but
% encoded with dashes '-' as delimiters
% instead of commas
mat = dlmread('ExampleFile1.txt', '-', 1, 0)
mat =
   19.1000   19.5000   15.4000   79.0000    0.2000
   19.2000   21.2000   15.9000   81.0000    0.2000
   18.8000   20.4000   15.9000   83.0000    0.2000
   18.4000   19.5000   15.5000   83.0000         0
   19.0000   19.4000   16.4000   85.0000         0
   18.4000   19.1000   16.0000   86.0000         0
   18.5000   19.2000   16.1000   86.0000         0
   18.4000   18.8000   16.2000   87.0000         0
   18.3000   18.3000   16.3000   88.0000         0
   18.2000   17.0000   16.2000   88.0000         0
   18.1000   17.4000   16.3000   89.0000         0
   17.9000   17.1000   16.1000   89.0000         0
   17.4000   16.1000   15.7000   90.0000         0
   17.4000   16.5000   15.9000   91.0000   17.4000
   17.0000   17.2000   15.7000   92.0000   17.4000
fx >>
```

Low-level File I/O

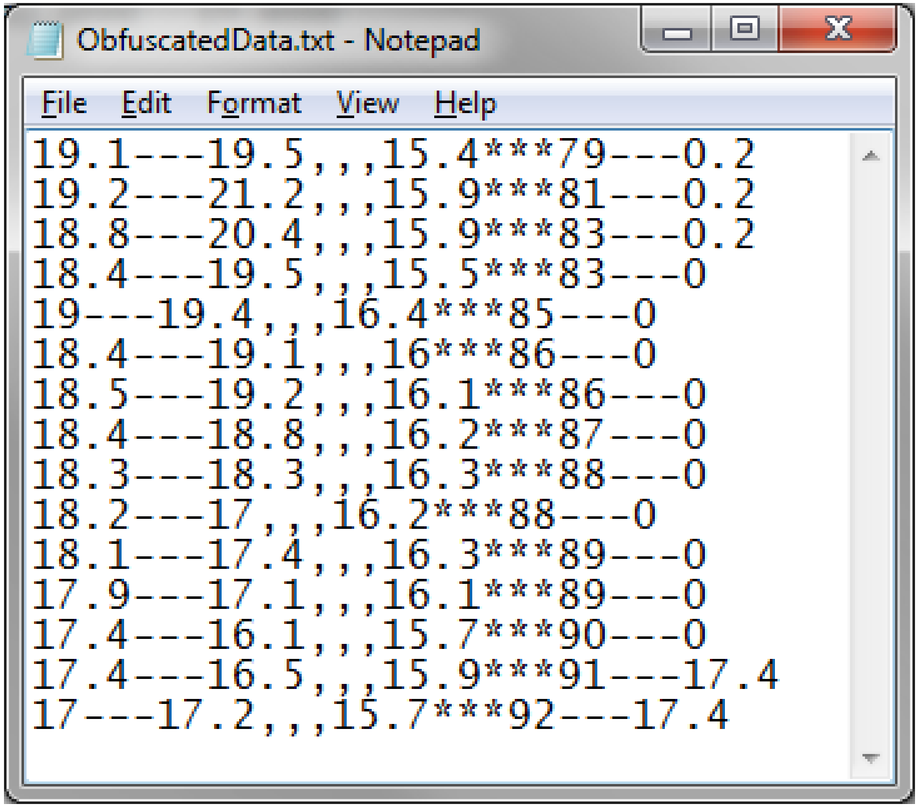
Note! every line has the same format:

```
[]---[],,,[]***[]---[]
```

where **[]** *stands for a number*

- The MATLAB built-in functions are great when we have data formatted in a standard way

- But what if the data we have is not stored in a common/known format?
  - We may need to read the data line-by-line
  - We may even need to read the data character-by-character



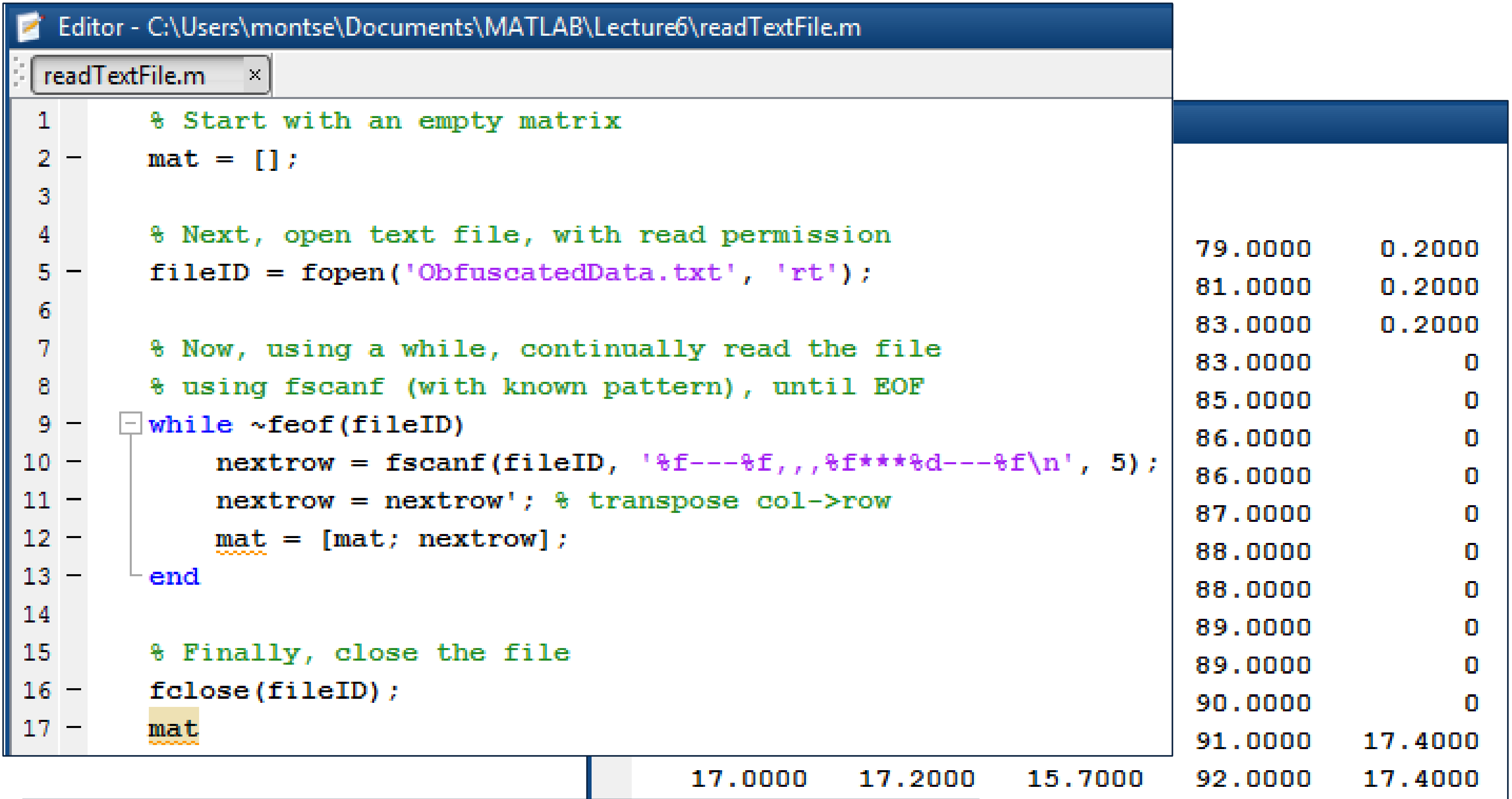
```
19.1---19.5,,,15.4***79---0.2
19.2---21.2,,,15.9***81---0.2
18.8---20.4,,,15.9***83---0.2
18.4---19.5,,,15.5***83---0
19---19.4,,,16.4***85---0
18.4---19.1,,,16***86---0
18.5---19.2,,,16.1***86---0
18.4---18.8,,,16.2***87---0
18.3---18.3,,,16.3***88---0
18.2---17,,,16.2***88---0
18.1---17.4,,,16.3***89---0
17.9---17.1,,,16.1***89---0
17.4---16.1,,,15.7***90---0
17.4---16.5,,,15.9***91---17.4
17---17.2,,,15.7***92---17.4
```

Low-level File I/O functions

Command	Purpose
fclose	Close one or all open files
feof	Test for end-of-file
ferror	Information about file I/O errors
fgetl	Read line from file, removing newline characters
fgets	Read line from file, keeping newline characters
fileread	Read contents of file into string
fopen	Open file, or obtain information about open files
fprintf	Write data to text file
fread	Read data from binary file
frewind	Move file position indicator to beginning of open file
fscanf	Read data from text file
fseek	Move to specified position in file
ftell	Position in open file
fwrite	Write data to binary file

To open files in text mode, attach the letter 't' to the permission argument, such as 'rt' or 'wt+'.

Using Low-Level functions



Editor - C:\Users\montse\Documents\MATLAB\Lecture6\readTextFile.m
readTextFile.m ×

```
    % Start with an empty matrix
    mat = [];

    % Next, open text file, with read permission
    fileID = fopen('ObfuscatedData.txt', 'rt');

    % Now, using a while, continually read the file
    % using fscanf (with known pattern), until EOF
    while ~feof(fileID)
        nextrow = fscanf(fileID, '%f---%f,,,%f***%d---%f\n', 5);
        nextrow = nextrow'; % transpose col->row
        mat = [mat; nextrow];
    end

    % Finally, close the file
    fclose(fileID);
    mat
```

```
                                          79.0000    0.2000
                                          81.0000    0.2000
                                          83.0000    0.2000
                                          83.0000         0
                                          85.0000         0
                                          86.0000         0
                                          86.0000         0
                                          87.0000         0
                                          88.0000         0
                                          88.0000         0
                                          89.0000         0
                                          89.0000         0
                                          90.0000         0
                                          91.0000   17.4000
   17.0000   17.2000   15.7000            92.0000   17.4000
```

Open the file for reading, and obtain the file identifier, fileID.

Use '%f' to specify floating-point numbers.

Plotting and GUIs

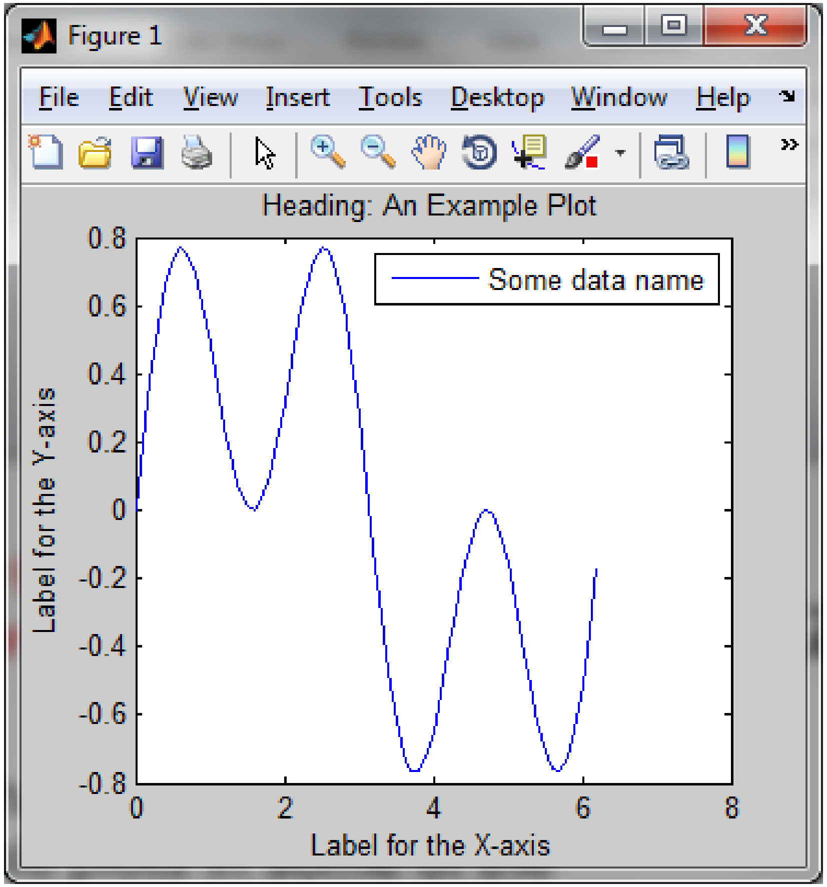
Plotting Data

x.axis array (1x63)

y-axis array (1x63)

- We have already seen some simple plots in some of the previous tuts/WSAs, but haven’t investigated further
  - Usually it is to display an array of values over time

```
x = 0:0.1:2*pi;
y = sin(2*x).*cos(x);
plot(x,y);
title('Heading: An Example Plot');
xlabel('Label for the X-axis');
ylabel('Label for the Y-axis');
legend('Some data name');
```



Plot the (x,y) coords

NB: vectors MUST be the same length!

LineSpec — Line style, marker, and color

Color	Description
y	yellow
m	magenta
c	cyan
r	red
g	green
b	blue
w	white
k	black

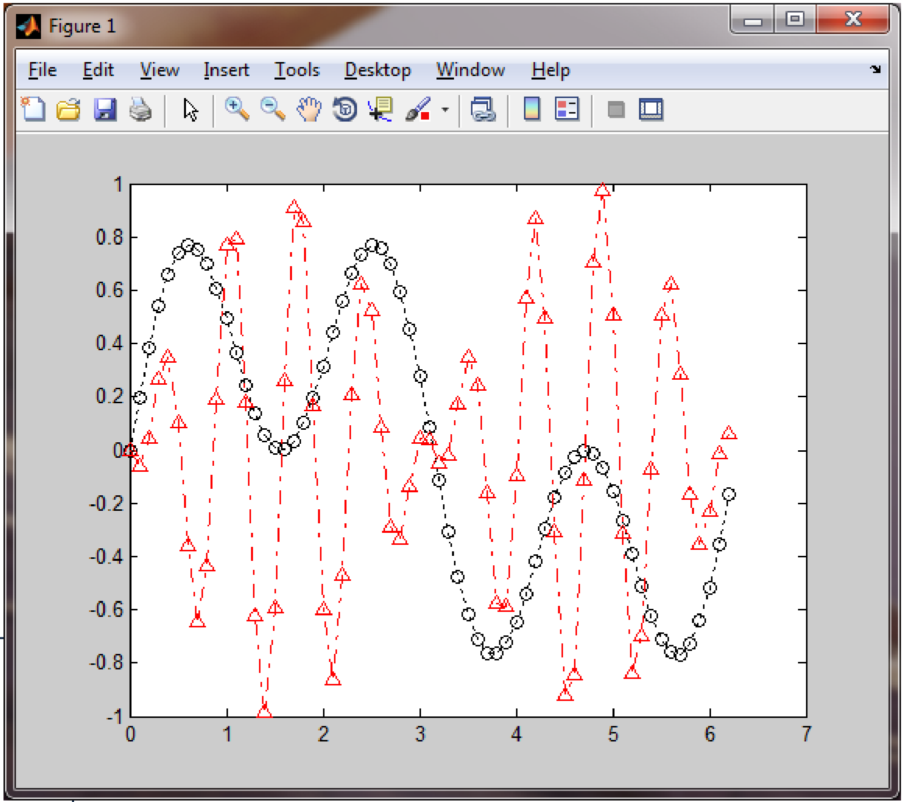
Line Style	Description
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

Marker	Description
o	Circle
+	Plus sign
*	Asterisk
.	Point
x	Cross
s	Square
d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
p	Pentagram
h	Hexagram

```
plot(x, y, 'ok:', x, z, '^r-.');
```

Multiple plots on same axes

- We can place multiple data series on the same graph
- We can also format the look by changing colour, marker and style.
    - – See the help file for the **`plot`** function to see the full range of styles



```
x = 0:0.1:2*pi;
y = sin(2*x).*cos(x);
z = -sin(x).*cos(9*x);
plot(x, y, 'ok:', x, z, '^r-.');
```

meshgrid

- [X,Y] = meshgrid(x,y) returns 2-D grid coordinates based on the coordinates contained in vectors x and y.

```
>> x = 1:3;
y = 1:5;
[X,Y] = meshgrid(x,y)
```

```
X =

     1     2     3
     1     2     3
     1     2     3
     1     2     3
     1     2     3
```

```
Y =

     1     1     1
     2     2     2
     3     3     3
     4     4     4
     5     5     5
```

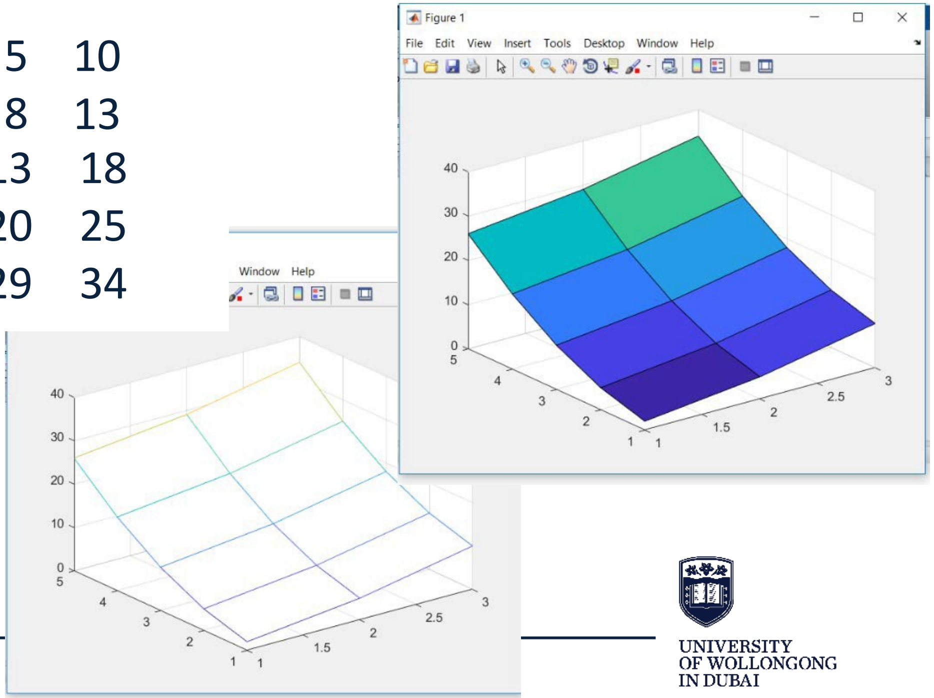
Evaluate the expression x^2+y^2 the 2-D grid.

```
>> f=X.^2 + Y.^2;
>> surf(X,Y,f)
or
>> mesh(X,Y,f)
```

```
ans =


     2     5    10
     5     8    13
    10    13    18
    17    20    25
    26    29    34
```

surf(X,Y,Z) = surface plot. The function al uses Z for the colour data, so colour is proportional to height.

mesh(X,Y,Z) = draws a wireframe mesh with colour determined by Z, so colour is proportional to surface height.



UNIVERSITY OF WOLLONGONG IN DUBAI

Graphing in 3D

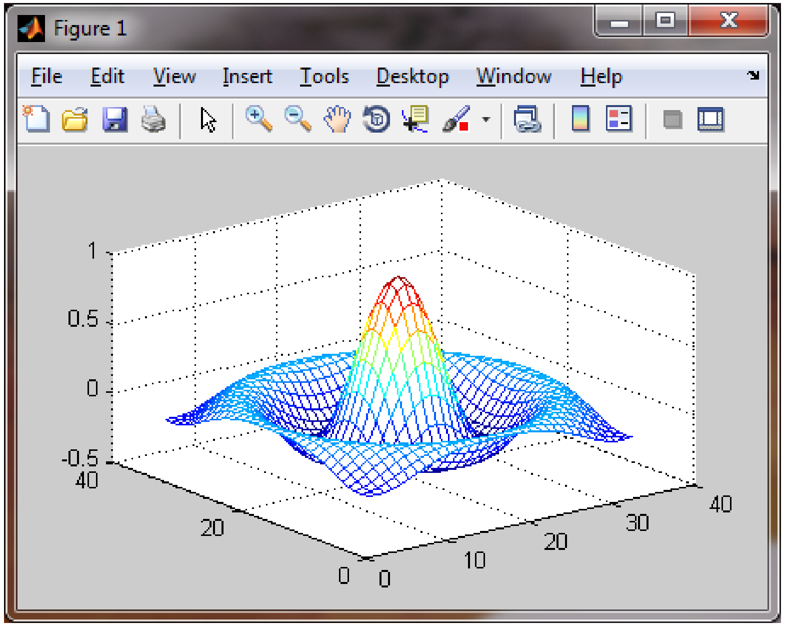
- Requires **x** / **y** component matrices using **`meshgrid`**
- And also a function:

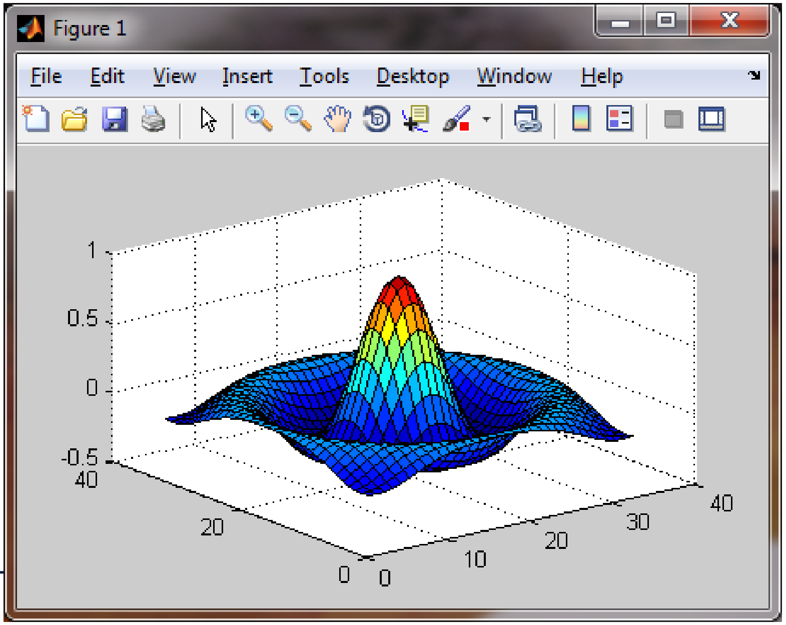
$$z = f(x, y)$$

- Then plot using either **`mesh`** or **`surf`**

$$z = \frac{\sin\left(\sqrt{x^2 + y^2}\right)}{\sqrt{x^2 + y^2}}$$

```
% Mexican Hat plot from Prac1
[x, y] = meshgrid(-8:0.5:8);
r = sqrt(x.^2 + y.^2);
z = sin(r) ./r;
mesh(z)   % or surf(z);
```



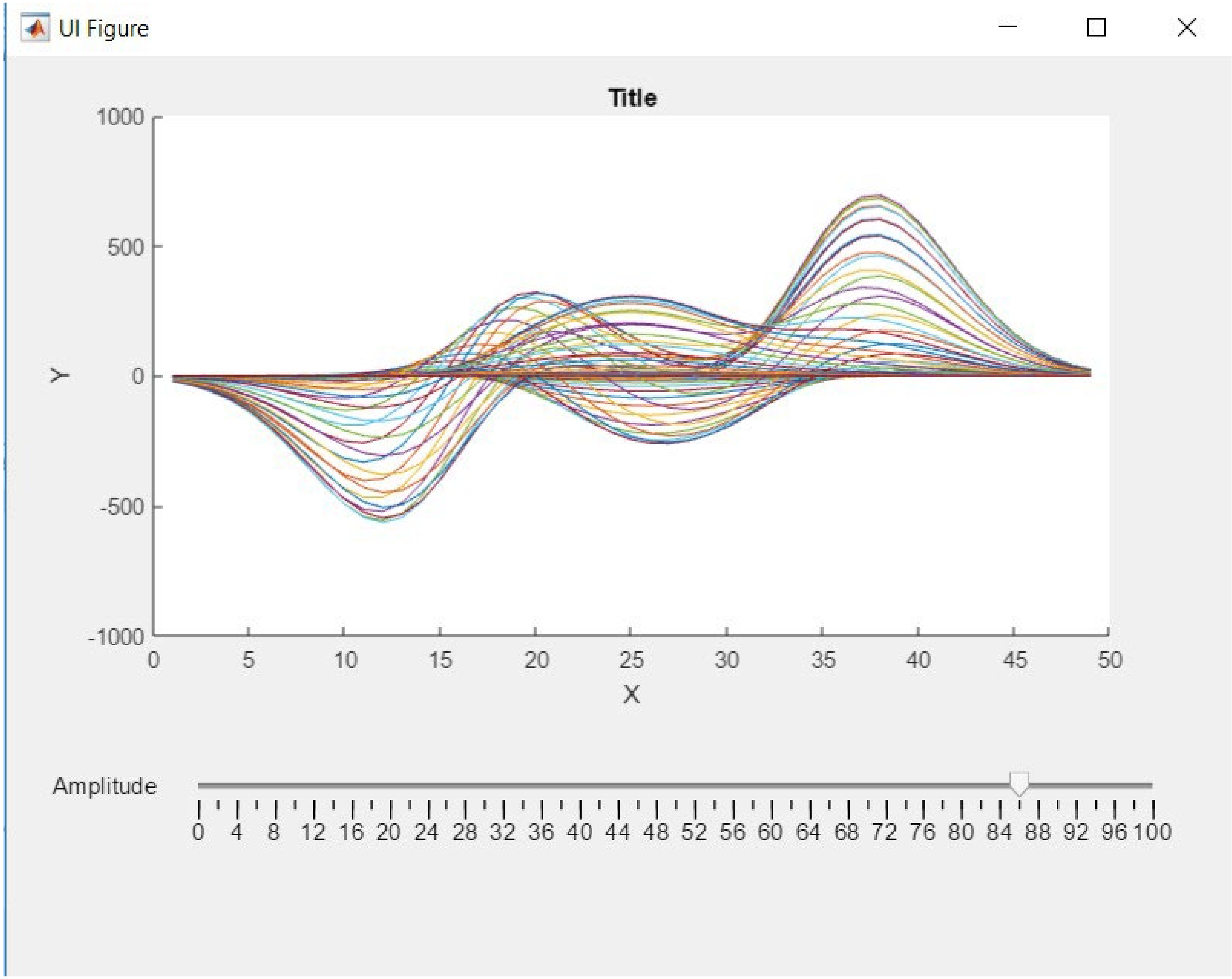


Graphical User Interface (GUI)

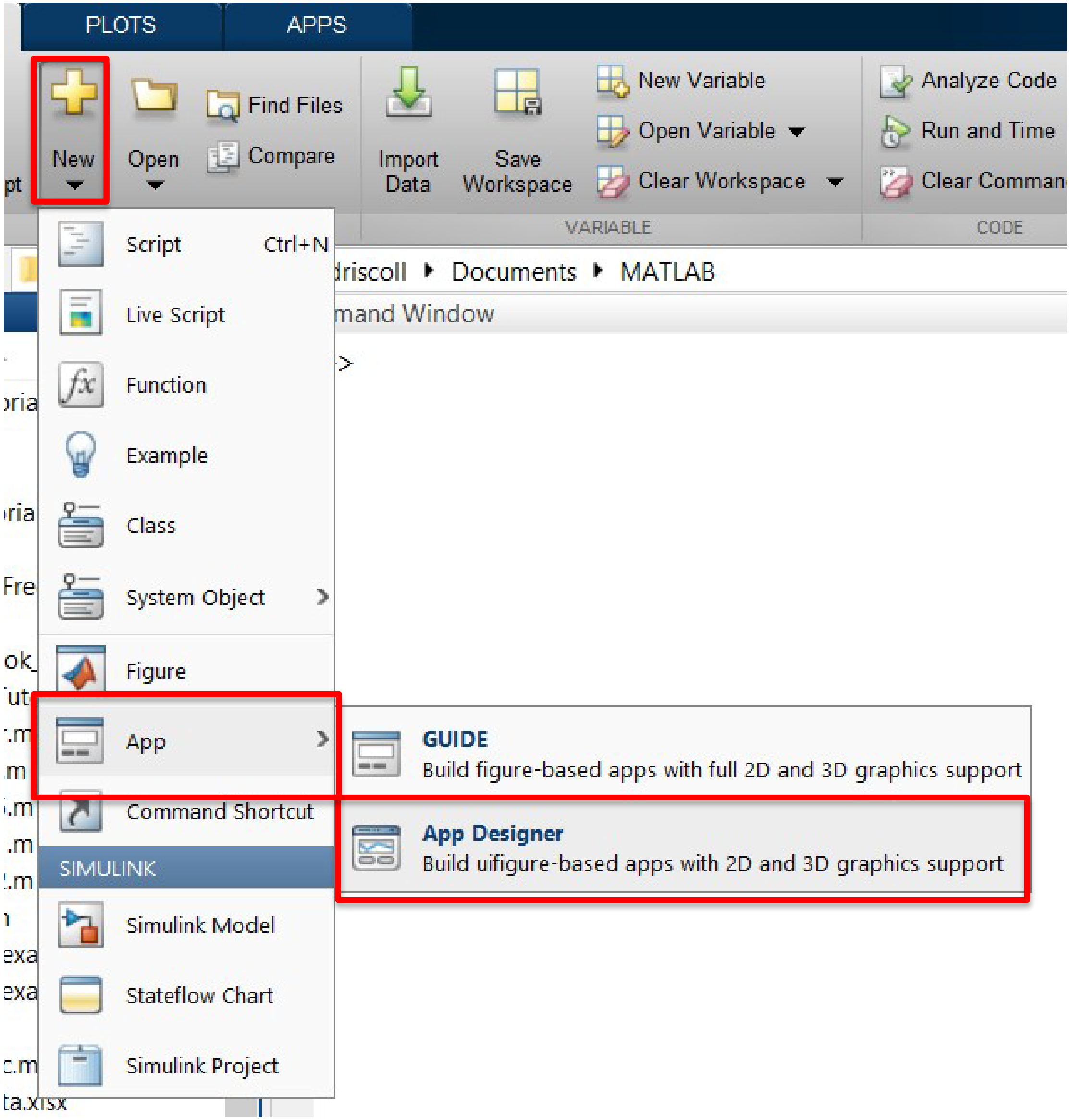
- Most engineering applications eventually require a user-friendly way to interact with the data
- As Engineers, we might know how the ‘computation engine’ works, but our clients may not have the necessary background
- We can build simple click-on-button graphical interfaces to allow a non-programmer to operate the functions we have written

We are going to use the new App Designer feature in MATLAB for GUI’s

GUI Example – App Designer



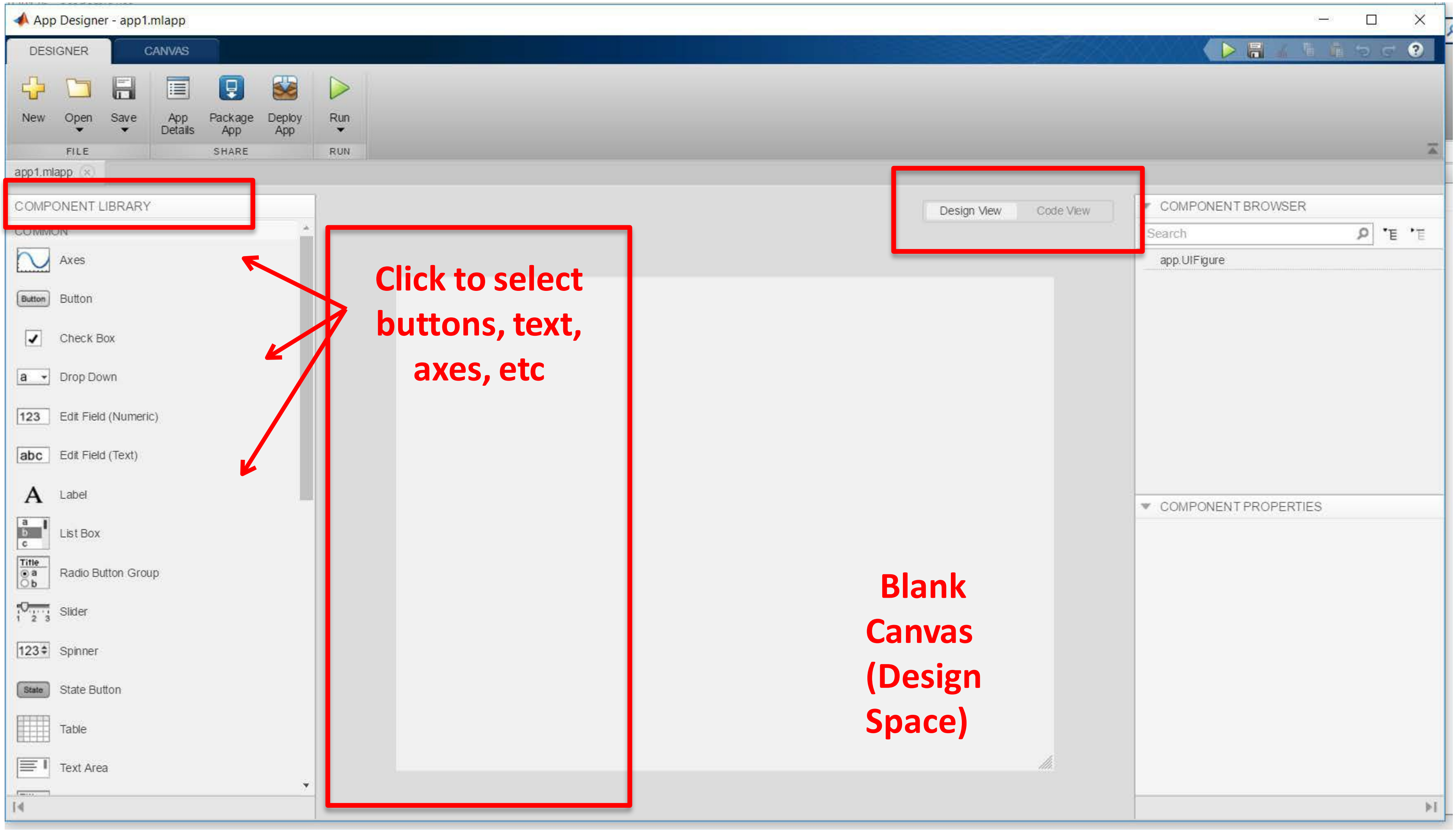
Creating a new GUI



GUIDE
Build figure-based apps with full 2D and 3D graphics support

App Designer
Build uifigure-based apps with 2D and 3D graphics support

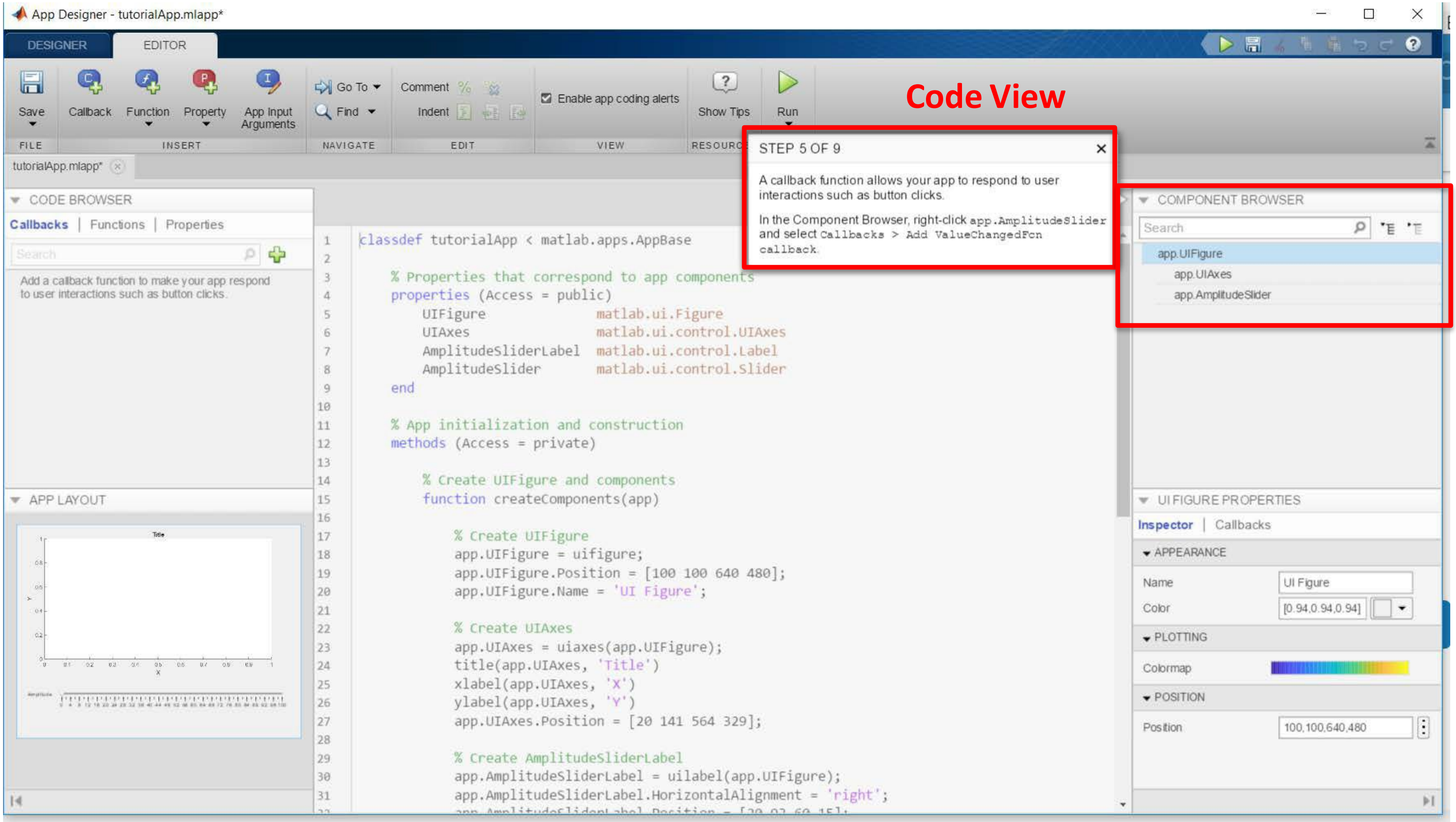
App Designer



Click to select buttons, text, axes, etc

Blank Canvas (Design Space)

App Designer – Example



Code View

STEP 5 OF 9

A callback function allows your app to respond to user interactions such as button clicks.

In the Component Browser, right-click `app.AmplitudeSlider` and select `Callbacks > Add ValueChangedFcn callback`.

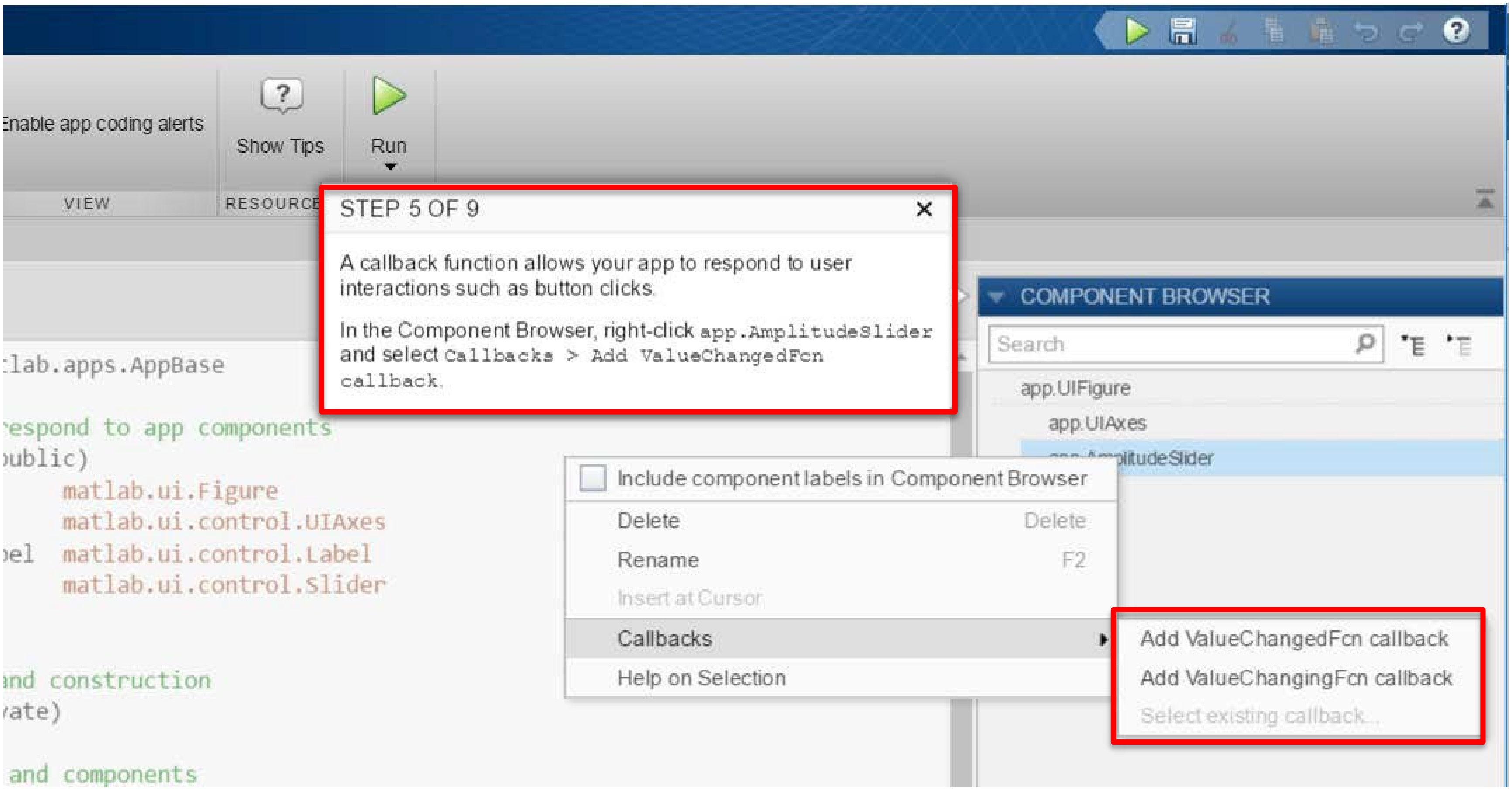
Add a callback function to make your app respond to user interactions such as button clicks.

```
classdef tutorialApp < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                 matlab.ui.Figure
        UIAxes                   matlab.ui.control.UIAxes
        AmplitudeSliderLabel     matlab.ui.control.Label
        AmplitudeSlider          matlab.ui.control.Slider
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'UI Figure';

            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'Title')
            xlabel(app.UIAxes, 'X')
            ylabel(app.UIAxes, 'Y')
            app.UIAxes.Position = [20 141 564 329];

            % Create AmplitudeSliderLabel
            app.AmplitudeSliderLabel = uilabel(app.UIFigure);
            app.AmplitudeSliderLabel.HorizontalAlignment = 'right';
```

Name	UI Figure
Color	[0.94,0.94,0.94]
Colormap	
Position	100,100,640,480

UNIVERSITY
OF WOLLONGONG
IN DUBAI

App Designer – Example



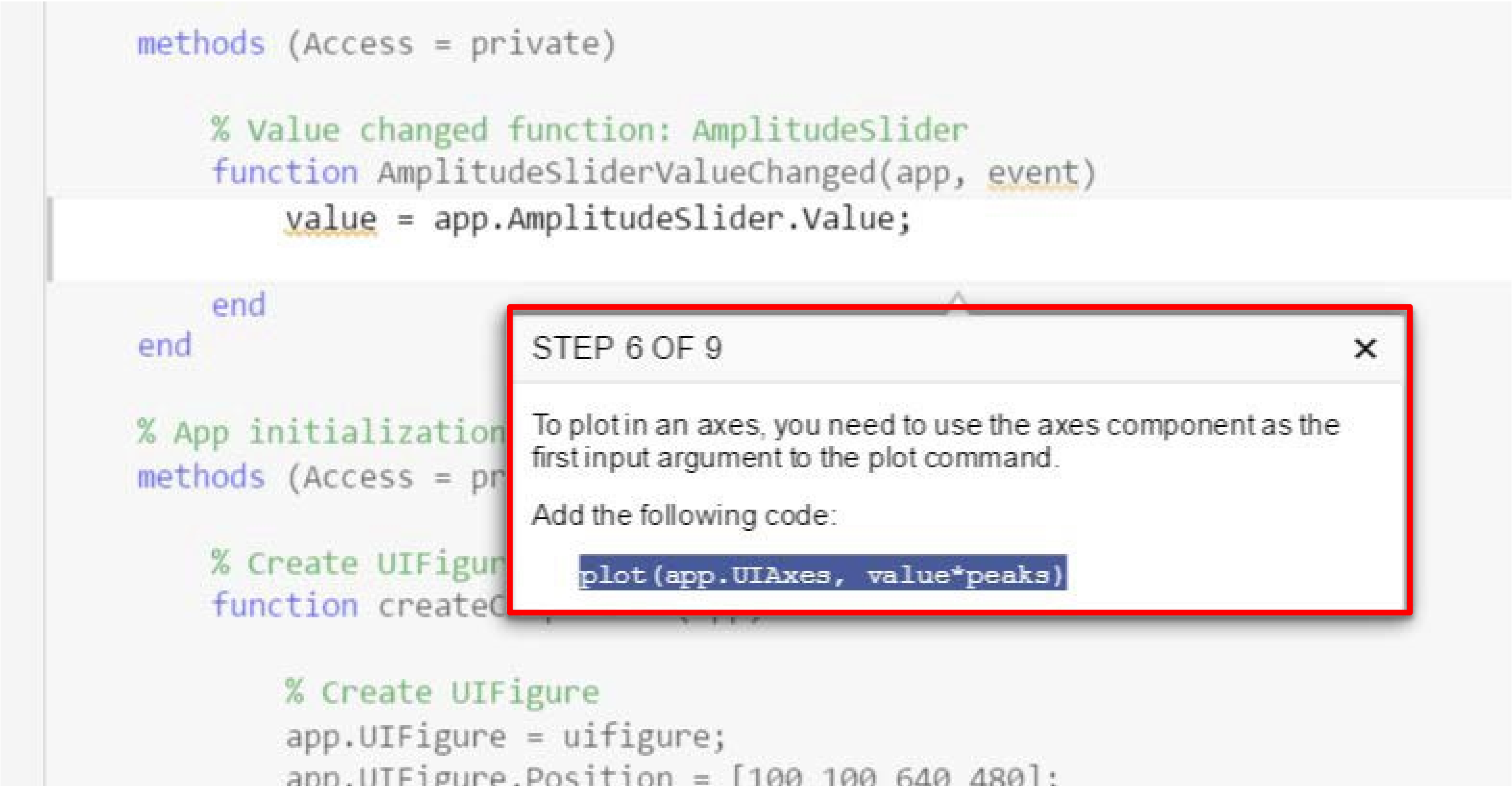
STEP 5 OF 9 ✕

A callback function allows your app to respond to user interactions such as button clicks.

In the Component Browser, right-click `app.AmplitudeSlider` and select `Callbacks > Add ValueChangedFcn callback`.

```
:lab.apps.AppBase

respond to app components
public)
        matlab.ui.Figure
        matlab.ui.control.UIAxes
bel     matlab.ui.control.Label
        matlab.ui.control.Slider


and construction
vate)

 and components
```

App Designer – Example



```
methods (Access = private)

    % Value changed function: AmplitudeSlider
    function AmplitudeSliderValueChanged(app, event)
        value = app.AmplitudeSlider.Value;

    end
end

% App initialization
methods (Access = pr

    % Create UIFigur
    function createC

        % Create UIFigure
        app.UIFigure = uifigure;
        app.UIFigure.Position = [100 100 640 480];
```

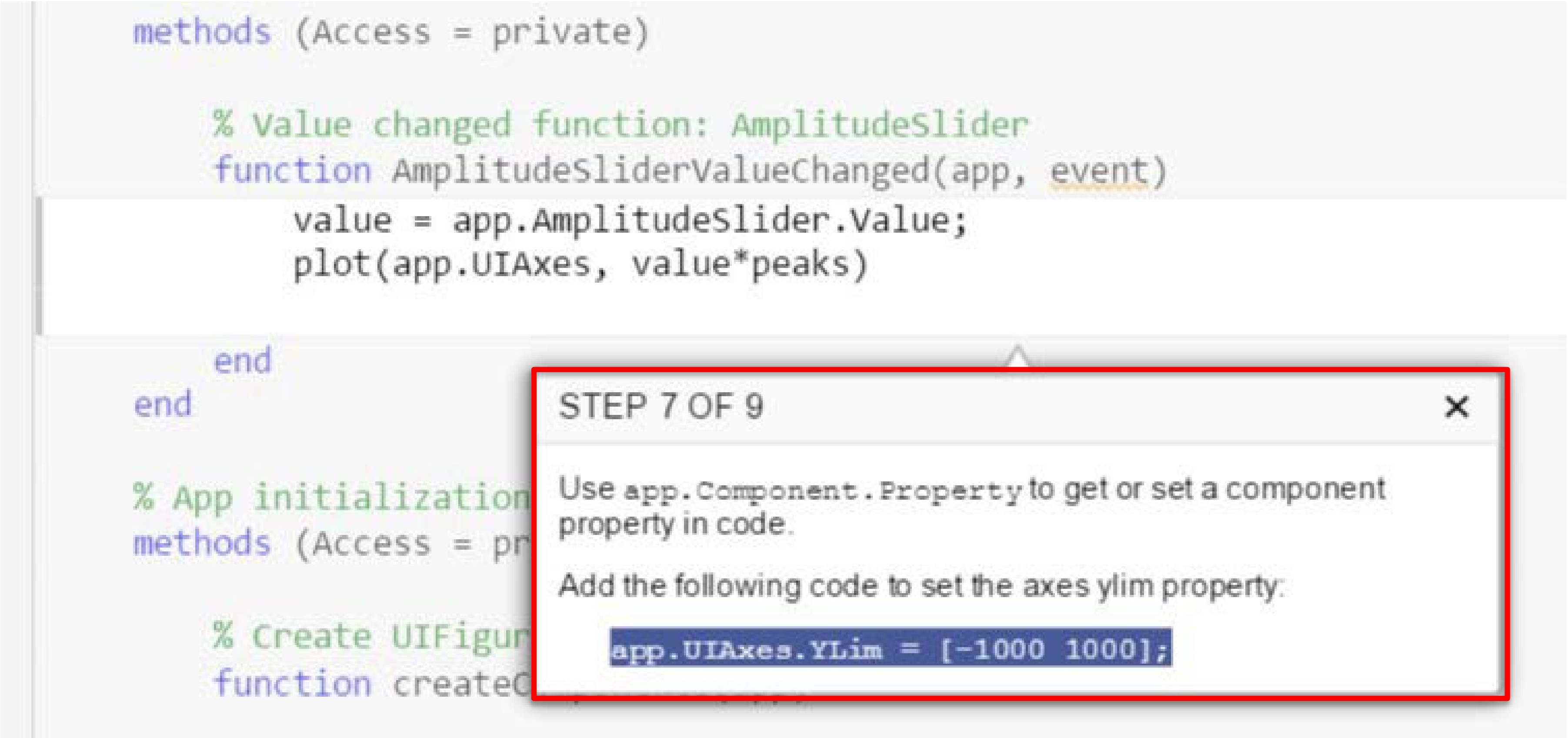
STEP 6 OF 9 ✕

To plot in an axes, you need to use the axes component as the first input argument to the plot command.

Add the following code:

```
plot(app.UIAxes, value*peaks)
```

App Designer – Example



```
methods (Access = private)

    % Value changed function: AmplitudeSlider
    function AmplitudeSliderValueChanged(app, event)
        value = app.AmplitudeSlider.Value;
        plot(app.UIAxes, value*peaks)

    end
end

% App initialization
methods (Access = pr

    % Create UIFigur
    function createC
```

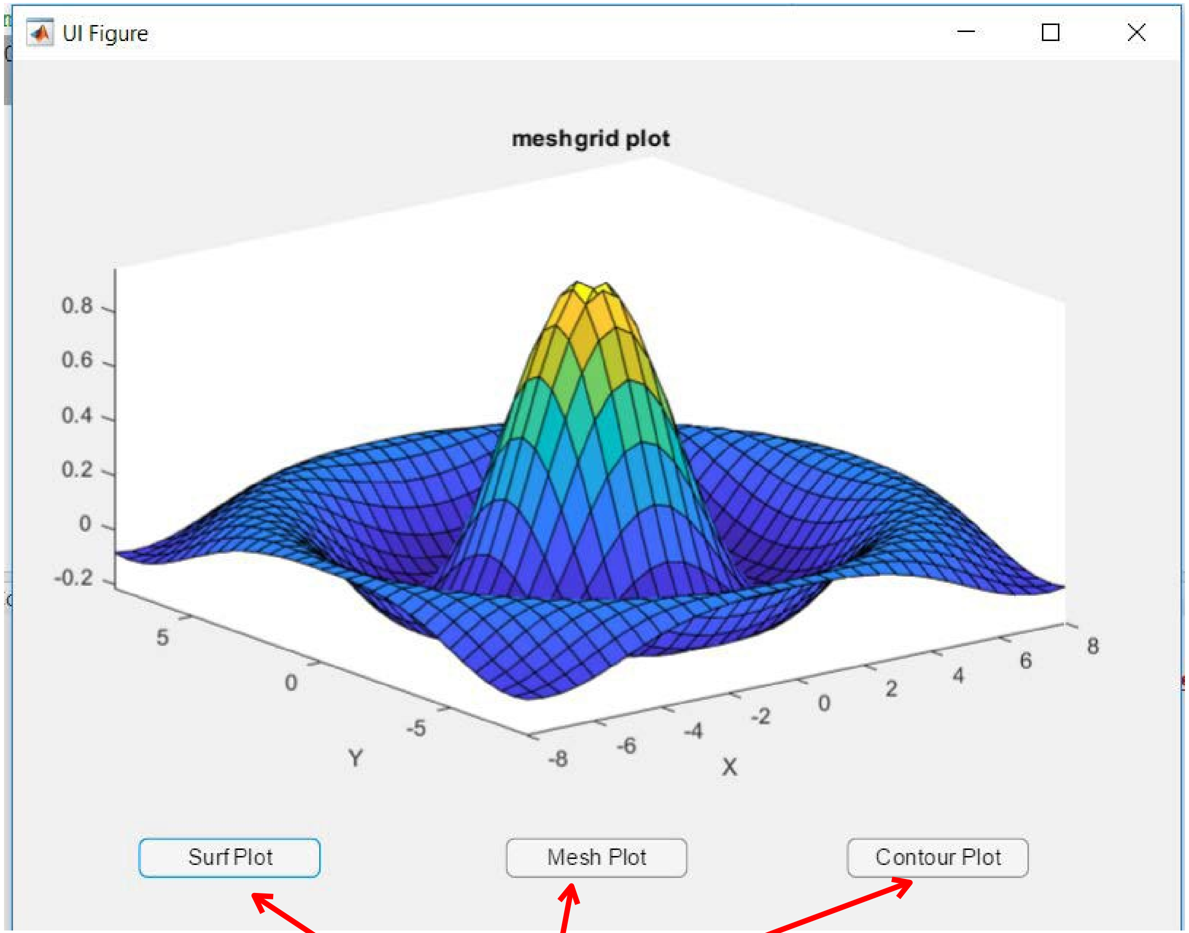
STEP 7 OF 9 ✕

Use `app.Component.Property` to get or set a component property in code.

Add the following code to set the axes ylim property:

```
app.UIAxes.YLim = [-1000 1000];
```

GUI – App Designer Example



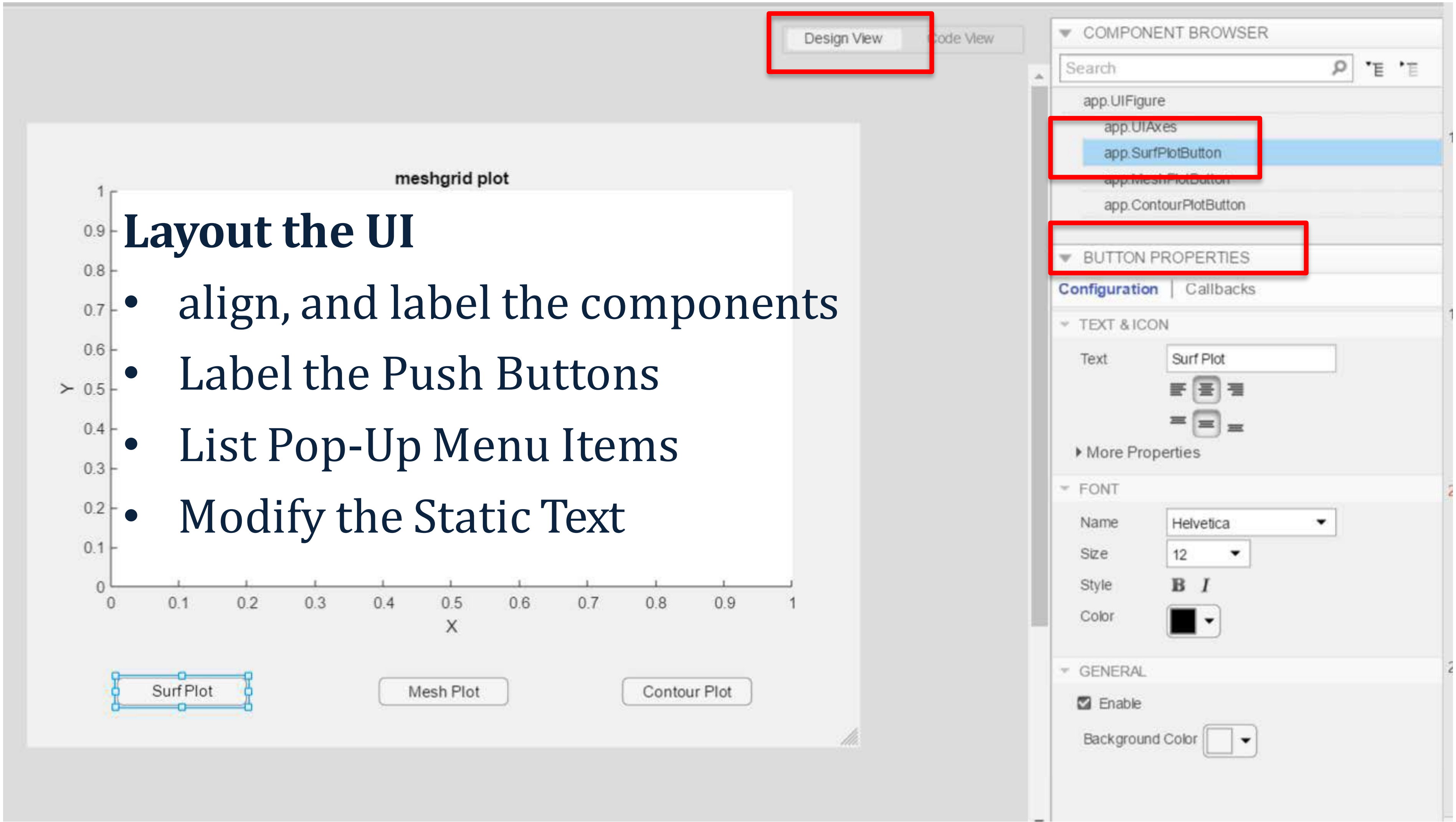
Example code:

```
[x, y] = meshgrid(-8:0.5:8);
r = sqrt(x.^2 + y.^2);
z = sin(r) ./r;
```

buttons

Callbacks for each button to reflect required functionality

Layout the UI



Layout the UI

- align, and label the components
- Label the Push Buttons
- List Pop-Up Menu Items
- Modify the Static Text

Text: Surf Plot

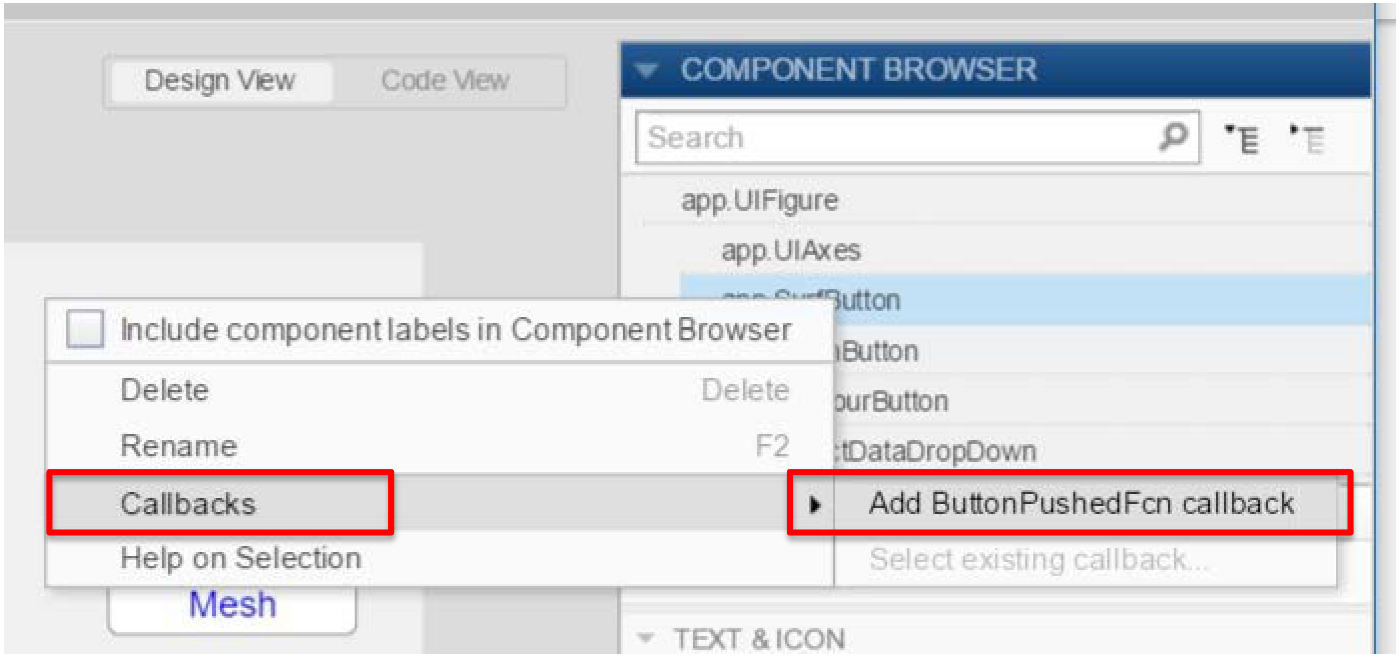
Name: Helvetica

Size: 12

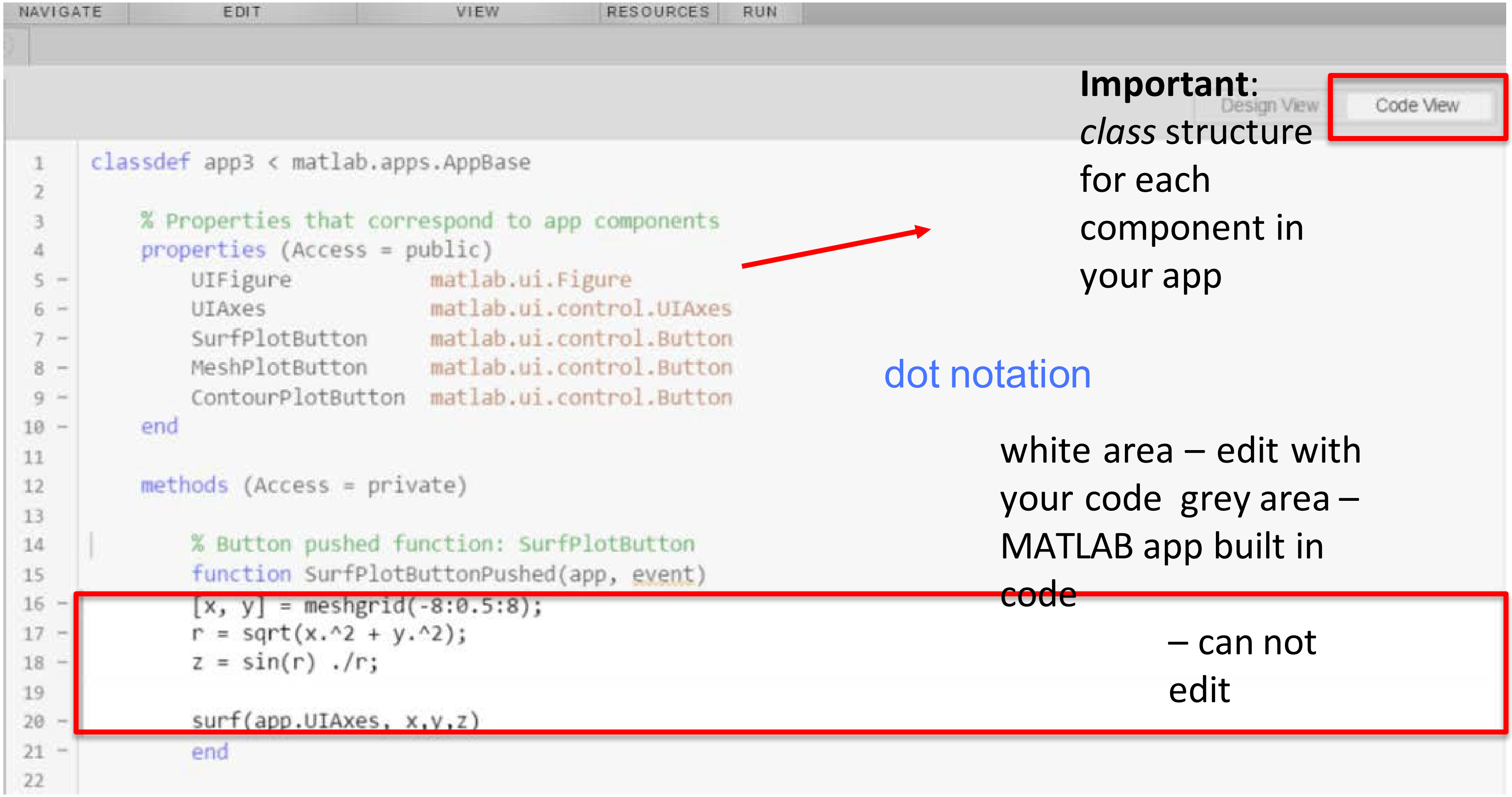
☑ Enable

Adding Callbacks

- A **callback** is a function that executes when a user interacts with a UI component in your app. Most components can have at least one callback.
- When we create a “pushbutton”, we also want to link it with a function that should be executed when the button is clicked.



Adding Callbacks



```
classdef app3 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure           matlab.ui.Figure
        UIAxes             matlab.ui.control.UIAxes
        SurfPlotButton     matlab.ui.control.Button
        MeshPlotButton     matlab.ui.control.Button
        ContourPlotButton  matlab.ui.control.Button
    end

    methods (Access = private)

        % Button pushed function: SurfPlotButton
        function SurfPlotButtonPushed(app, event)
        [x, y] = meshgrid(-8:0.5:8);
        r = sqrt(x.^2 + y.^2);
        z = sin(r) ./r;

        surf(app.UIAxes, x,y,z)
        end
```

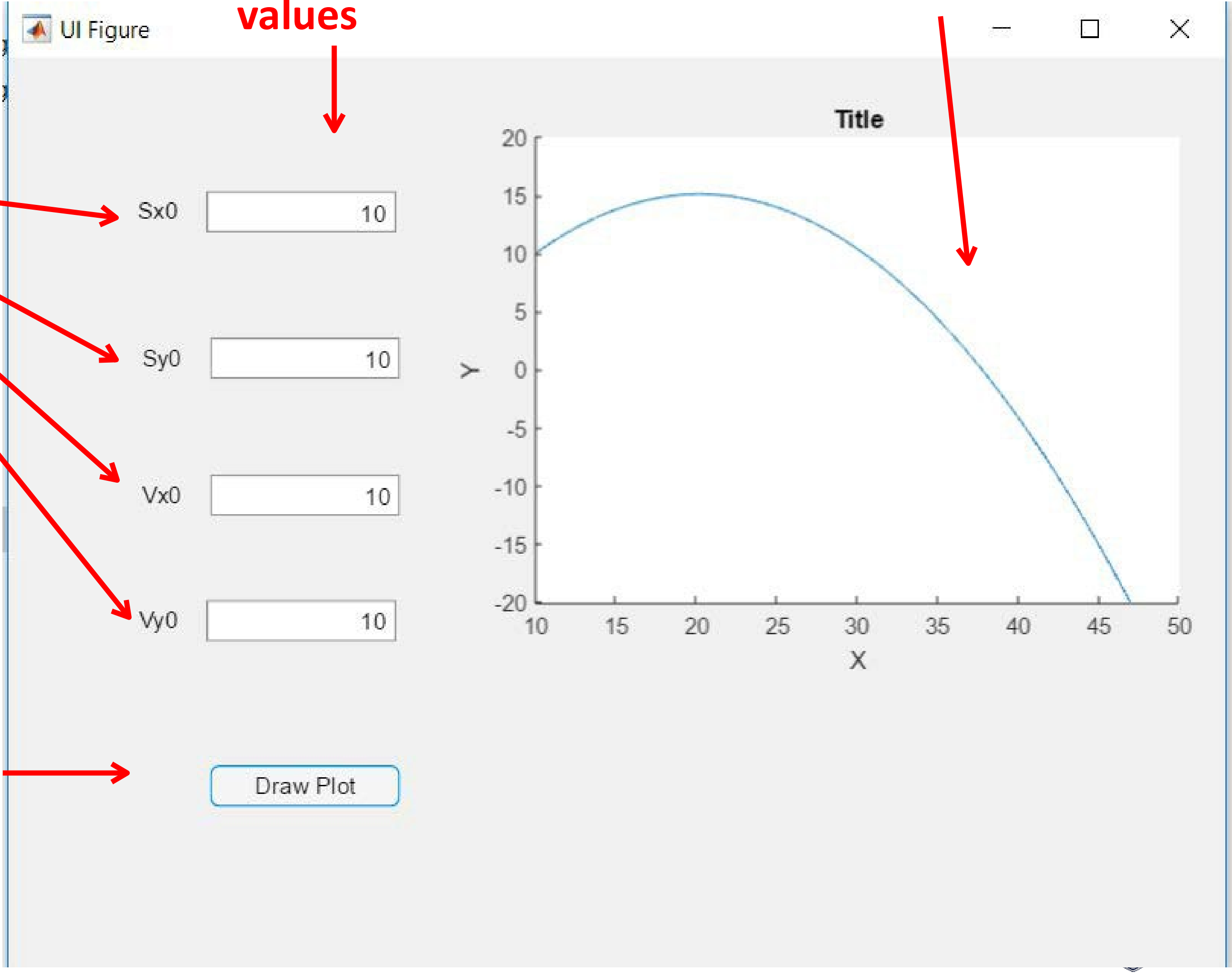
Important: *class* structure for each component in your app

dot notation

white area – edit with your code grey area – MATLAB app built in code

– can not edit

For the lab…



UI Figure

Sx0 10

Sy0 10

Vx0 10

Vy0 10

1. User inputs initial values

2. When this button is pressed

3. Collect these four values

4. Run the projectile motion function and redraw this plot