Coursework Tutorial

Introduction to Computer Vision

ECS709

2021/2022

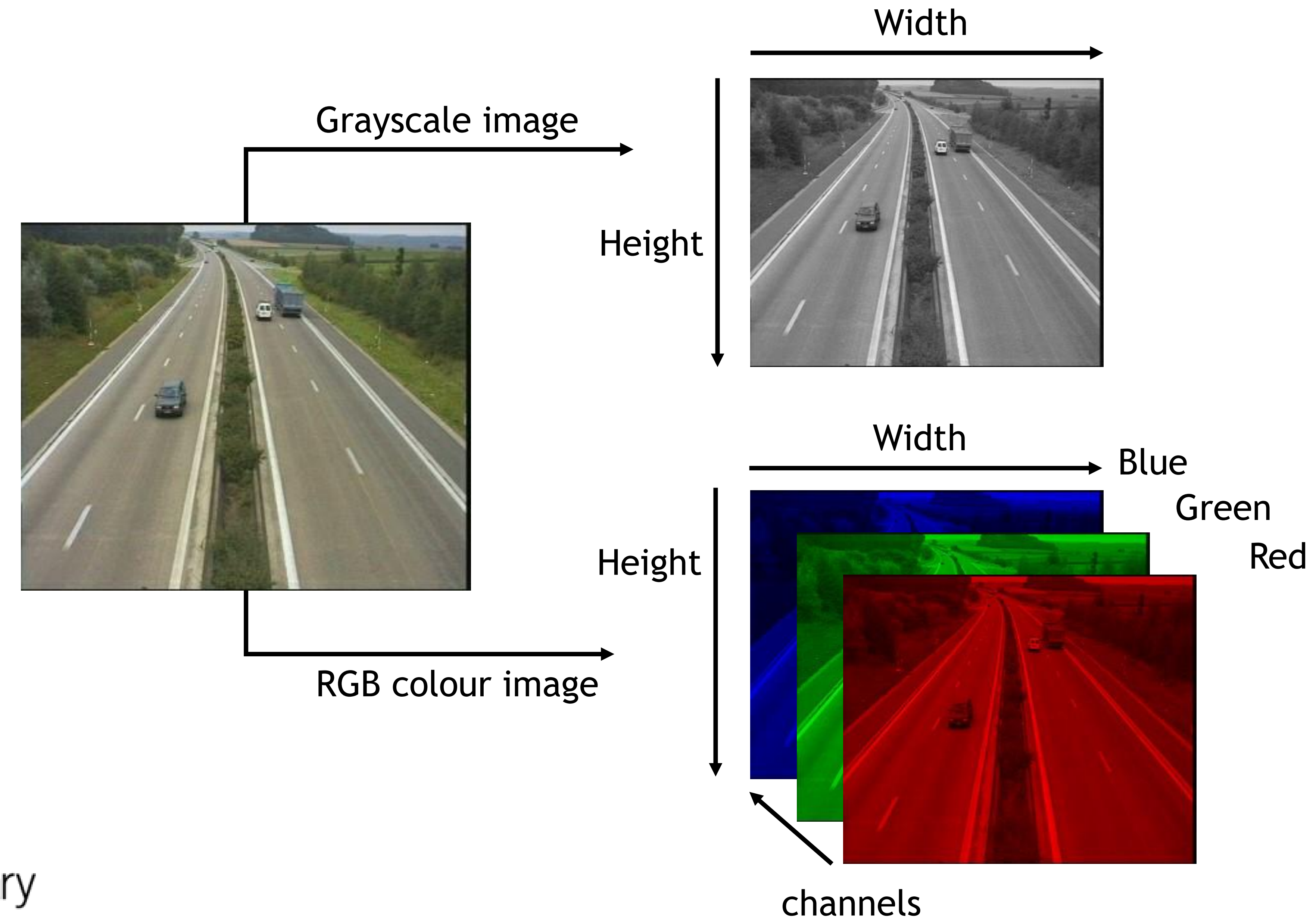
Yik Lung Pang

Queen Mary
University of London

Overview

- Image and video data
- MATLAB introduction
- Coursework requirements
- Introduction to Question 1
- Useful links

Image



Grayscale image

Width

Height

RGB colour image

Width

Height

Blue

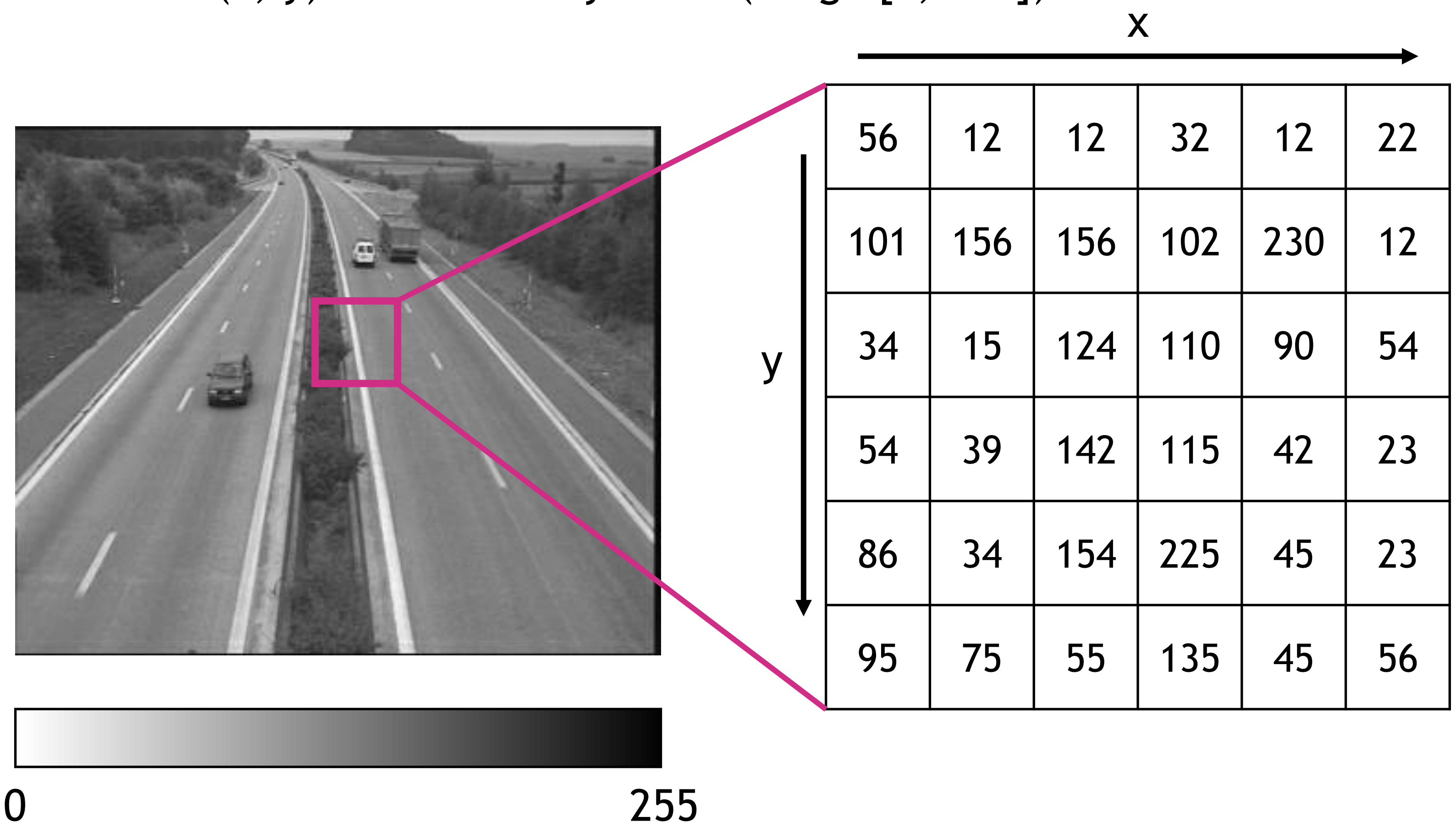
Green

Red

channels

Grayscale Image

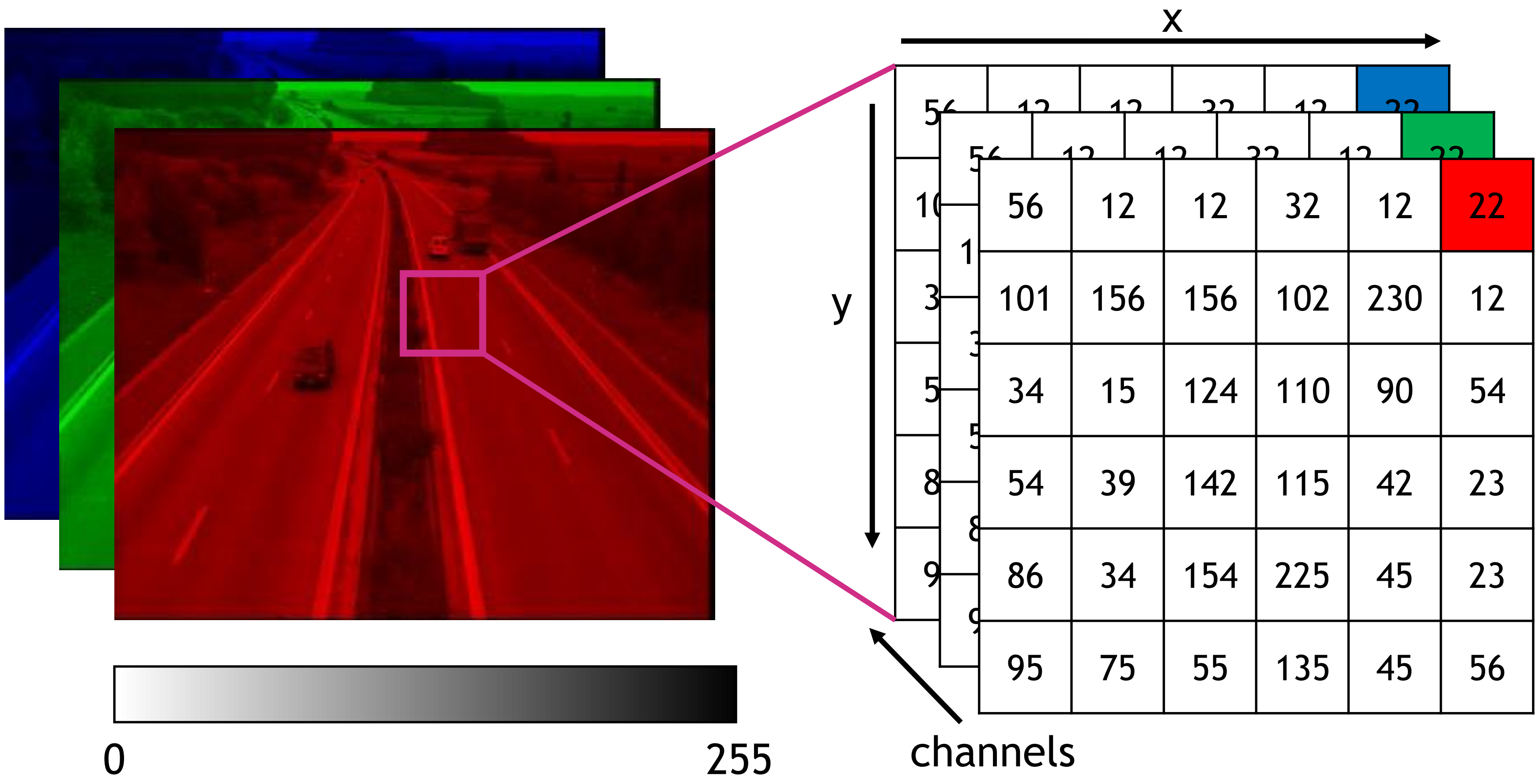
- Image: matrix of pixels
- Pixel: 2D coordinate (x, y) with intensity value (range [0, 255])



56	12	12	32	12	22
101	156	156	102	230	12
34	15	124	110	90	54
54	39	142	115	42	23
86	34	154	225	45	23
95	75	55	135	45	56

Colour Image (RGB)

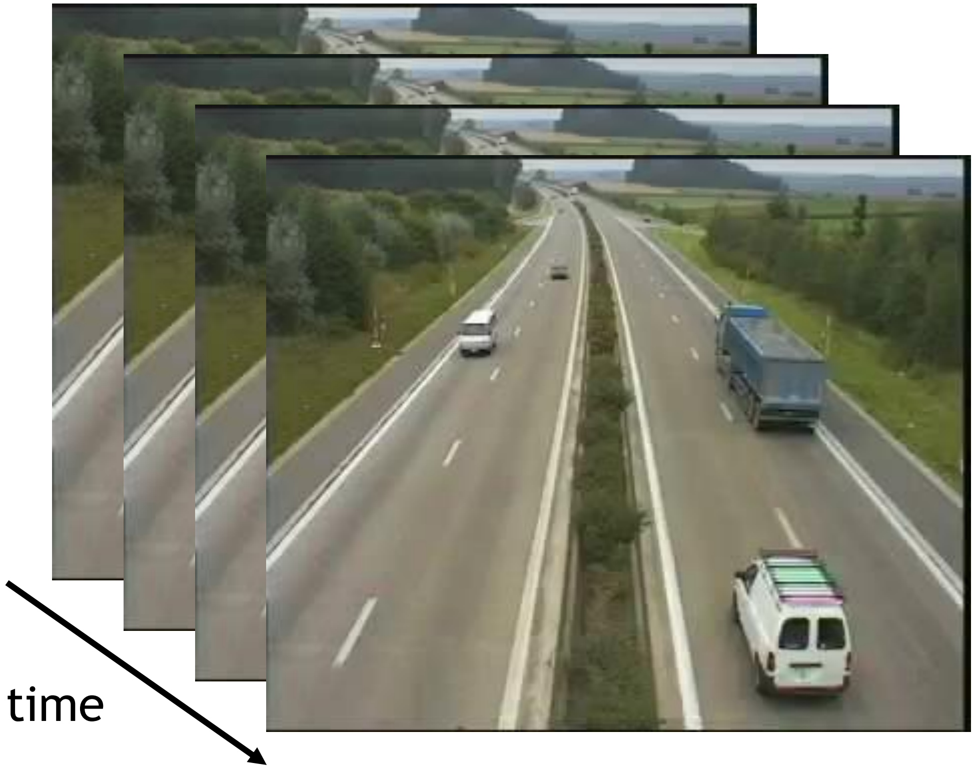
- Image: matrix of pixels
- Pixel: 2D coordinate (x, y) with RGB triplet of intensity value (range [0, 255])



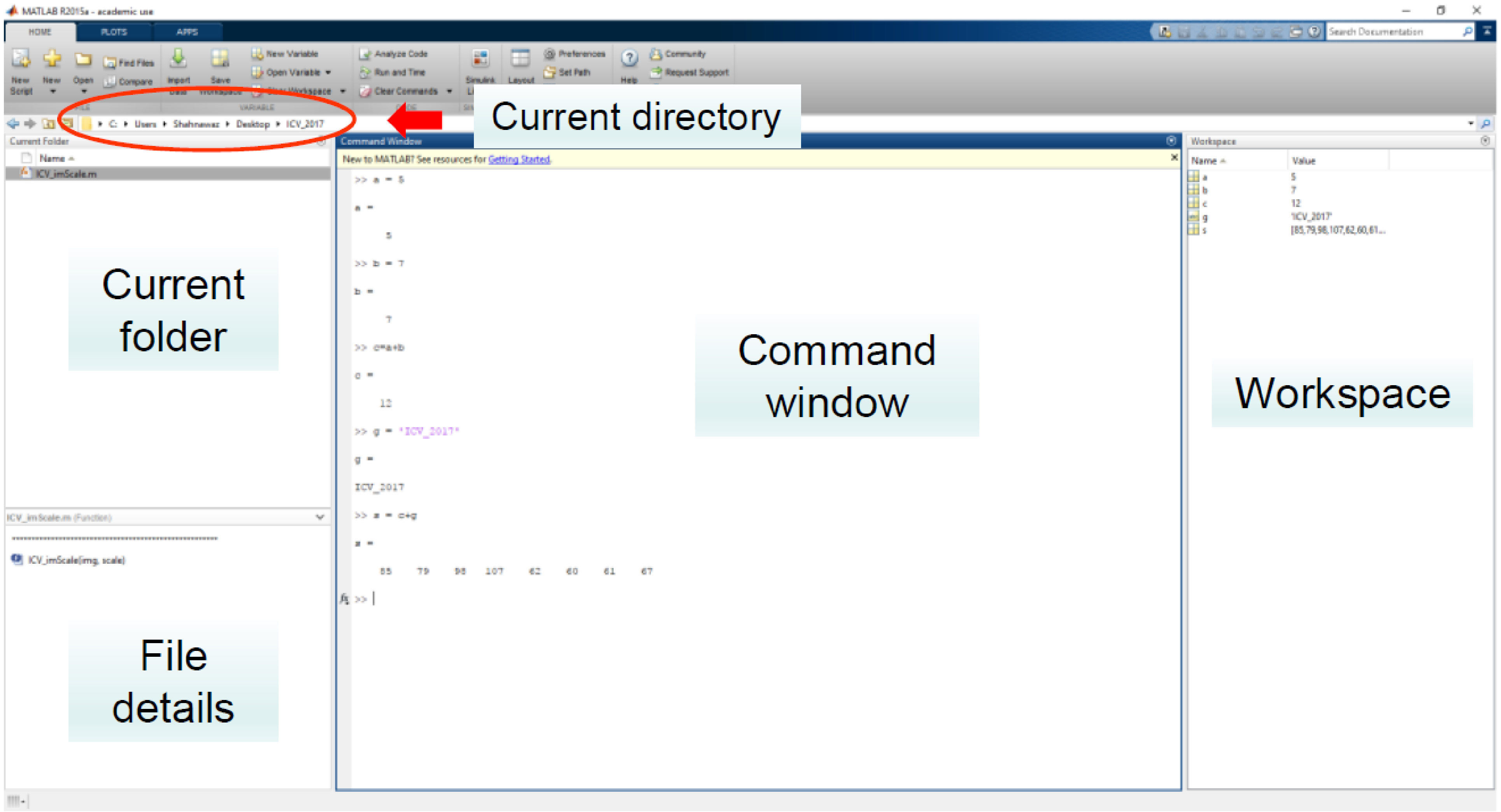
56	12	12	32	12	22
101	156	156	102	230	12
34	15	124	110	90	54
54	39	142	115	42	23
86	34	154	225	45	23
95	75	55	135	45	56

Video

- Collection of images (frames)
- Frame rate (frames per second, fps)



MATLAB introduction



Current directory

C: ▸ Users ▸ Shahnawaz ▸ Desktop ▸ ICV_2017

Current folder

ICV_imScale.m

File details

ICV_imScale.m (Function)

ICV_imScale(img, scale)

Command window

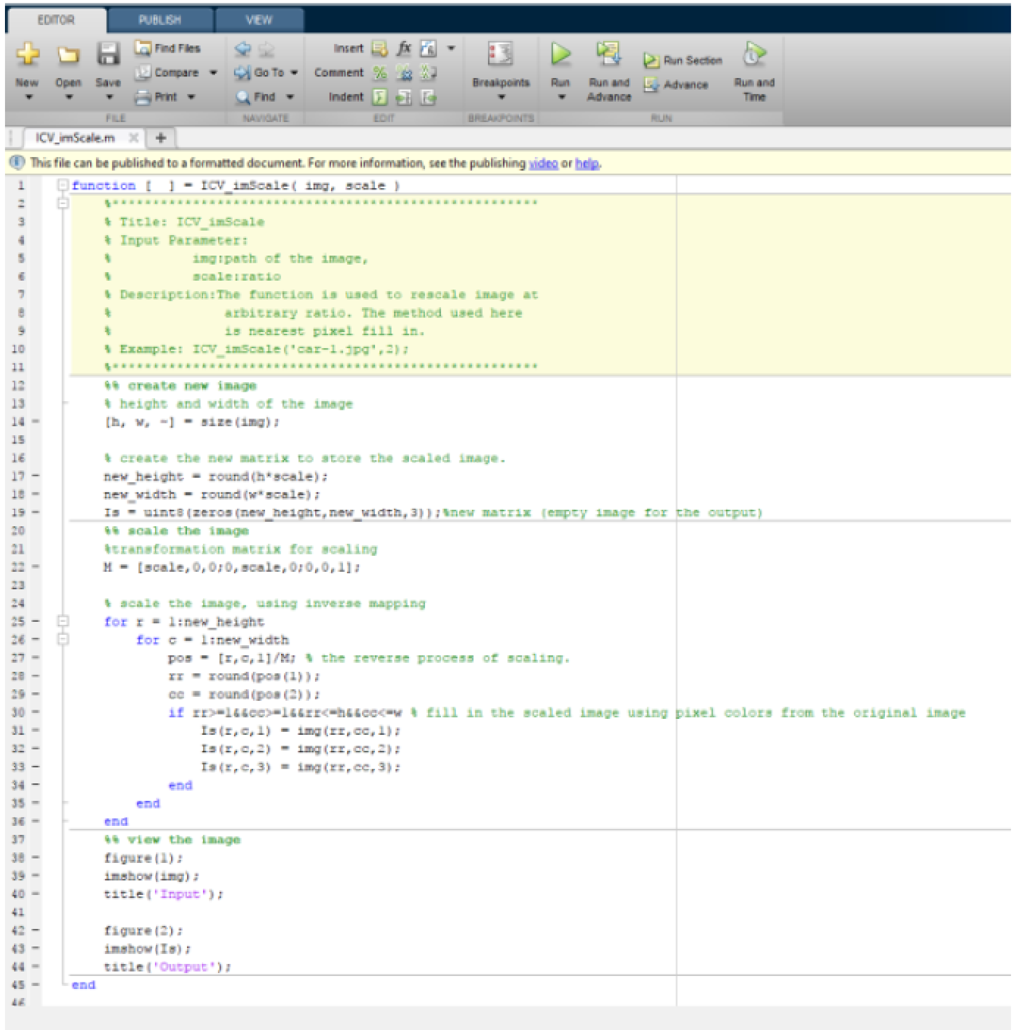
New to MATLAB? See resources for Getting Started.

```
>> a = 5

a =

     5

>> b = 7

b =

     7

>> c=a+b

c =

    12

>> g = 'ICV_2017'

g =

ICV_2017

>> s = c+g

s =

    85    79    98   107    62    60    61    67

>> |
```

Workspace

Name	Value
a	5
b	7
c	12
g	'ICV_2017'
s	[85,79,98,107,62,60,61...

Editor



This file can be published to a formatted document. For more information, see the publishing video or help.

```
function [   ] = ICV_imScale( img, scale )
    %*******************************************************
    % Title: ICV_imScale
    % Input Parameter:
    %           img:path of the image,
    %           scale:ratio
    % Description:The function is used to rescale image at
    %               arbitrary ratio. The method used here
    %               is nearest pixel fill in.
    % Example: ICV_imScale('car-1.jpg',2);
    %*******************************************************
    %% create new image
    % height and width of the image
    [h, w, ~] = size(img);

    % create the new matrix to store the scaled image.
    new_height = round(h*scale);
    new_width = round(w*scale);
    Is = uint8(zeros(new_height,new_width,3));%new matrix (empty image for the output)
    %% scale the image
    %transformation matrix for scaling
    M = [scale,0,0;0,scale,0;0,0,1];

    % scale the image, using inverse mapping
    for r = 1:new_height
        for c = 1:new_width
            pos = [r,c,1]/M; % the reverse process of scaling.
            rr = round(pos(1));
            cc = round(pos(2));
            if rr>=1&&cc>=1&&rr<=h&&cc<=w % fill in the scaled image using pixel colors from the original image
                Is(r,c,1) = img(rr,cc,1);
                Is(r,c,2) = img(rr,cc,2);
                Is(r,c,3) = img(rr,cc,3);
            end
        end
    end
    %% view the image
    figure(1);
    imshow(img);
    title('Input');

    figure(2);
    imshow(Is);
    title('Output');
end
```

Write and run scripts or functions

Creating array and matrices

- Scalar
  - *c = 5;*

- Array (e.g. 4D vector)
  - row vector:
  - *myArray = [1 2 3 4];* or *myArray = [1,2,3,4];*
  - column vector:
  - *myArray = [1 2 3 4]';* or *myArray = [1;2;3;4];*

- Matrix (e.g. 3x3)
  - *myMatrix = [1 2 3; 4 5 6; 7 8 10];*

Creating array and matrices

- Create a 3x2 matrix of zeros
  - *A = zeros(3,2)*

$$A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

- Create a 2x4 matrix of ones
  - *B = ones(2,4)*

$$\mathrm{B} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

- Create a 3x3 identity matrix
  - *E = eye(3)*

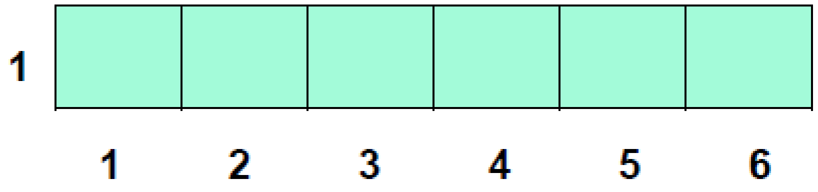
$$\mathrm{E} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Create a 3x4 matrix of uniformly distributed random numbers
  - *R = rand(3)*

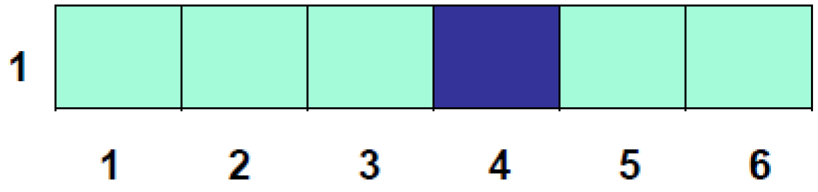
$$\mathrm{R} = \begin{bmatrix} 0.8147 & 0.0975 & 0.1576 \\ 0.1419 & 0.6557 & 0.9058 \\ 0.2785 & 0.9706 & 0.4218 \end{bmatrix}$$

Array indexing (1D)

- How to access to an element of an array
  - *elem = myArray(pos_n);*



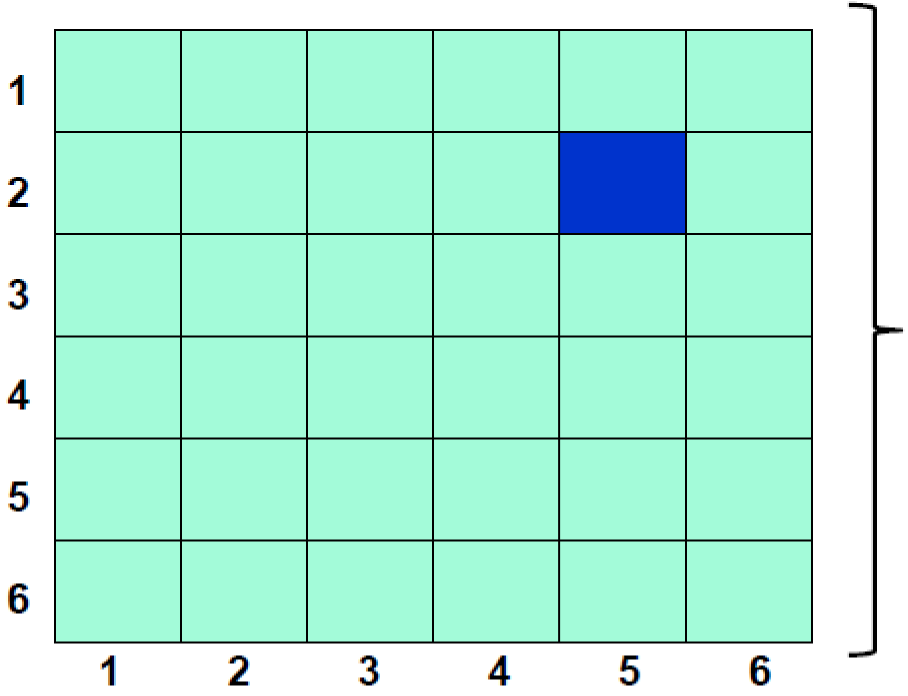
- Example: fourth position
  - *elem = myArray(4);*



<u>1-index</u> based language
(MATLAB)
vs
0-index based languages
(e.g. C or Python)

Array indexing (2D)

- How to access to an element of a matrix
  - *elem = myMatrix(mRow, mCol);*



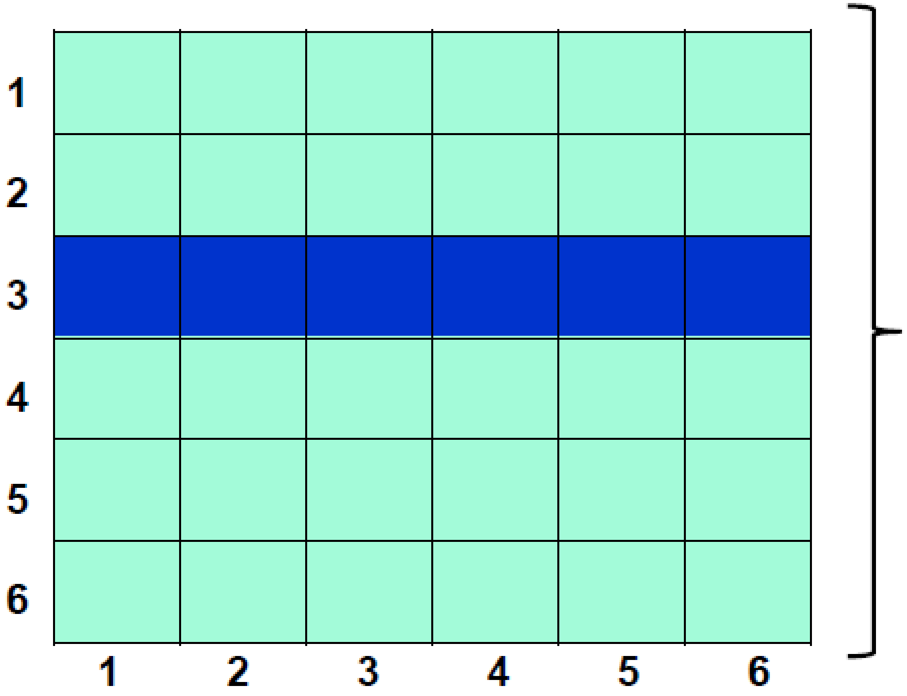
"myMatrix"

Accessing any element:
myMatrix(2, 5)

Array indexing (2D)

Colon operator -> ‘:`

- Extract a row
  - *elem = myMatrix(mRow, :);*



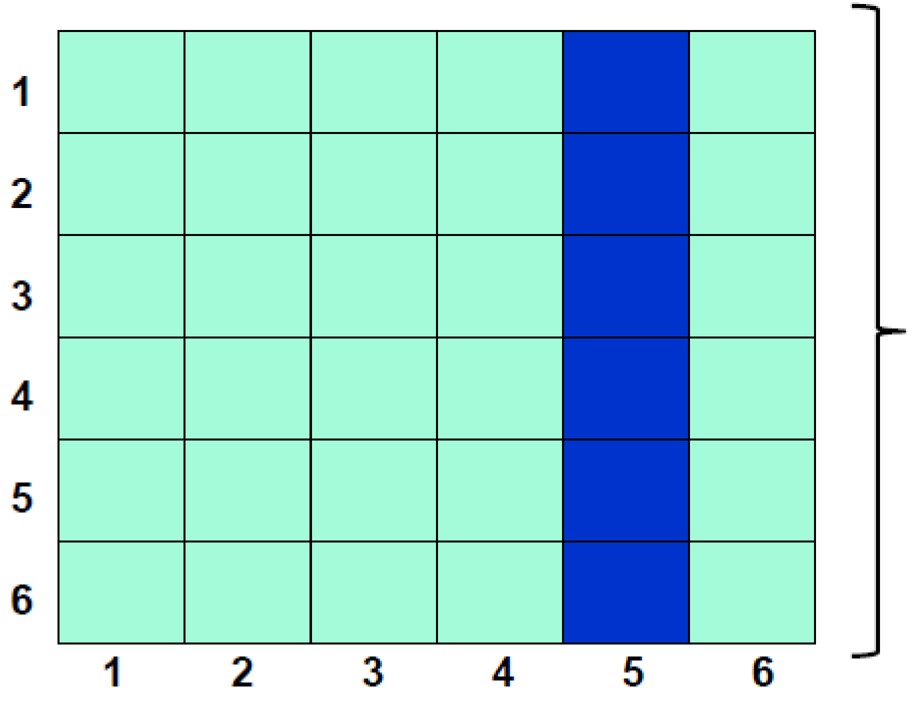
“myMatrix”

Accessing any element:
myMatrix(3, :)

Array indexing (2D)

Colon operator -> ‘:`

- Extract a column
  - *elem = myMatrix(:, mCol);*



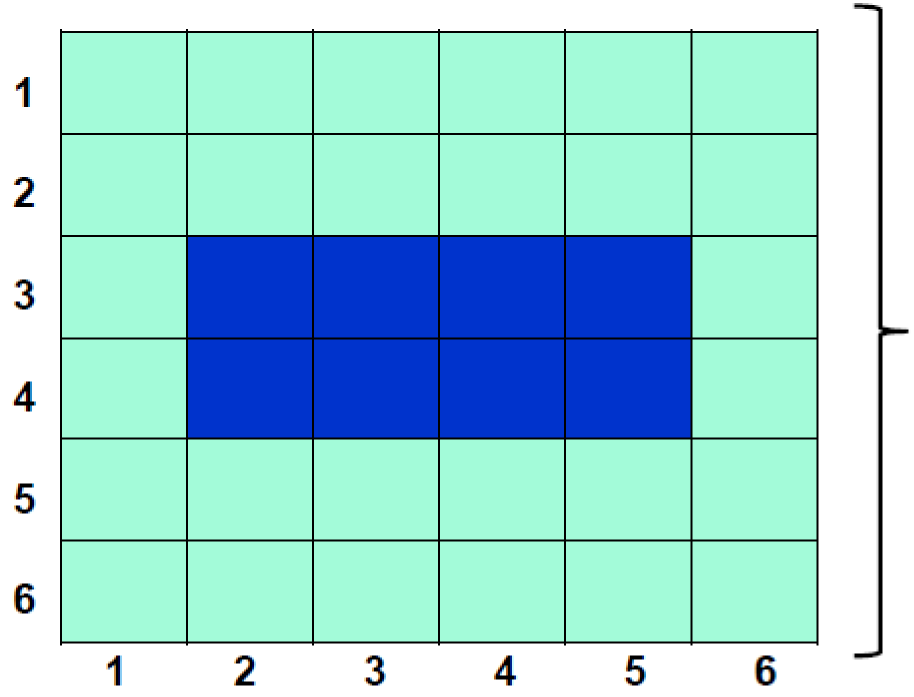
“myMatrix”

Accessing any element:
myMatrix(:, 5)

Array indexing (2D)

Colon operator -> ‘:`

- Extract sub-matrix
  - *elem = myMatrix(mRow1:mRow2, mCol1:mCol2);*

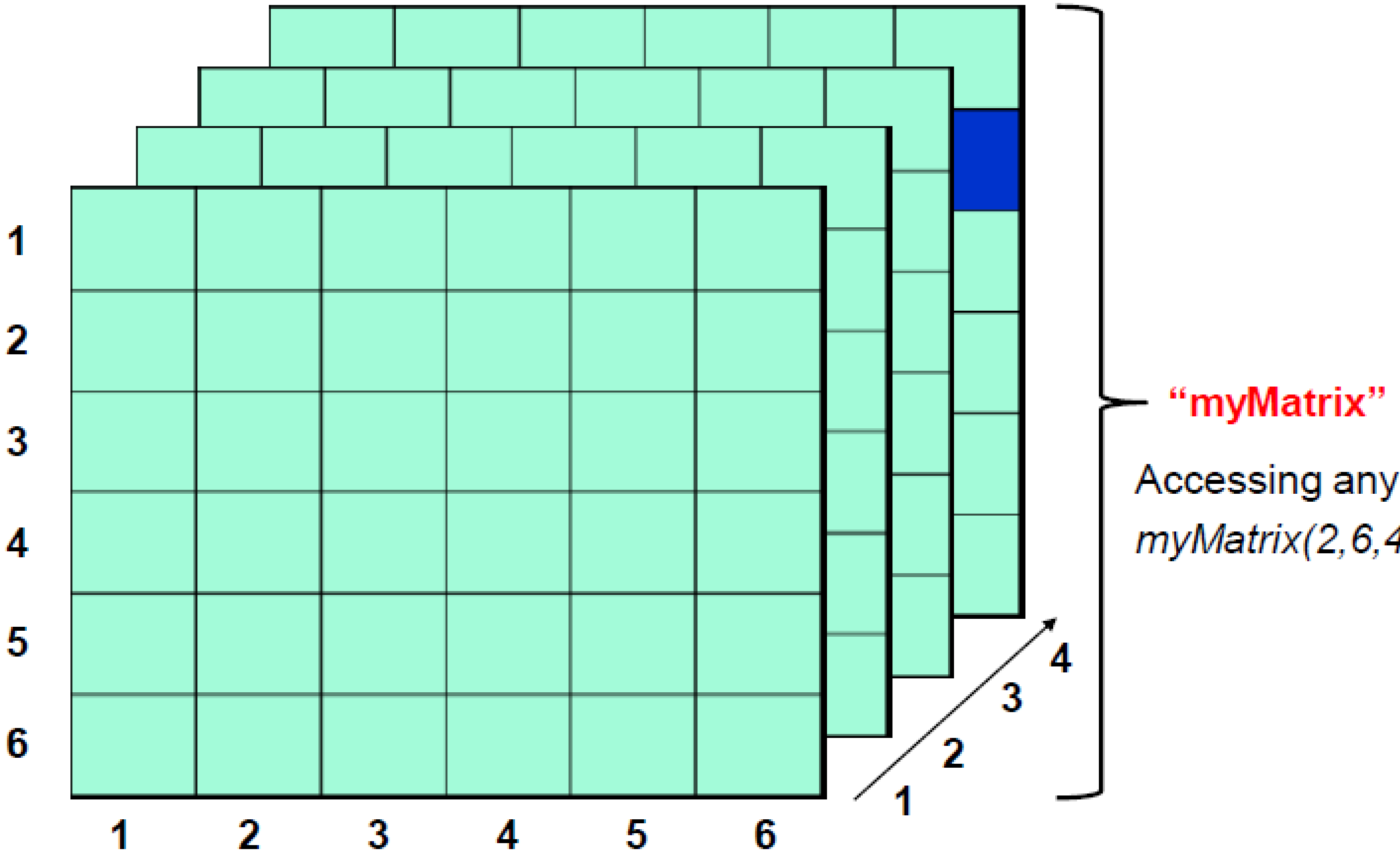


“myMatrix”

Accessing any element:
myMatrix(3:4, 2:5)

Array indexing (3D)

- How to access to an element of a multi-dimensional array
  - *elem = myMatrix(mRow, mCol, mThird);*

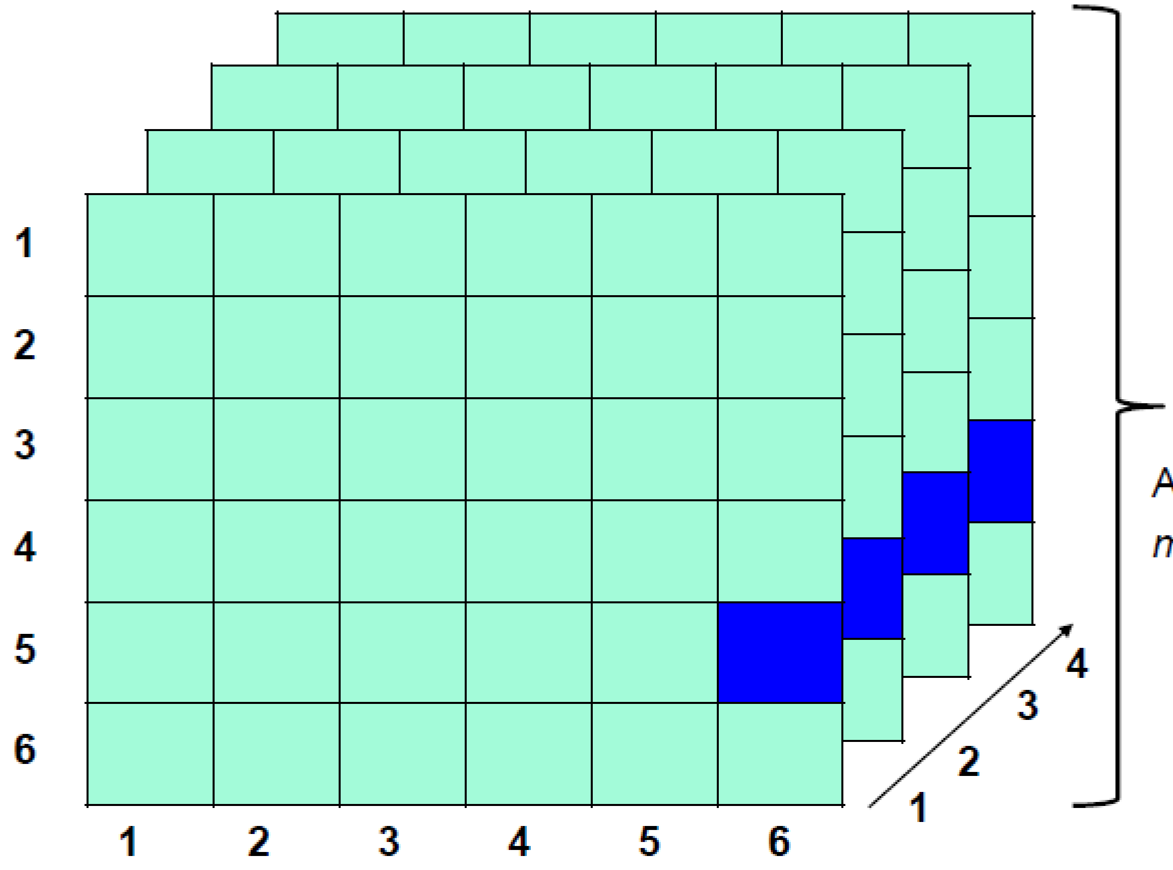


Accessing any element:

myMatrix(2,6,4)

Array indexing (3D)

- How to access to an element of a multi-dimensional array
  - *elem = myMatrix(mRow, mCol, mThird);*



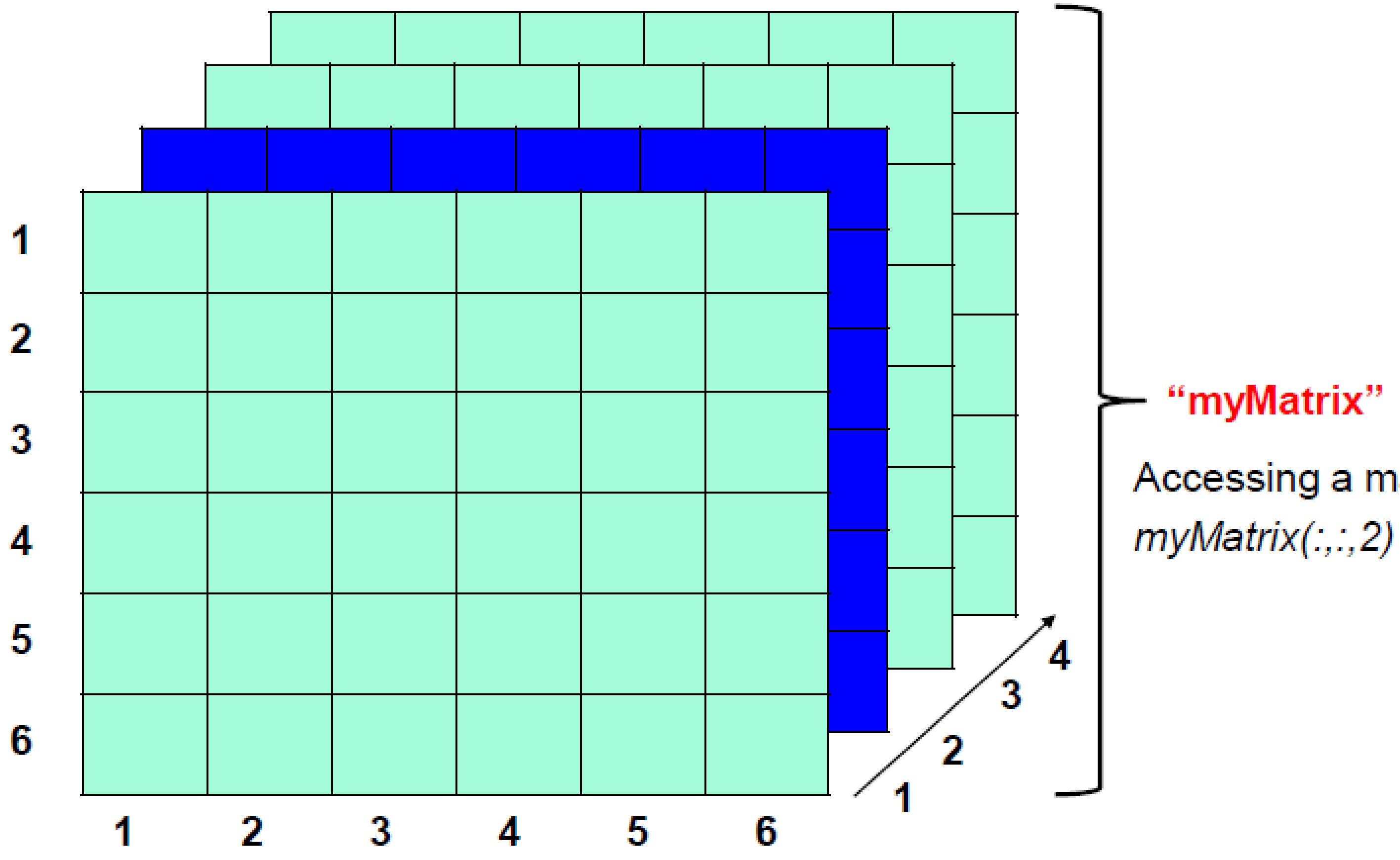
"myMatrix"

Accessing a vector:

myMatrix(5,6,:)

Array indexing (3D)

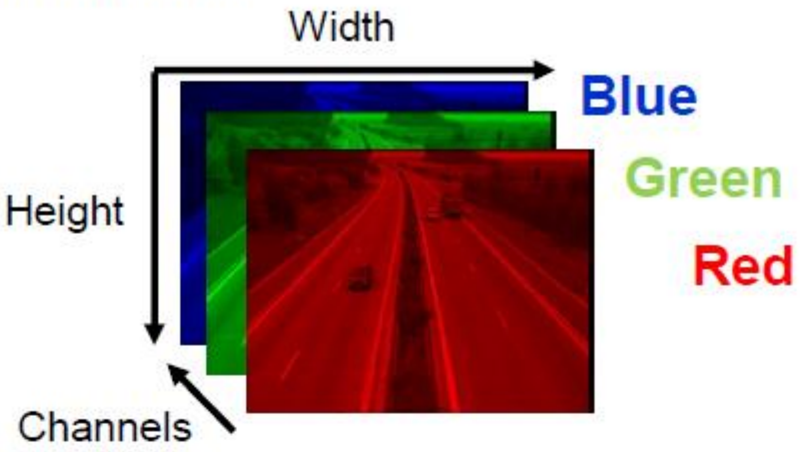
- How to access to an element of a multi-dimensional array
  - *elem = myMatrix(mRow, mCol, mThird);*



Accessing a matrix:
myMatrix(:,:,2)

Array indexing (example)

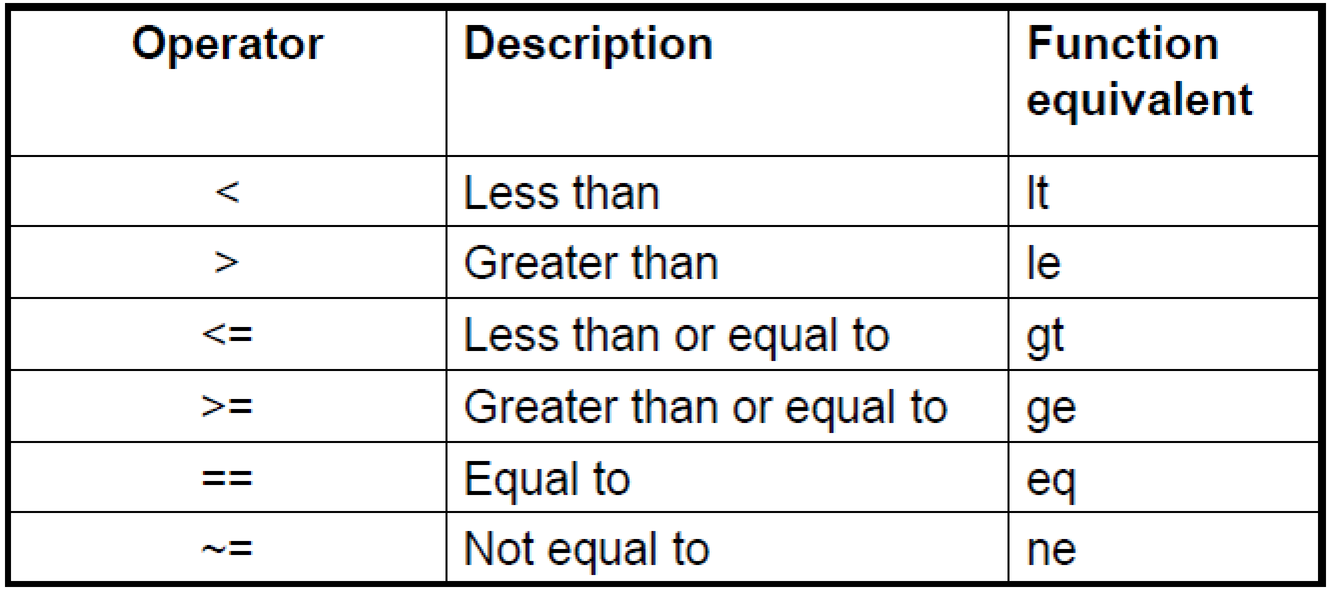
- All elements in individual channels
  - TEST(: , : , 1) Red
  - TEST(: , : , 2) Green
  - TEST(: , : , 3) Blue



- First element in

– Red channel:	TEST(1, 1, 1)
– Green channel:	TEST(1, 1, 2)
– Blue channel:	TEST(1, 1, 3)
– All channels:	TEST(1, 1, :)

Relational operation



Operator	Description	Function equivalent
<	Less than	lt
>	Greater than	le
<=	Less than or equal to	gt
>=	Greater than or equal to	ge
==	Equal to	eq
~=	Not equal to	ne

Example:

- *A = [2 4 6; 8 10 12]* ➔ $\begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}$
- *B = [5 5 5; 9 9 9]* ➔ $\begin{bmatrix} 5 & 5 & 5 \\ 9 & 9 & 9 \end{bmatrix}$
- A < B ➔ $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$

Relational operation

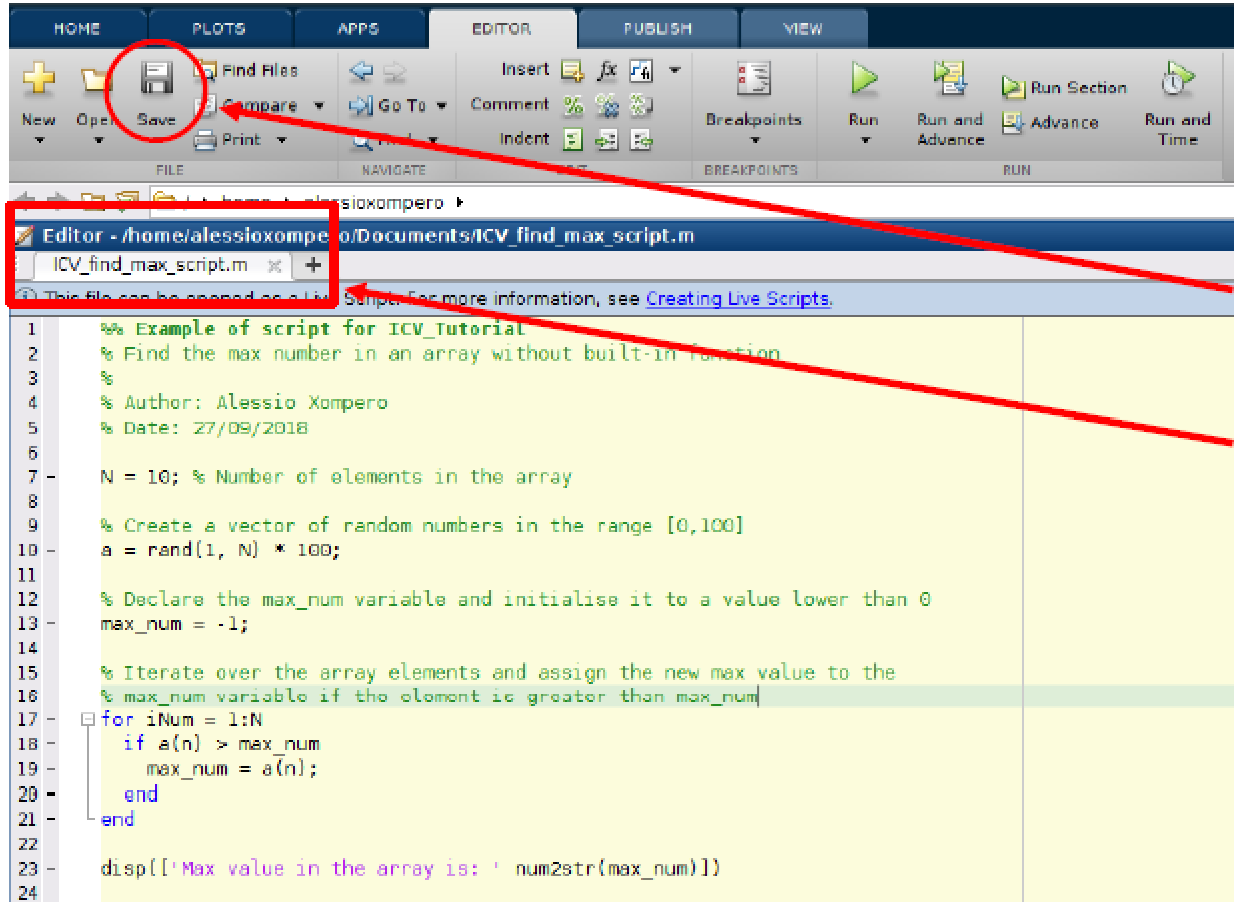
Operator	Description	Function equivalent
&	Logical AND	and
|	Logical OR	or
~	Logical NOT	not
xor	Logical exclusive-OR	xor
true	Return logical 1 (true)	true
false	Return logical 0 (false)	false

Example:

- *A = [2 -1; -3 10]* ➔ $\begin{bmatrix} 2 & -1 \\ -3 & 10 \end{bmatrix}$
- *B = [0 -2; -3 -1]* ➔ $\begin{bmatrix} 0 & -2 \\ -3 & -1 \end{bmatrix}$
- A<0 & B<0 ➔ $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

Scripts

- Automating sequence of instructions (MATLAB commands)



Editor - /home/alessioxompero/Documents/ICV_find_max_script.m

This file can be opened as a Live Script. For more information, see Creating Live Scripts.

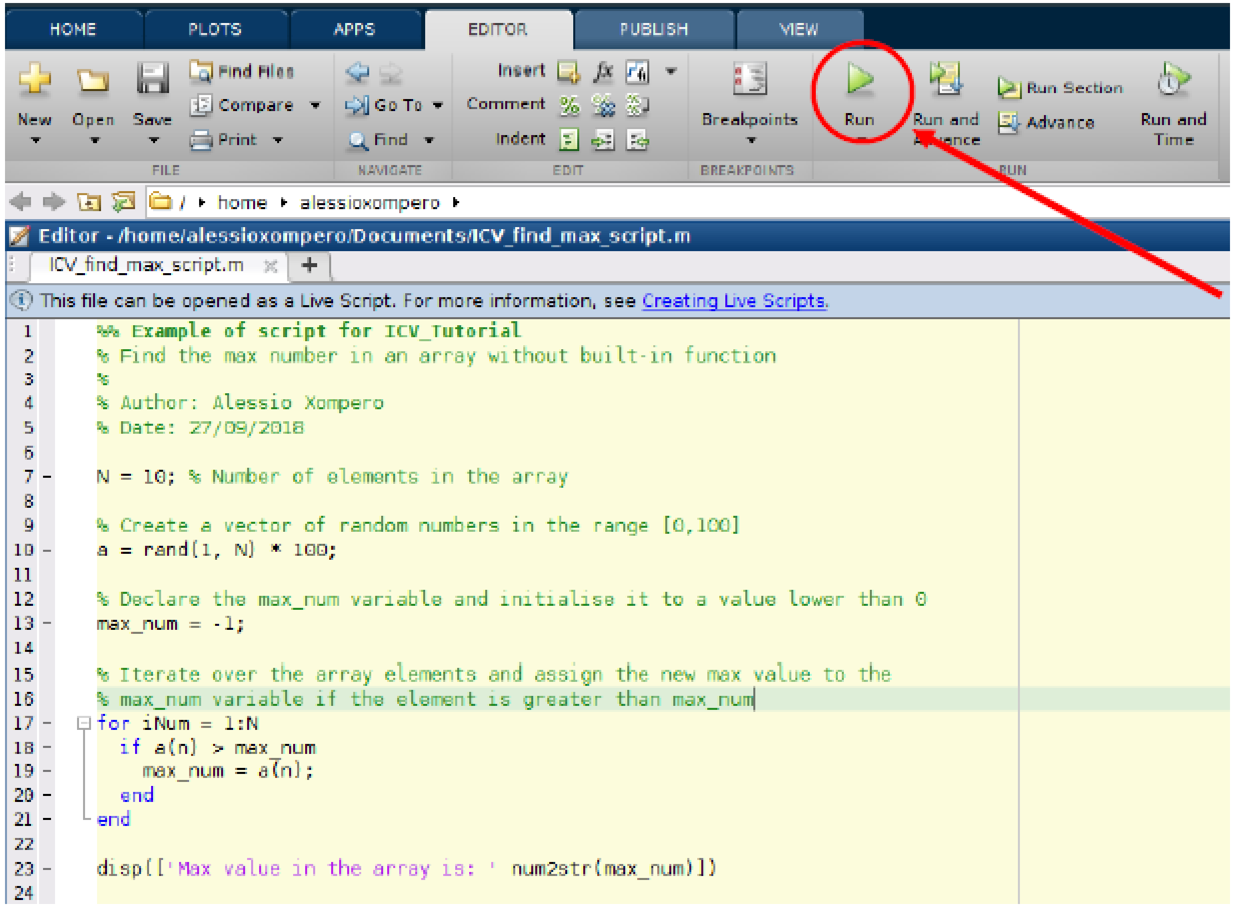
```matlab
%% Example of script for ICV_Tutorial
% Find the max number in an array without built-in function
%
% Author: Alessio Xompero
% Date: 27/09/2018

N = 10; % Number of elements in the array

% Create a vector of random numbers in the range [0,100]
a = rand(1, N) * 100;

% Declare the max_num variable and initialise it to a value lower than 0
max_num = -1;

% Iterate over the array elements and assign the new max value to the
% max_num variable if the element is greater than max_num
for iNum = 1:N
  if a(n) > max_num
    max_num = a(n);
  end
end

disp(['Max value in the array is: ' num2str(max_num)])

```

Save the script
(or press Ctrl + S)

Remember to use
.m as extension

Scripts

- Automating sequence of instructions (MATLAB commands)



This file can be opened as a Live Script. For more information, see Creating Live Scripts.

```matlab
%% Example of script for ICV_Tutorial
% Find the max number in an array without built-in function
%
% Author: Alessio Xompero
% Date: 27/09/2018

N = 10; % Number of elements in the array

% Create a vector of random numbers in the range [0,100]
a = rand(1, N) * 100;

% Declare the max_num variable and initialise it to a value lower than 0
max_num = -1;

% Iterate over the array elements and assign the new max value to the
% max_num variable if the element is greater than max_num
for iNum = 1:N
  if a(n) > max_num
    max_num = a(n);
  end
end

disp(['Max value in the array is: ' num2str(max_num)])
```

Run the script
(or press F5)

Functions

- Automating sequence of instructions (like scripts)

- Flexible, reusable and extendible

- Syntax

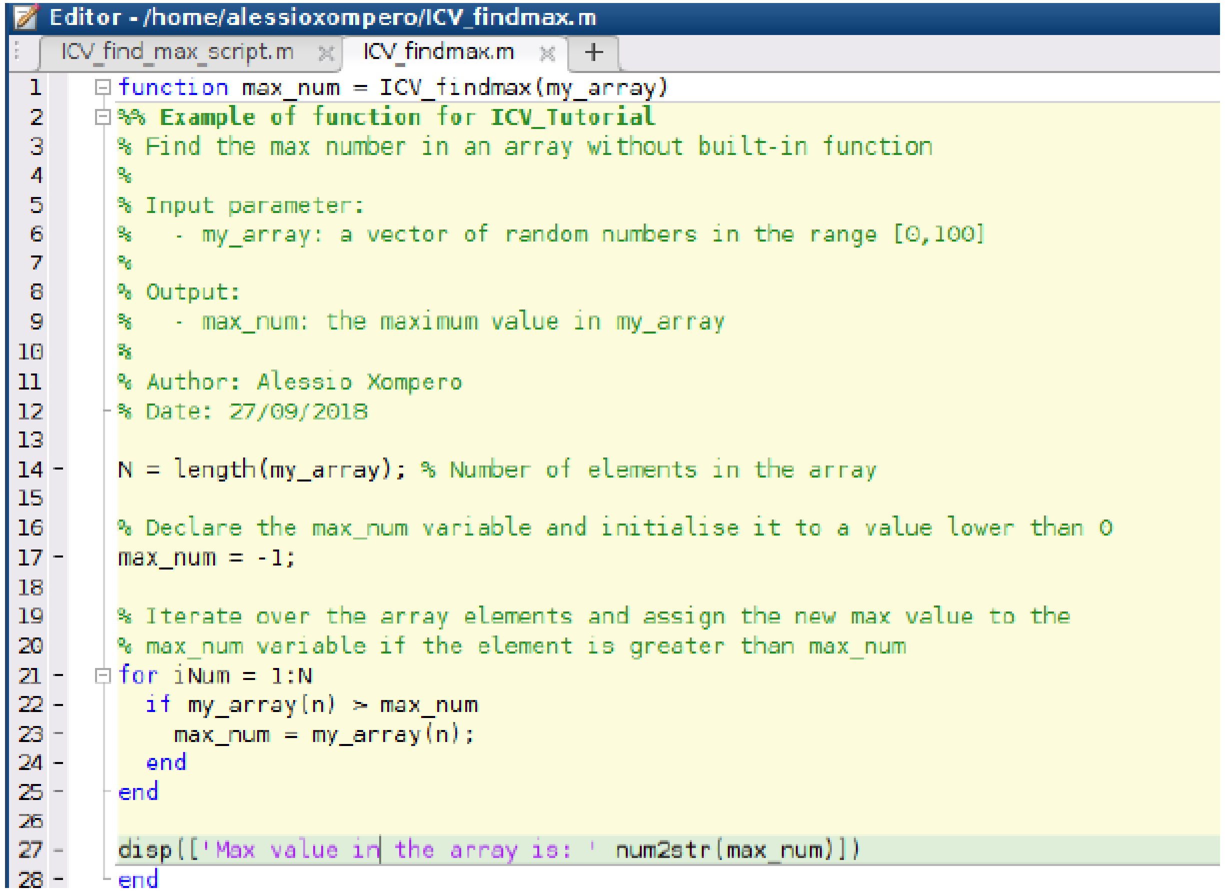
  function [*out1*, *out2*] = *function_name(input_1, …, input_n2)*

  statements

  end

- Important notes:
  - Function name = filename
  - avoid keywords and inbuild function names

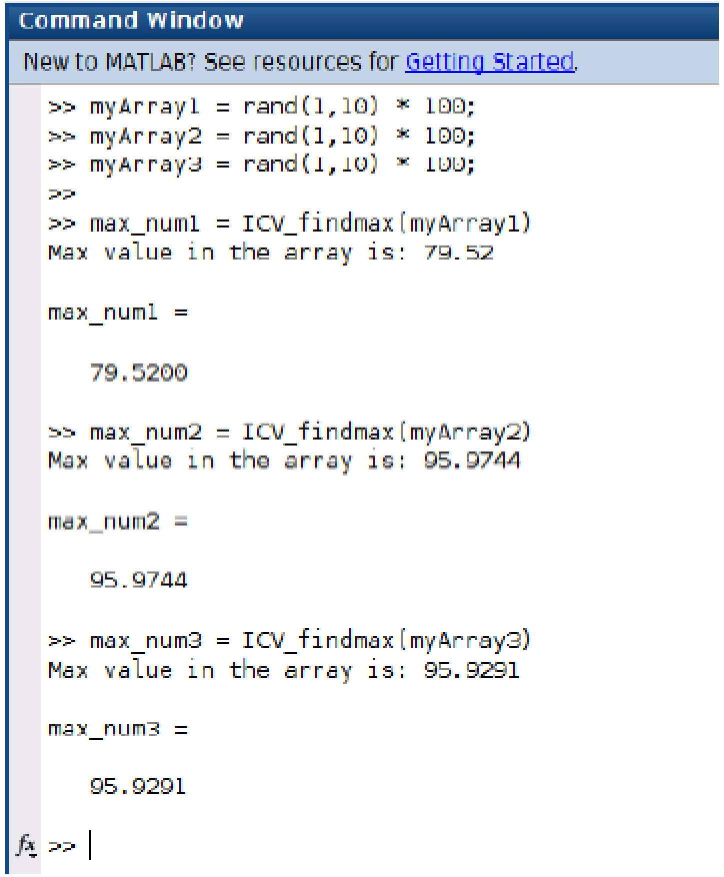
Functions



```
function max_num = ICV_findmax(my_array)
%% Example of function for ICV_Tutorial
% Find the max number in an array without built-in function
%
% Input parameter:
%   - my_array: a vector of random numbers in the range [0,100]
%
% Output:
%   - max_num: the maximum value in my_array
%
% Author: Alessio Xompero
% Date: 27/09/2018

N = length(my_array); % Number of elements in the array

% Declare the max_num variable and initialise it to a value lower than 0
max_num = -1;

% Iterate over the array elements and assign the new max value to the
% max_num variable if the element is greater than max_num
for iNum = 1:N
  if my_array(n) > max_num
    max_num = my_array(n);
  end
end

disp(['Max value in the array is: ' num2str(max_num)])
end
```

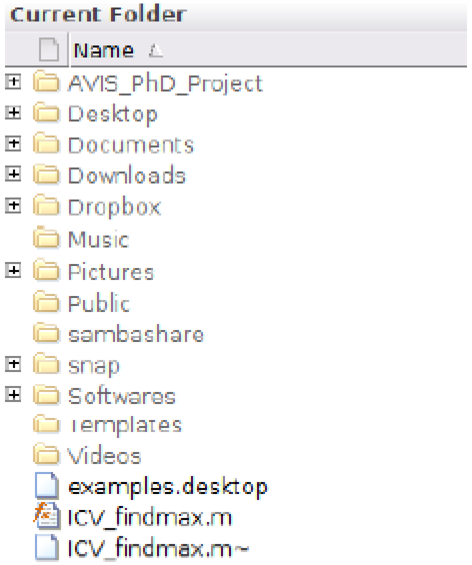
Functions

- Calling the function



New to MATLAB? See resources for Getting Started.

```
>> myArray1 = rand(1,10) * 100;
>> myArray2 = rand(1,10) * 100;
>> myArray3 = rand(1,10) * 100;
>>
>> max_num1 = ICV_findmax(myArray1)
Max value in the array is: 79.52

max_num1 =

   79.5200

>> max_num2 = ICV_findmax(myArray2)
Max value in the array is: 95.9744

max_num2 =

   95.9744

>> max_num3 = ICV_findmax(myArray3)
Max value in the array is: 95.9291

max_num3 =

   95.9291

fx >>
```



```
⊞ AVIS_PhD_Project
⊞ Desktop
⊞ Documents
⊞ Downloads
⊞ Dropbox
  Music
⊞ Pictures
  Public
  sambashare
⊞ snap
⊞ Softwares
  Templates
  Videos
  examples.desktop
  ICV_findmax.m
  ICV_findmax.m~
```

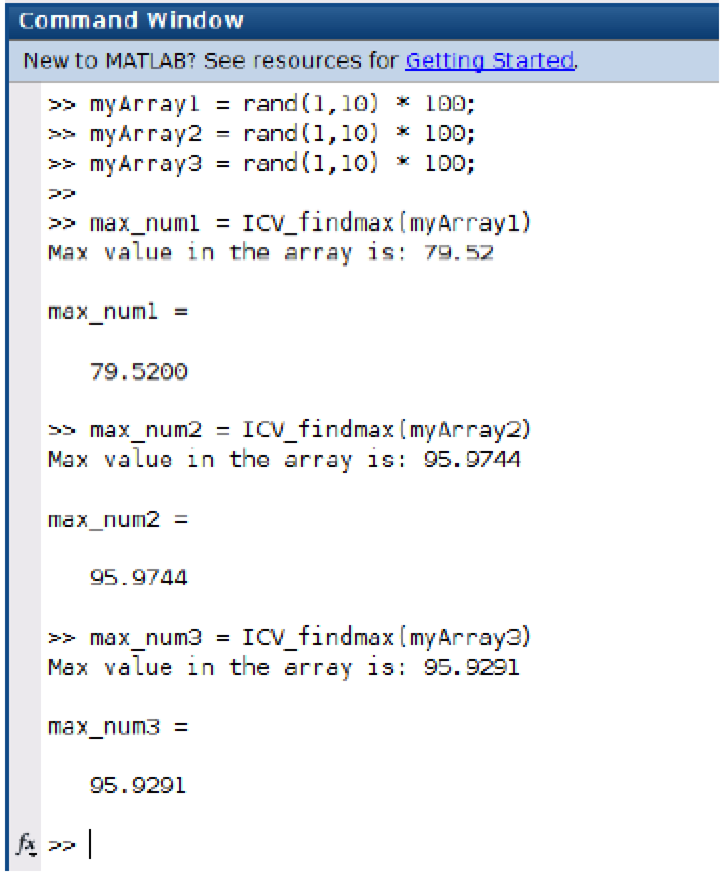
To call the function, the file must be visible in the current folder

If not, right click and Add to Path

Queen Mary
University of London

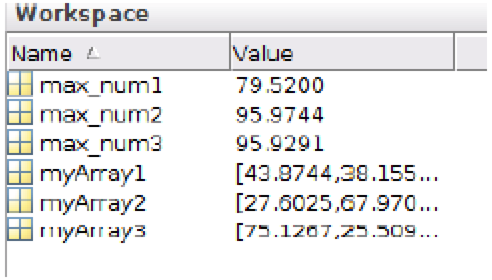
Functions

- Calling the function



New to MATLAB? See resources for Getting Started.

```
>> myArray1 = rand(1,10) * 100;
>> myArray2 = rand(1,10) * 100;
>> myArray3 = rand(1,10) * 100;
>>
>> max_num1 = ICV_findmax(myArray1)
Max value in the array is: 79.52

max_num1 =

   79.5200

>> max_num2 = ICV_findmax(myArray2)
Max value in the array is: 95.9744

max_num2 =

   95.9744

>> max_num3 = ICV_findmax(myArray3)
Max value in the array is: 95.9291

max_num3 =

   95.9291

>> |
```

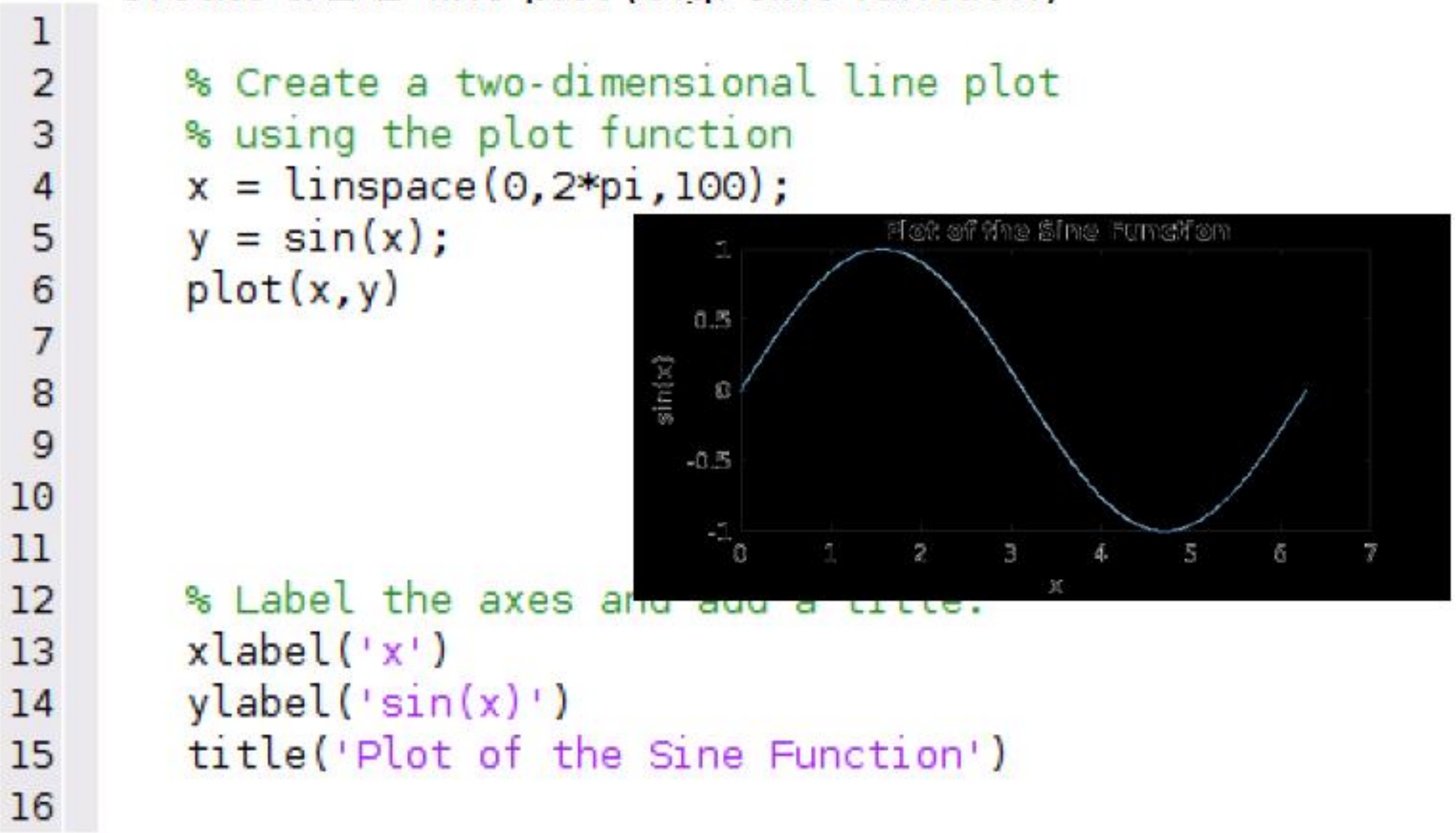


Name	Value
max_num1	79.5200
max_num2	95.9744
max_num3	95.9291
myArray1	[43.8744,38.155...
myArray2	[27.6025,67.970...
myArray3	[75.1267,25.509...

Variables inside the function are not visible in the workspace

Plotting basics

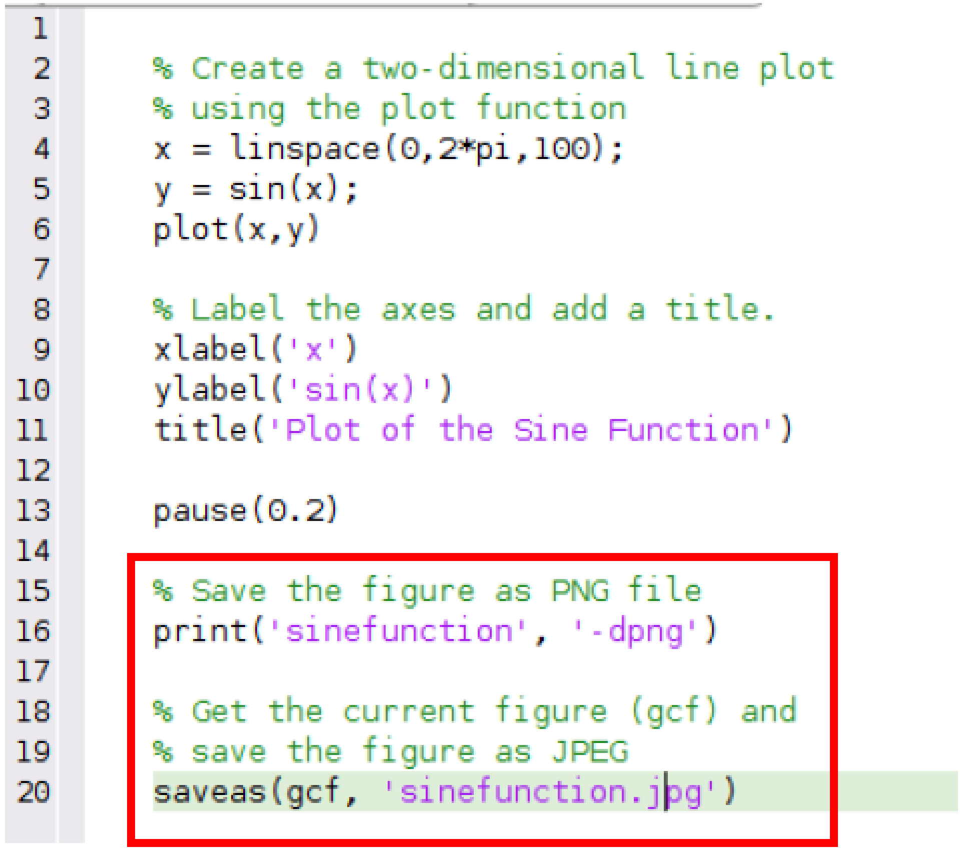
- Create a 2-D line plot (e.g. sine function)



```

% Create a two-dimensional line plot
% using the plot function
x = linspace(0,2*pi,100);
y = sin(x);
plot(x,y)





% Label the axes and add a title.
xlabel('x')
ylabel('sin(x)')
title('Plot of the Sine Function')

```

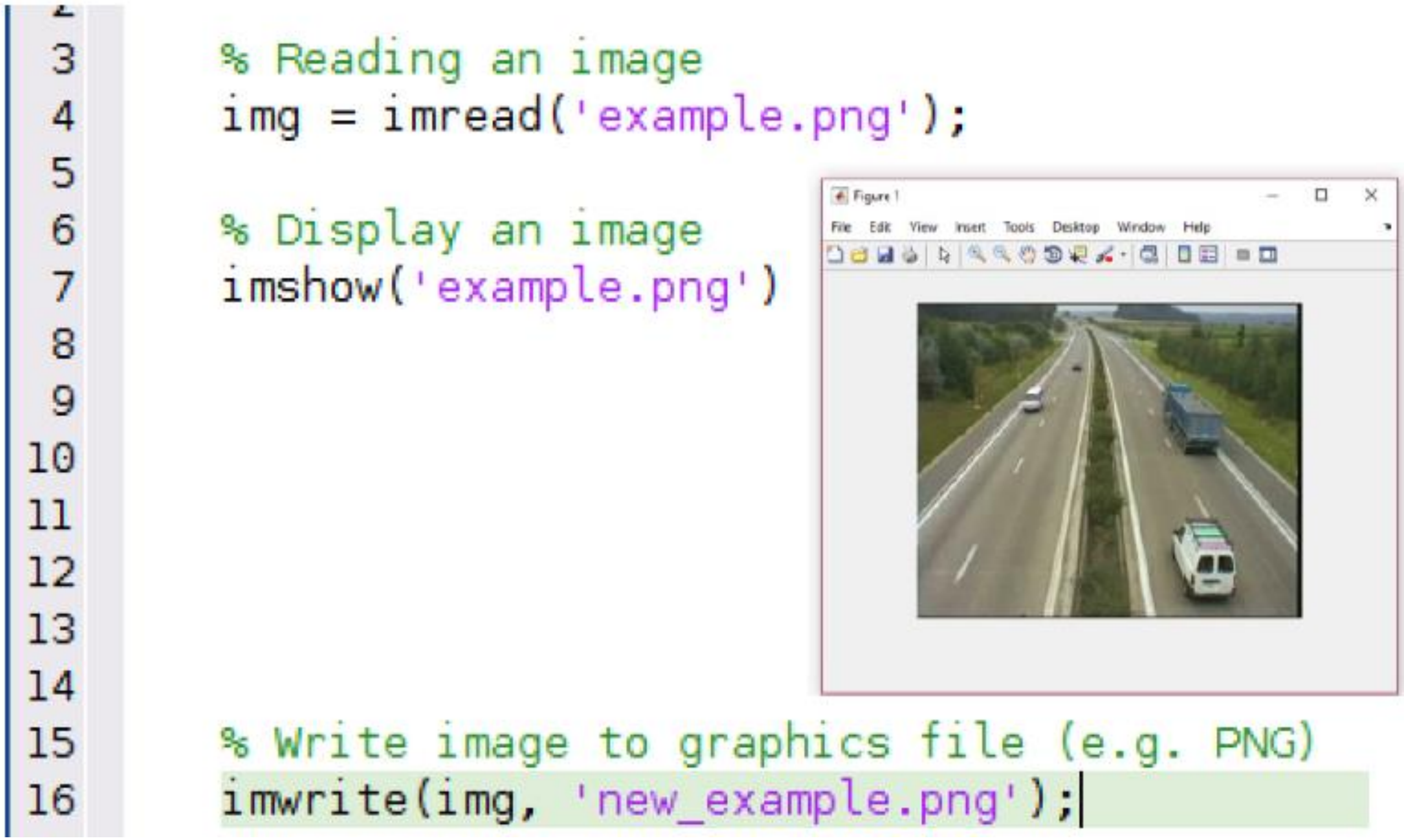
Saving figures

- Examples of commands to save a figure to a specific image file format



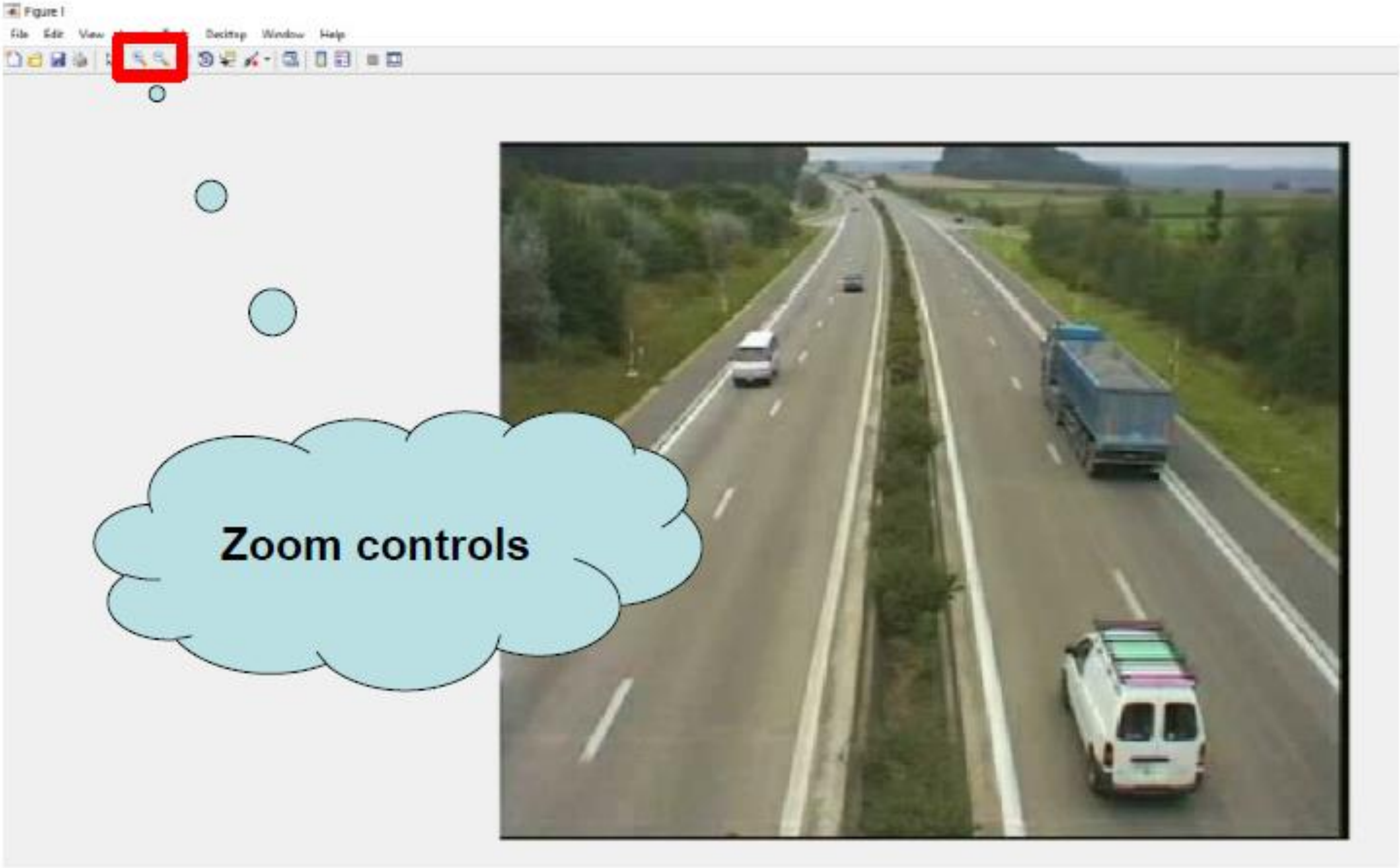
```
% Create a two-dimensional line plot
% using the plot function
x = linspace(0,2*pi,100);
y = sin(x);
plot(x,y)

% Label the axes and add a title.
xlabel('x')
ylabel('sin(x)')
title('Plot of the Sine Function')

pause(0.2)

% Save the figure as PNG file
print('sinefunction', '-dpng')

% Get the current figure (gcf) and
% save the figure as JPEG
saveas(gcf, 'sinefunction.jpg')
```

Load, display and saving an image



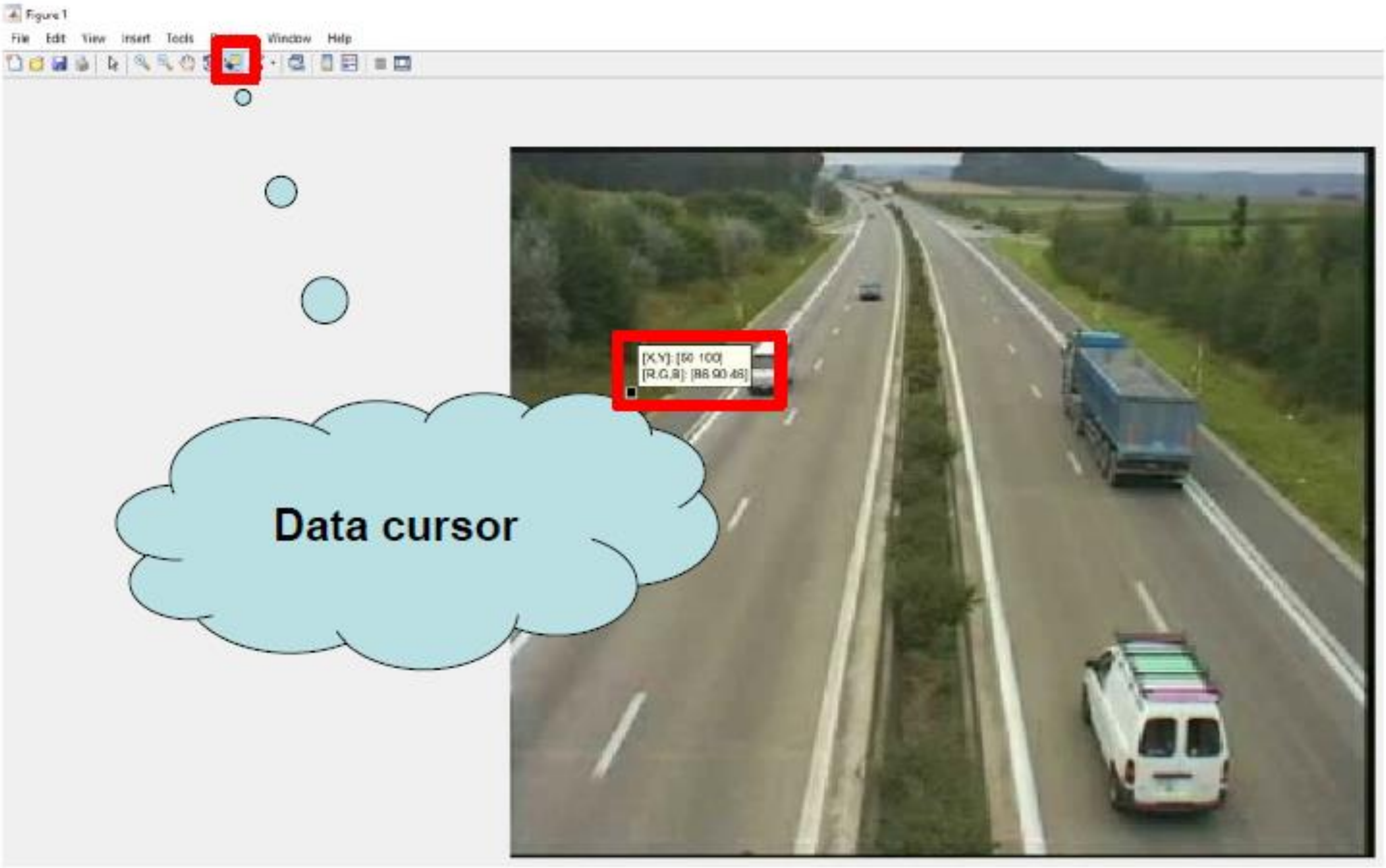
```
% Reading an image
img = imread('example.png');

% Display an image
imshow('example.png')








% Write image to graphics file (e.g. PNG)
imwrite(img, 'new_example.png');
```

Figure controls: zoom



Zoom controls

Figure controls: zoom

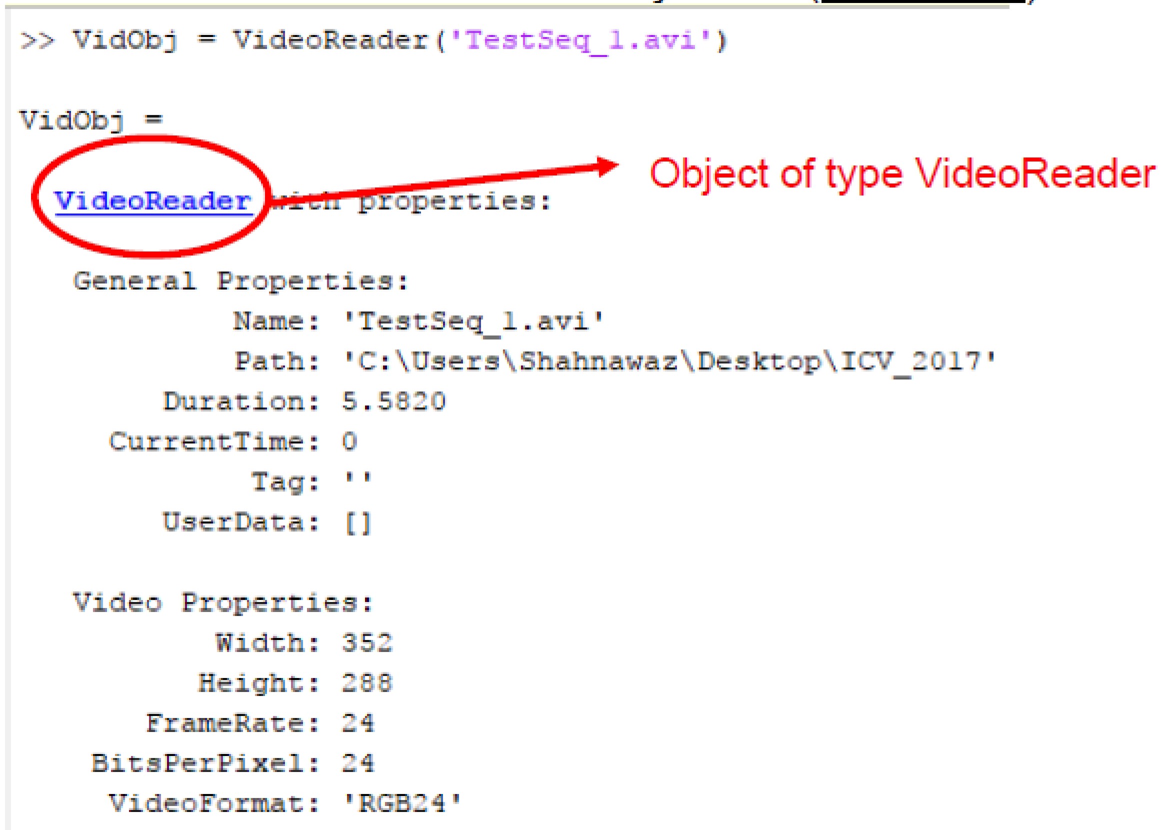


[X,Y]: [50 100]
[R,G,B]: [86 90 46]

Data cursor

Reading a video

- *VideoReader(video_filename)*
  - Obtain a struct with many fields (<u>properties</u>) related to the video



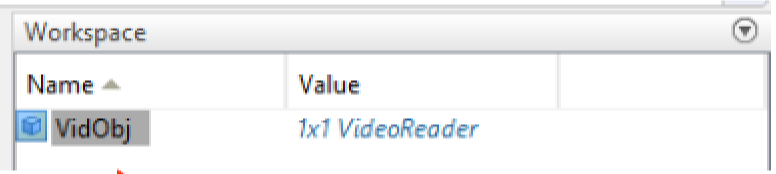
```
>> VidObj = VideoReader('TestSeq_1.avi')

VidObj = 

  VideoReader with properties:

   General Properties:
            Name: 'TestSeq_1.avi'
            Path: 'C:\Users\Shahnawaz\Desktop\ICV_2017'
        Duration: 5.5820
     CurrentTime: 0
             Tag: ''
        UserData: []

   Video Properties:
           Width: 352
          Height: 288
       FrameRate: 24
    BitsPerPixel: 24
     VideoFormat: 'RGB24'
```

Object of type VideoReader

Reading a video

- *VideoReader(video_filename)*
  - Obtain a struct with many fields (properties) related to the video

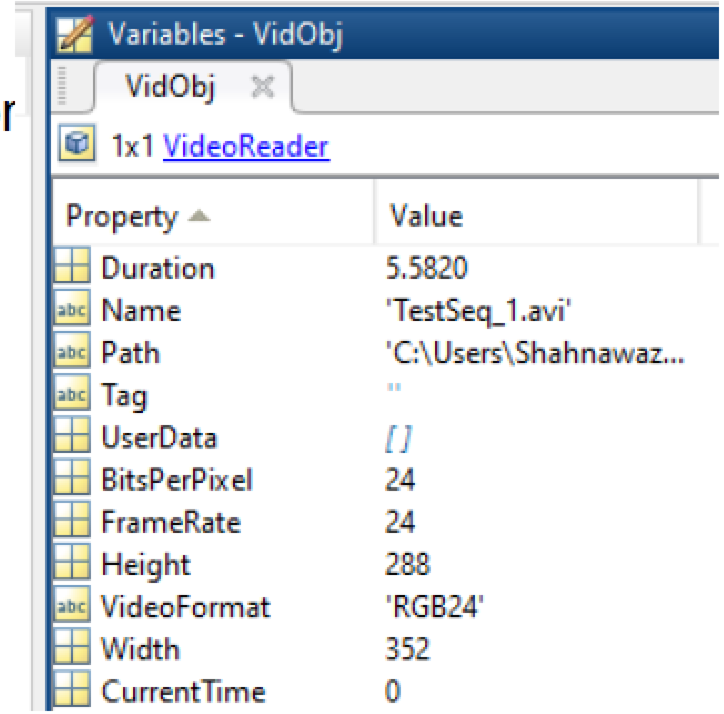
- Alternatively, to visualise the pr



Workspace

Name	Value
VidObj	1x1 VideoReader

Click on the variable in the workspace



Variables - VidObj

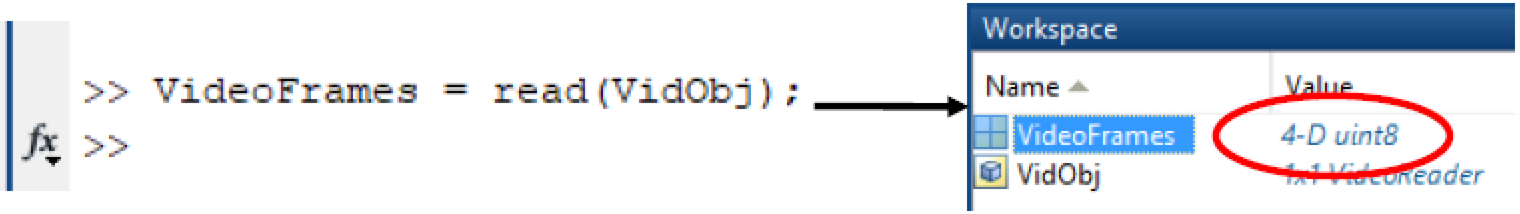
VidObj

1x1 VideoReader

Property	Value
Duration	5.5820
Name	'TestSeq_1.avi'
Path	'C:\Users\Shahnawaz...
Tag	''
UserData	[ ]
BitsPerPixel	24
FrameRate	24
Height	288
VideoFormat	'RGB24'
Width	352
CurrentTime	0

Queen Mary
University of London

Reading a video



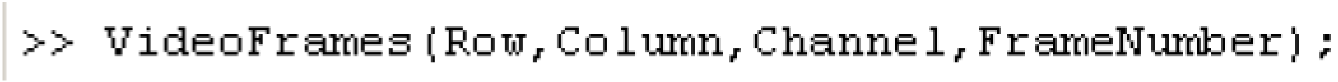
```
>> VideoFrames = read(VidObj);
>>
```

Workspace

Name	Value
VideoFrames	4-D uint8
VidObj	1x1 VideoReader

- All video frames are stored in a 4-D array as uint8 data type
  - Fourth dimension: Time / Frame number

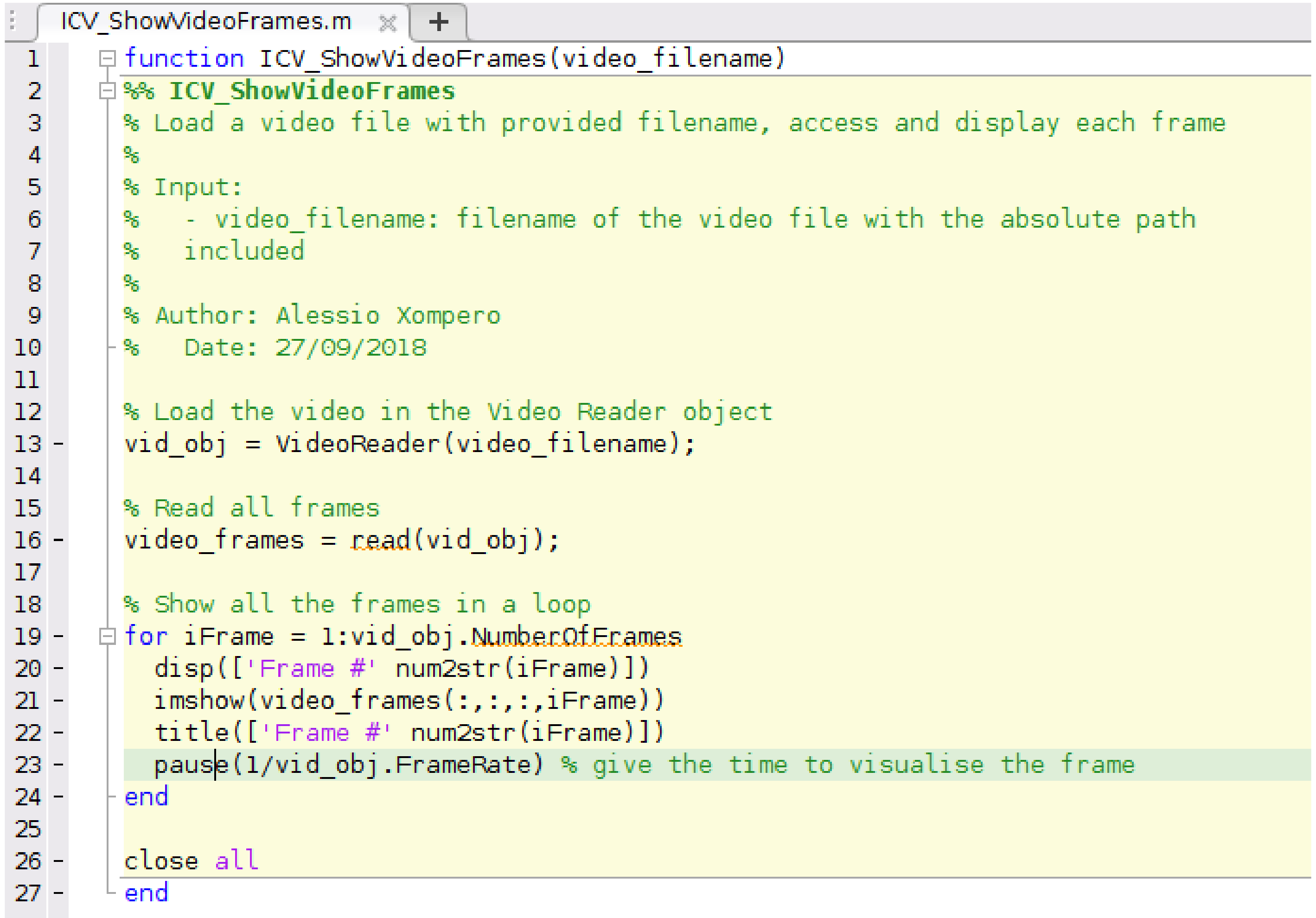
- Accessing any element of our 4-D array



```
>> VideoFrames(Row,Column,Channel,FrameNumber);
```

Queen Mary University of London

Reading a video



```
function ICV_ShowVideoFrames(video_filename)
%% ICV_ShowVideoFrames
% Load a video file with provided filename, access and display each frame
%
% Input:
%   - video_filename: filename of the video file with the absolute path
%   included
%
% Author: Alessio Xompero
%   Date: 27/09/2018

% Load the video in the Video Reader object
vid_obj = VideoReader(video_filename);

% Read all frames
video_frames = read(vid_obj);

% Show all the frames in a loop
for iFrame = 1:vid_obj.NumberOfFrames
  disp(['Frame #' num2str(iFrame)])
  imshow(video_frames(:,:,:,iFrame))
  title(['Frame #' num2str(iFrame)])
  pause(1/vid_obj.FrameRate) % give the time to visualise the frame
end

close all
end
```

Coursework requirements

- You can use your preferred programming language (that is supported in the ITL)
- The functions/procedures/classes you write will start with the prefix ICV_
- You can use freeware software, as long as the source is acknowledged
- The software shall be commented (the comments should allow an intermediate programmer to understand each part of the code)

Question 1

1) **Transformations.**
Rotation, translation and skew are useful operations for matching, tracking, and data augmentation.

a) Write a function that takes as input an image *I*, rotates it by an angle θ_1 and horizontally skews it by an angle, θ_2. Write the matrix formulation for image rotation R(.) and skewing S(.). Define all the variables. Note that the origin of the coordinate system of the programming environment you use might be different from the one shown in the lectures.

b) Create an image that contains your name written in Arial, point 72, capital letters. Rotate clockwise the image you created by 30, 60, 120 and -50 degrees. Skew the same image by 10, 40 and 60 degrees. *Complete the process so that all the pixels have a value*. Discuss in the report the advantages and disadvantages of different approaches.

c) Analyse the results when you change the order of the two operators: R(S(*I*)) and S(R(*I*)).
i) Rotate the image by $\theta_1 = 20$ clockwise and then skew the result by $\theta_2 = 50$.
ii) Skew the image by $\theta_2 = 50$ and then rotate the result by $\theta_1 = 20$ clockwise.
Are the results of (i) and (ii) the same? Why?

“Forbidden” functions

- Each exercise has a list of functions that should not be used
- Please DO NOT use image processing libraries or toolboxes that automatically solve the main tasks of the coursework
- Q1: Any function that does rotation, shear (skew), affine transformrations, warping, rescaling, resizing.

Matlab

- imrotate
- imtranslate
- imwarp
- imresize
- marketform

Python

- cv2.getRotationMatrix2D
- cv2.warpAffine
- cv2.getAffineTransform
- cv2.resize

If you are unsure please ask the demonstrator

Useful links

- MATLAB tutorial: https://www.youtube.com/watch?v=T_ekAD7U-wU
- Python tutorial: https://www.youtube.com/watch?v=QXeEoD0pB3E&list=PLY-UbAd0uV4N98dg5_vImpHhL30qkvvK4

- Setting up a virtual environment
  - Anaconda: https://www.youtube.com/watch?v=kU_ZtZhmmEU&list=PLsyeobzWxl7poL9JTVyndKe62ieoN-MZ3&index=83

- Computerphile – resizing images: https://www.youtube.com/watch?v=AqscP7rc8_M

- EECS IT services: http://support.eecs.qmul.ac.uk/

- QMUL student MATLAB license: https://www.its.qmul.ac.uk/support/self-help/software/free-and-discounted-software/matlab/