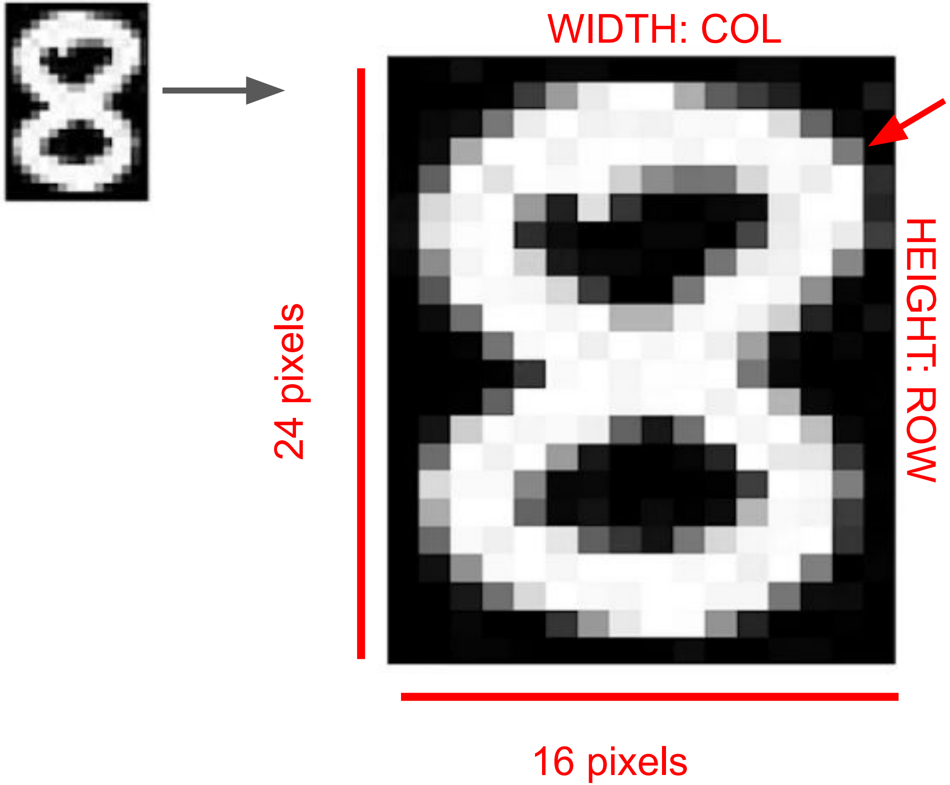
ENME 303 LAB

Week 11: Imaging In Matlab

Nameless Lab

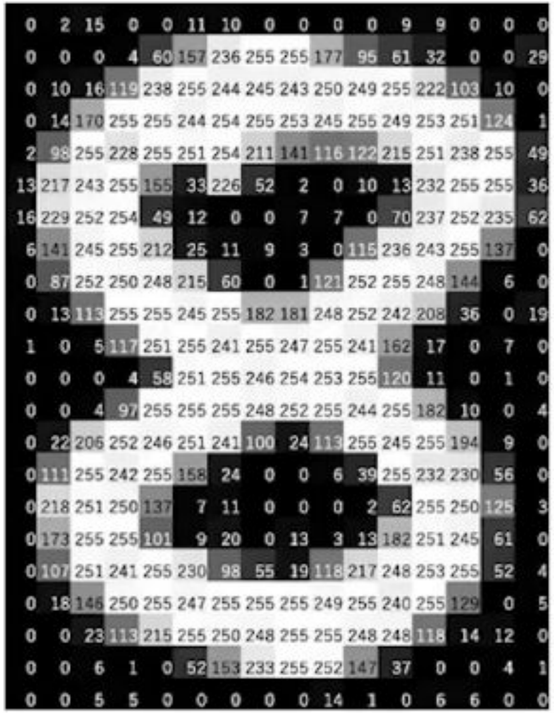
How Images are stored on Computers- Greyscale

- Images composed of individual boxes
  - Pixels
- Dimensions of image: X by Y
  - X is number of pixels **in the height (rows)**
  - Y is number of pixels **in the width (columns)**
- The dimensions this image is then: **24 x 16 pixels**

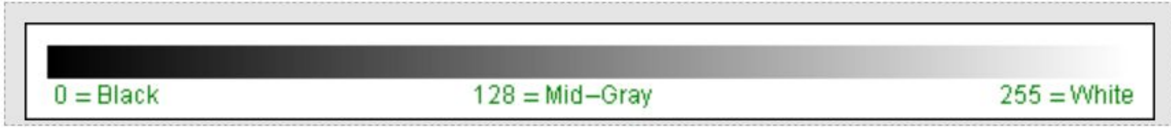


How Images are stored on Computers- Greyscale

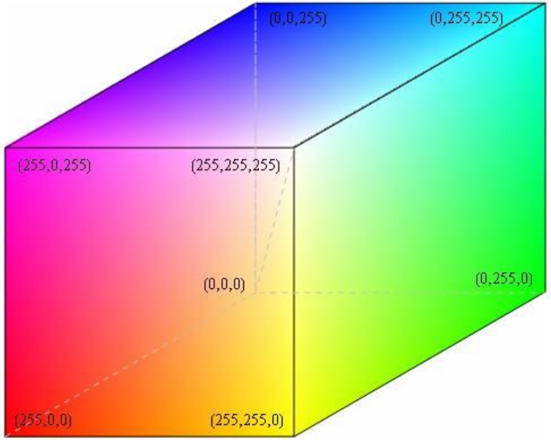
- Each pixel has a corresponding pixel values (0→ 255)
- Pixel values represent the intensity of each pixel
- Pixel values stored in a matrix
    - 1 Channel for greyscale
    - 24 x 16

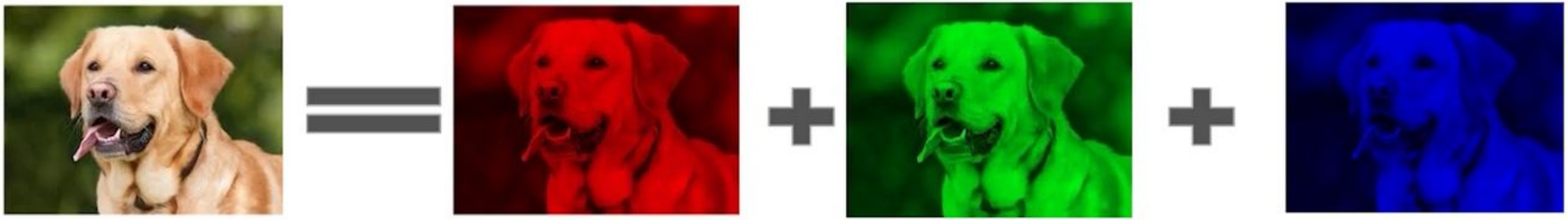


0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0



How Images are stored on Computers- Color





Colour Image

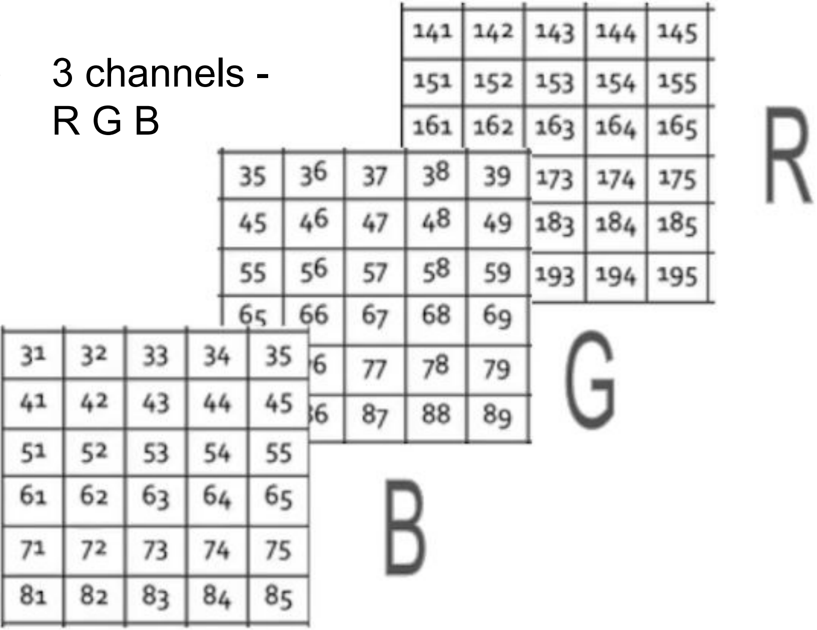
Red

Green

Blue

How Images are stored on Computers- Color

- 3 channels - R G B



R

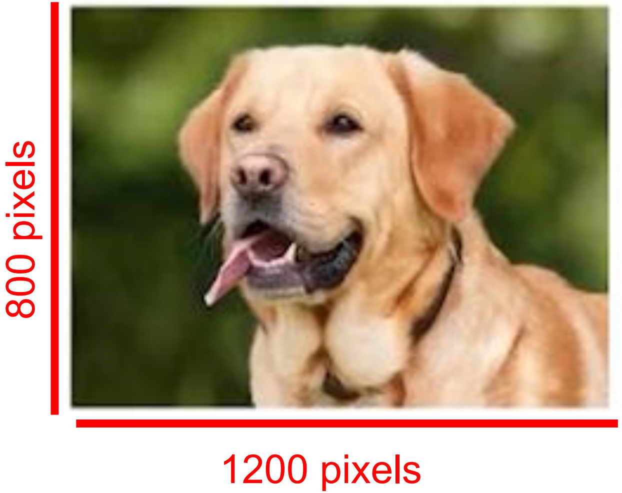
141	142	143	144	145
151	152	153	154	155
161	162	163	164	165
		173	174	175
		183	184	185
		193	194	195

G

35	36	37	38	39
45	46	47	48	49
55	56	57	58	59
65	66	67	68	69
	6	77	78	79
	6	87	88	89

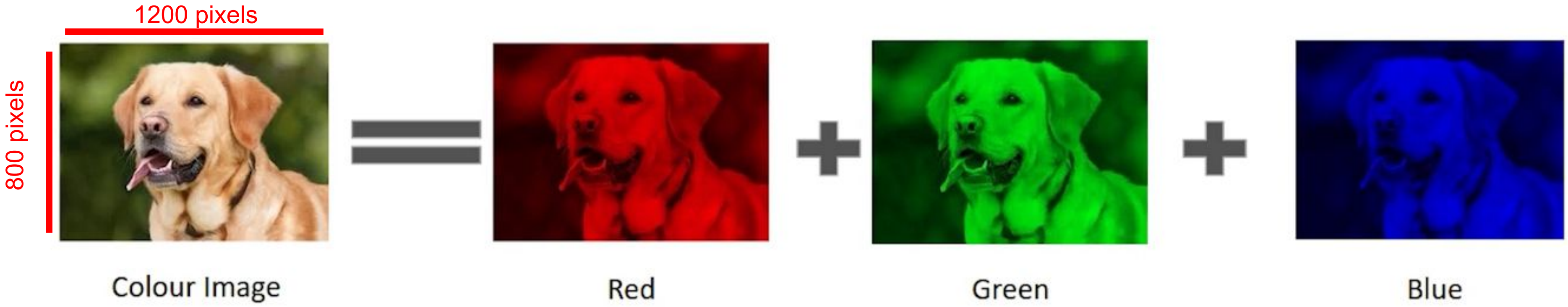
B

31	32	33	34	35
41	42	43	44	45
51	52	53	54	55
61	62	63	64	65
71	72	73	74	75
81	82	83	84	85



- Color pixel value domain: 0 → 255
  - Low to high intensity

How Images are stored on Computers- Color



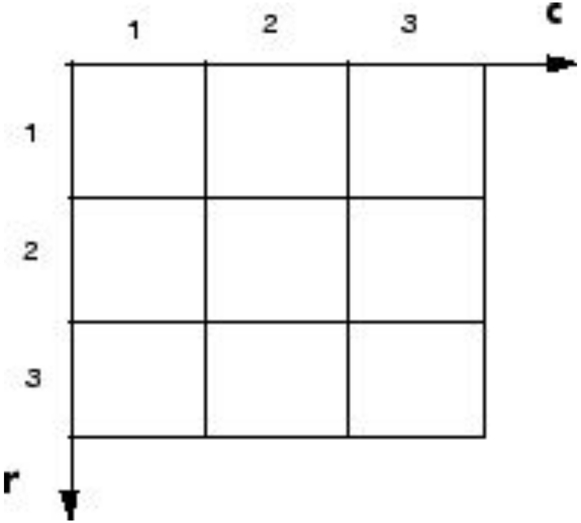
RGB color matrices superimposed to create color image

Dimension: R x C x 3 → **our image : 800 x 1200 x 3**

Where R is number of pixels in the height (row), C is the number of pixels in the width (column), and 3 is the number of channels (R + G + B =3)

MATLAB Image Storage

- MATLAB stores images as 2D matrices of pixel values
  - Each (row, column) index of matrix refers to a single pixel in the image
  - Color images are multi-channel
- MATLAB's Image Coordinate System:



Pixel indices are integers, ranging from 1 to the length of the row or column.

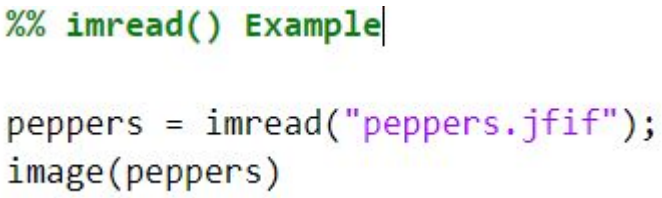
R: top → bottom
C: left → right

MATLAB’s Built-in Functions

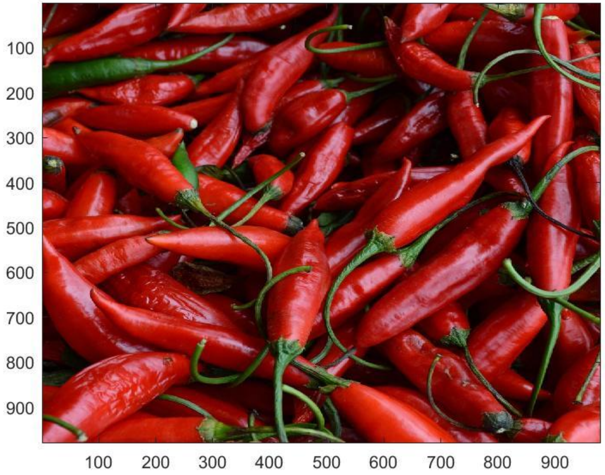
To read in an image in MATLAB we use: imread()

General syntax: A = imread(‘filename’)

For MATLAB to be able to locate your file, be sure to be working in the current folder



```
%% imread() Example

peppers = imread("peppers.jfif");
image(peppers)
```



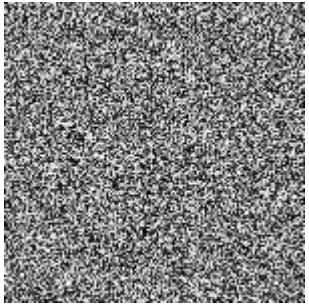
MATLAB’s Built-in Functions

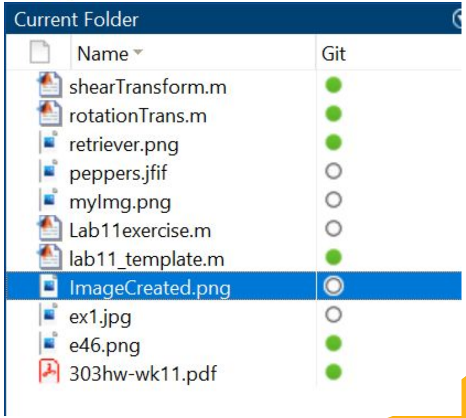
To generate and output an image in MATLAB we use: imwrite()

General syntax: imwrite(A, ‘filename’)

Take image data A and outputs an image with the filename

```
%% imwrite() Example
%creates a 150-by-150 matrix of random pixel value (1 channel- B&W)
A = rand(150);
%Outputs this randomized matrix of pixel values
% to an image in my working folder
imwrite(A,'ImageCreated.png')
```



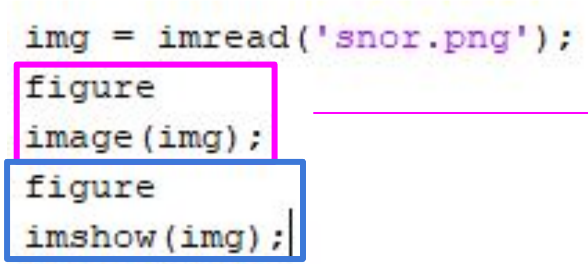


Current Folder

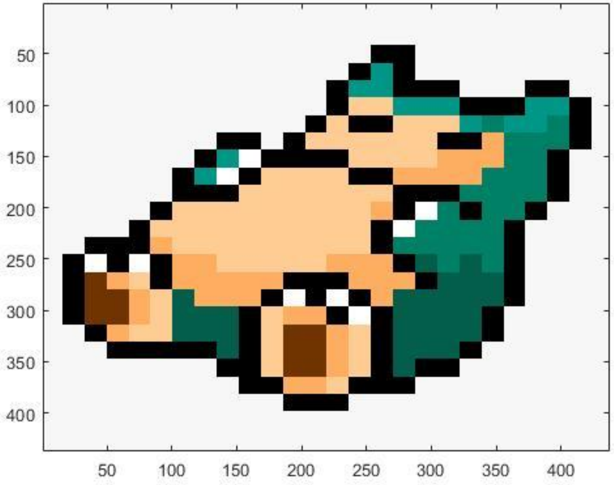
Name	Git
shearTransform.m	●
rotationTrans.m	●
retriever.png	●
peppers.jfif	○
myImg.png	○
Lab11exercise.m	○
lab11_template.m	●
ImageCreated.png	○
ex1.jpg	○
e46.png	●
303hw-wk11.pdf	●

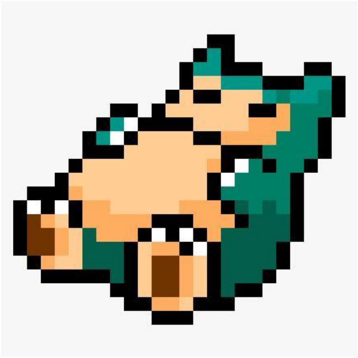
MATLAB’s Built-in Functions

To show an image in MATLAB we use: imshow()



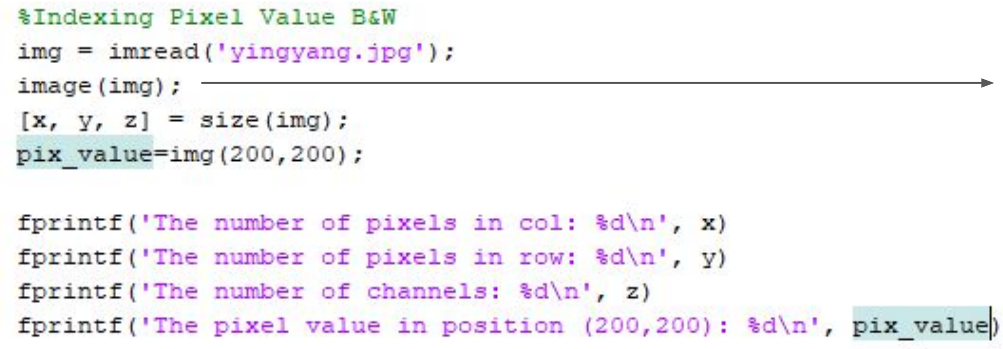
```
img = imread('snor.png');
figure
image(img);
figure
imshow(img);
```



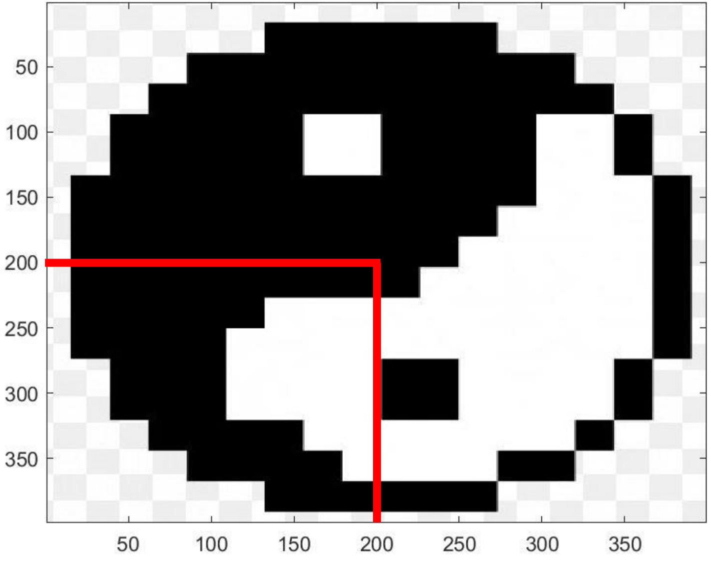


Indexing a B&W Image

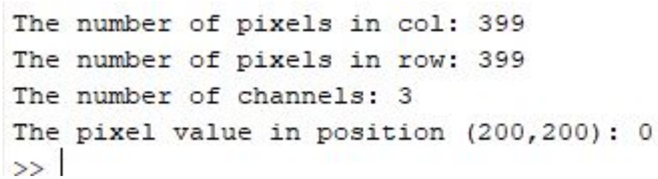
Let's index a B&W image:



```
%Indexing Pixel Value B&W
img = imread('yingyang.jpg');
image(img);
[x, y, z] = size(img);
pix_value=img(200,200);

fprintf('The number of pixels in col: %d\n', x)
fprintf('The number of pixels in row: %d\n', y)
fprintf('The number of channels: %d\n', z)
fprintf('The pixel value in position (200,200): %d\n', pix_value)
```



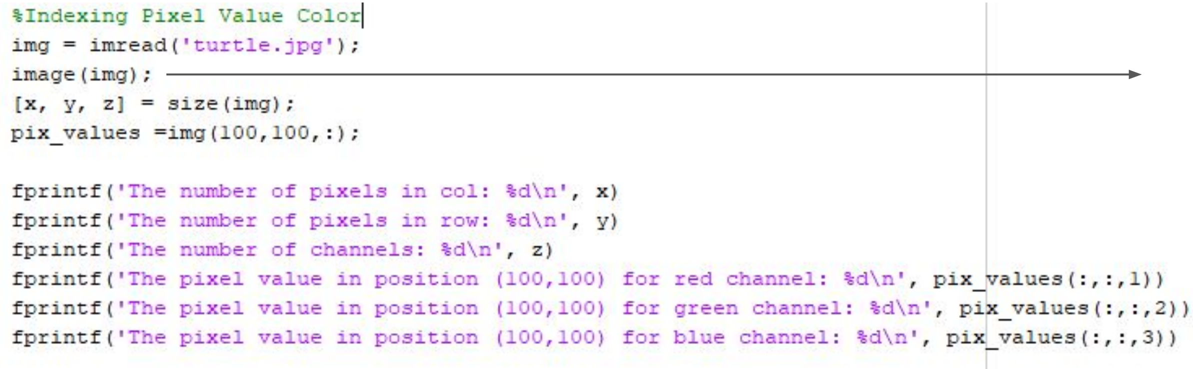
Returns:



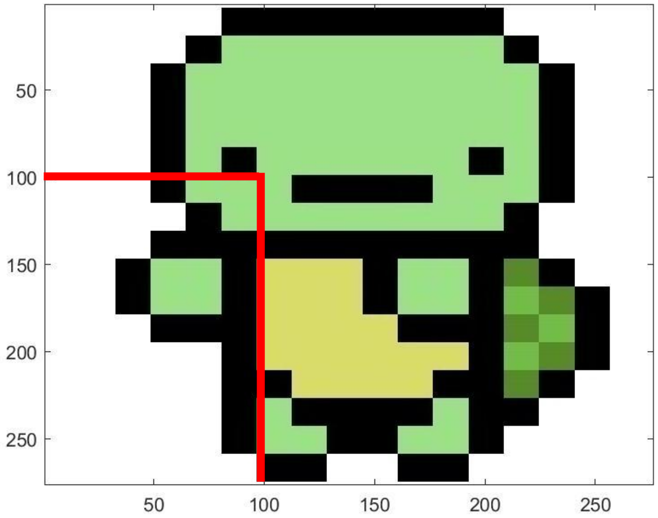
```
The number of pixels in col: 399
The number of pixels in row: 399
The number of channels: 3
The pixel value in position (200,200): 0
>> |
```

Indexing a Color Image

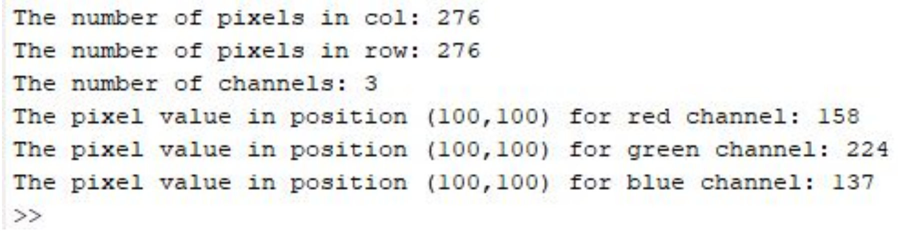
Let's index a color image:



```
%Indexing Pixel Value Color
img = imread('turtle.jpg');
image(img);
[x, y, z] = size(img);
pix_values =img(100,100,:);

fprintf('The number of pixels in col: %d\n', x)
fprintf('The number of pixels in row: %d\n', y)
fprintf('The number of channels: %d\n', z)
fprintf('The pixel value in position (100,100) for red channel: %d\n', pix_values(:,:,1))
fprintf('The pixel value in position (100,100) for green channel: %d\n', pix_values(:,:,2))
fprintf('The pixel value in position (100,100) for blue channel: %d\n', pix_values(:,:,3))
```

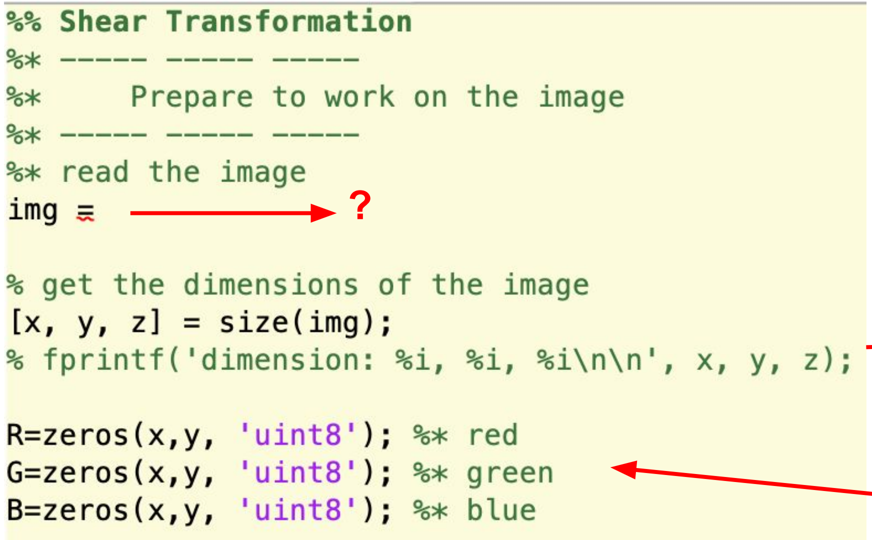


Returns:



```
The number of pixels in col: 276
The number of pixels in row: 276
The number of channels: 3
The pixel value in position (100,100) for red channel: 158
The pixel value in position (100,100) for green channel: 224
The pixel value in position (100,100) for blue channel: 137
>>
```

Reviewing Lab 11 Template- Set up

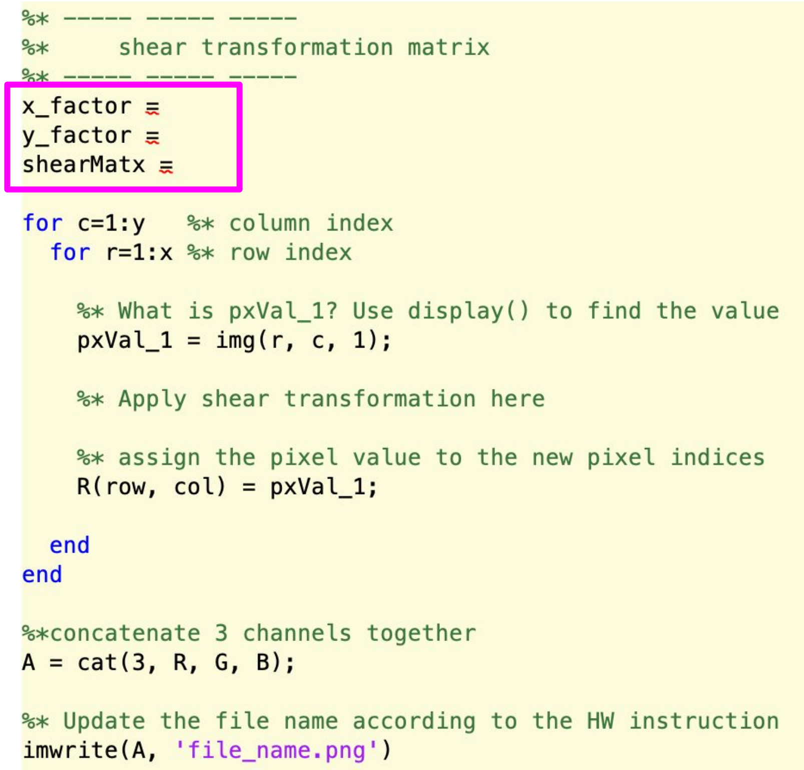


```
%% Shear Transformation
%* ----- ----- -----
%*     Prepare to work on the image
%* ----- ----- -----
%* read the image
img =

% get the dimensions of the image
[x, y, z] = size(img);
% fprintf('dimension: %i, %i, %i\n\n', x, y, z);

R=zeros(x,y, 'uint8'); %* red
G=zeros(x,y, 'uint8'); %* green
B=zeros(x,y, 'uint8'); %* blue
```

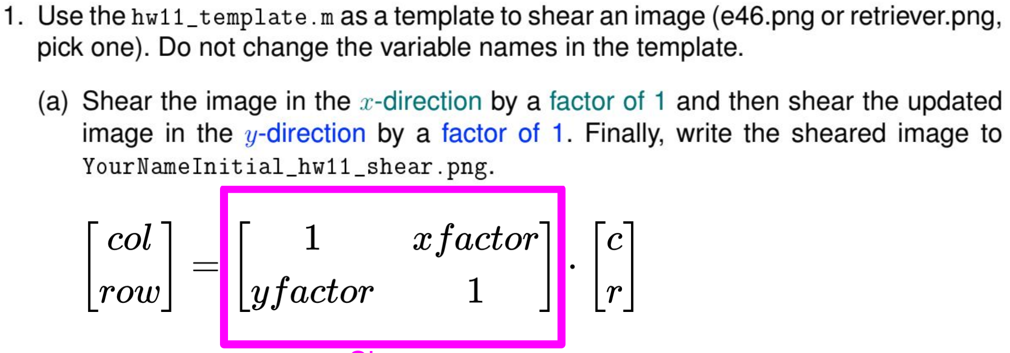
dimension: 1300, 1300, 3

What do these lines do?

Reviewing Lab 11 Template- Shearing



```
%* ----- ----- -----
%*     shear transformation matrix
%* ----- ----- -----
x_factor =
y_factor =
shearMatx =

for c=1:y   %* column index
  for r=1:x %* row index

    %* What is pxVal_1? Use display() to find the value
    pxVal_1 = img(r, c, 1);

    %* Apply shear transformation here

    %* assign the pixel value to the new pixel indices
    R(row, col) = pxVal_1;

  end
end

%*concatenate 3 channels together
A = cat(3, R, G, B);

%* Update the file name according to the HW instruction
imwrite(A, 'file_name.png')
```

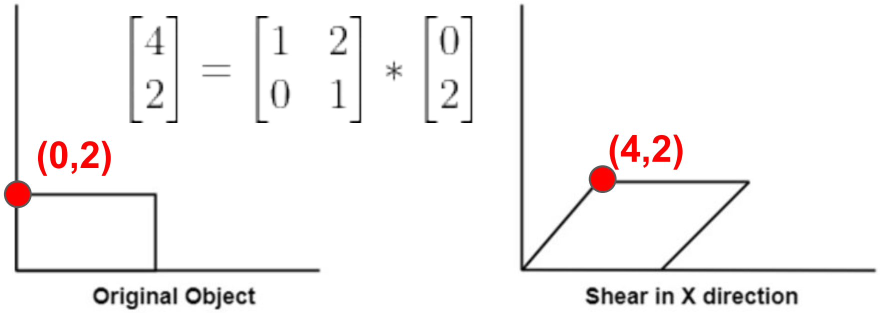


1. Use the `hw11_template.m` as a template to shear an image (e46.png or retriever.png, pick one). Do not change the variable names in the template.

   (a) Shear the image in the x-direction by a factor of 1 and then shear the updated image in the y-direction by a factor of 1. Finally, write the sheared image to `YourNameInitial_hw11_shear.png`.

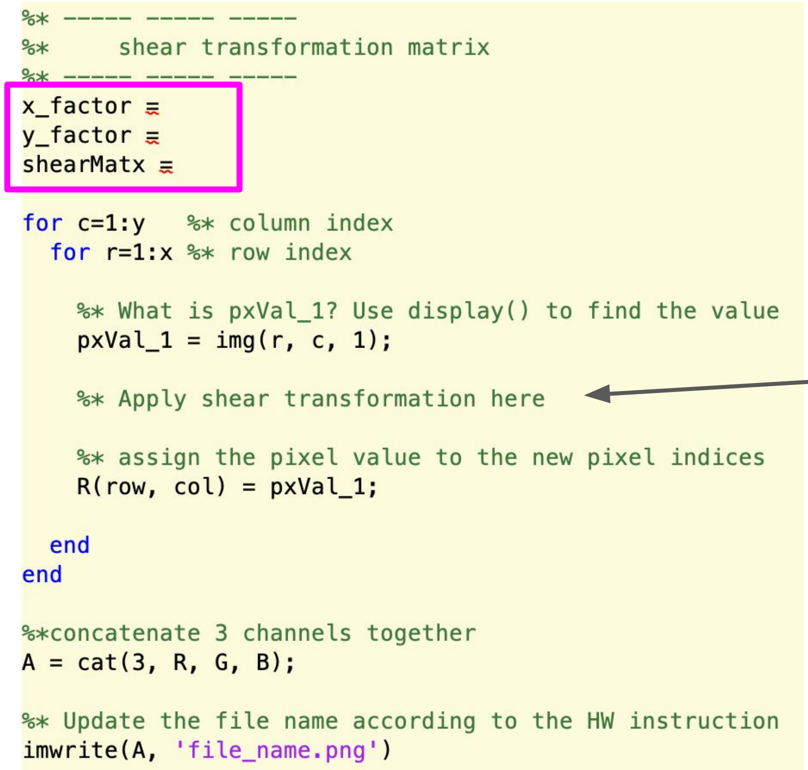
$$\begin{bmatrix} col \\ row \end{bmatrix} = \boxed{\begin{bmatrix} 1 & xfactor \\ yfactor & 1 \end{bmatrix}} \cdot \begin{bmatrix} c \\ r \end{bmatrix}$$

Shear Transformation Matrix

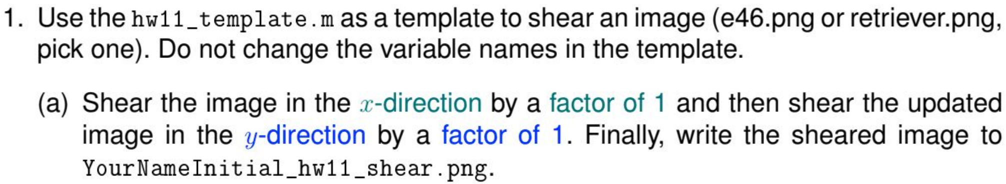


$$\begin{bmatrix} 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

Reviewing Lab 11 Template- Shearing



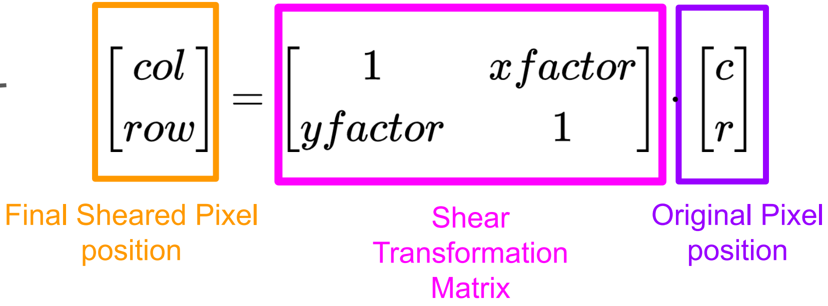
```
%* ----- ----- -----
%*     shear transformation matrix
%* ----- ----- -----
x_factor =
y_factor =
shearMatx =

for c=1:y   %* column index
  for r=1:x %* row index

    %* What is pxVal_1? Use display() to find the value
    pxVal_1 = img(r, c, 1);

    %* Apply shear transformation here

    %* assign the pixel value to the new pixel indices
    R(row, col) = pxVal_1;

  end
end

%*concatenate 3 channels together
A = cat(3, R, G, B);

%* Update the file name according to the HW instruction
imwrite(A, 'file_name.png')
```



1. Use the hw11_template.m as a template to shear an image (e46.png or retriever.png, pick one). Do not change the variable names in the template.

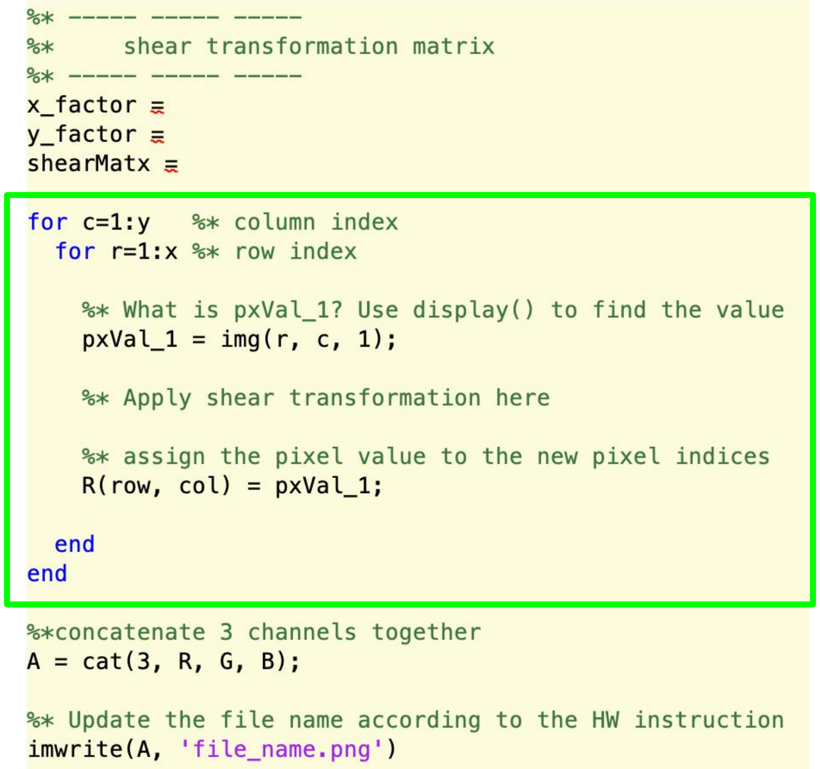
   (a) Shear the image in the x-direction by a factor of 1 and then shear the updated image in the y-direction by a factor of 1. Finally, write the sheared image to YourNameInitial_hw11_shear.png.

Transformation work!



$$\underbrace{\begin{bmatrix} col \\ row \end{bmatrix}}_{\text{Final Sheared Pixel position}} = \underbrace{\begin{bmatrix} 1 & xfactor \\ yfactor & 1 \end{bmatrix}}_{\text{Shear Transformation Matrix}} \cdot \underbrace{\begin{bmatrix} c \\ r \end{bmatrix}}_{\text{Original Pixel position}}$$

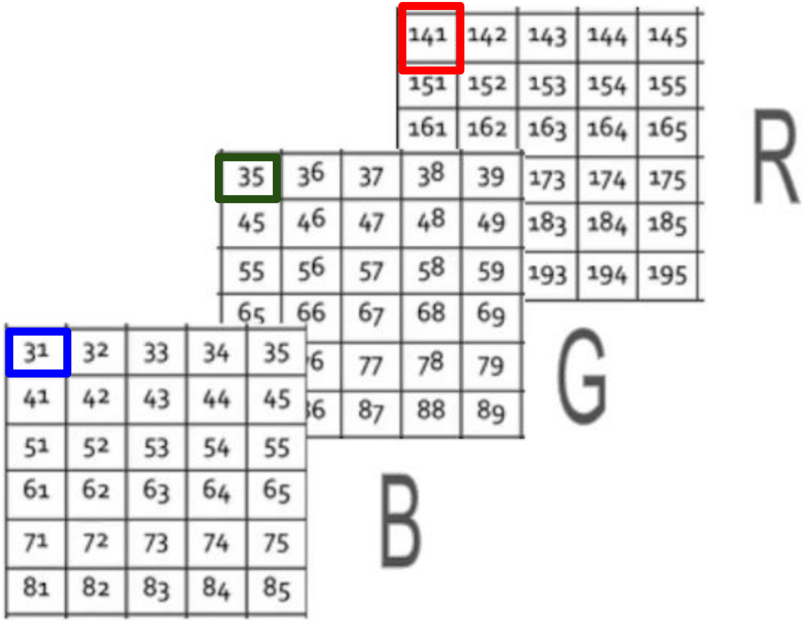
Reviewing Lab 11 Template- Shearing



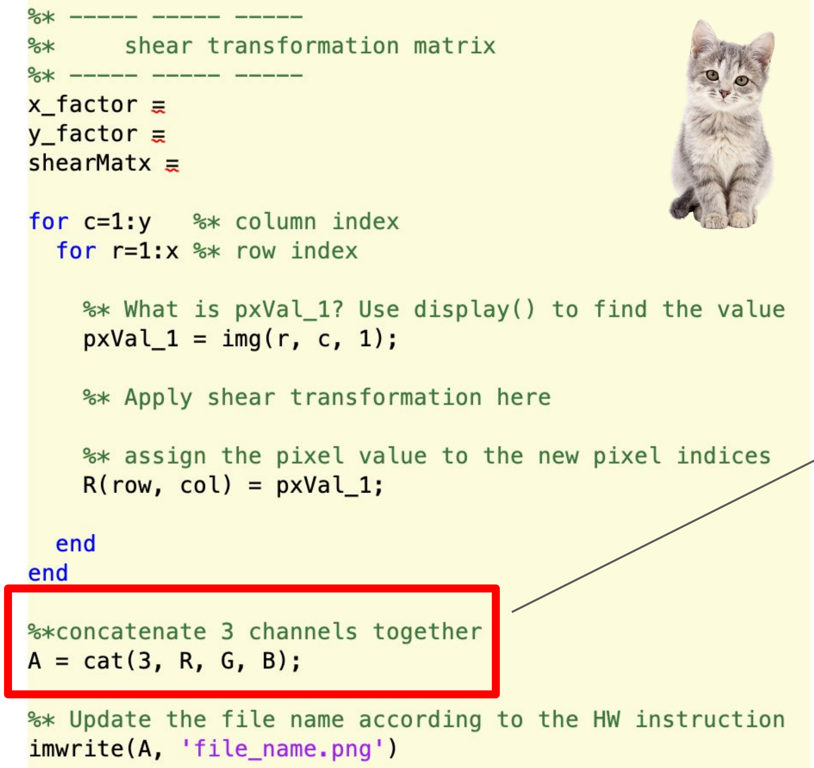
```
%* ----- ----- -----
%*     shear transformation matrix
%* ----- ----- -----
x_factor =
y_factor =
shearMatx =

for c=1:y   %* column index
  for r=1:x %* row index

    %* What is pxVal_1? Use display() to find the value
    pxVal_1 = img(r, c, 1);

    %* Apply shear transformation here

    %* assign the pixel value to the new pixel indices
    R(row, col) = pxVal_1;

  end
end

%*concatenate 3 channels together
A = cat(3, R, G, B);

%* Update the file name according to the HW instruction
imwrite(A, 'file_name.png')
```

1. Use the `hw11_template.m` as a template to shear an image (e46.png or retriever.png, pick one). Do not change the variable names in the template.

   (a) Shear the image in the x-direction by a factor of 1 and then shear the updated image in the y-direction by a factor of 1. Finally, write the sheared image to `YourNameInitial_hw11_shear.png`.

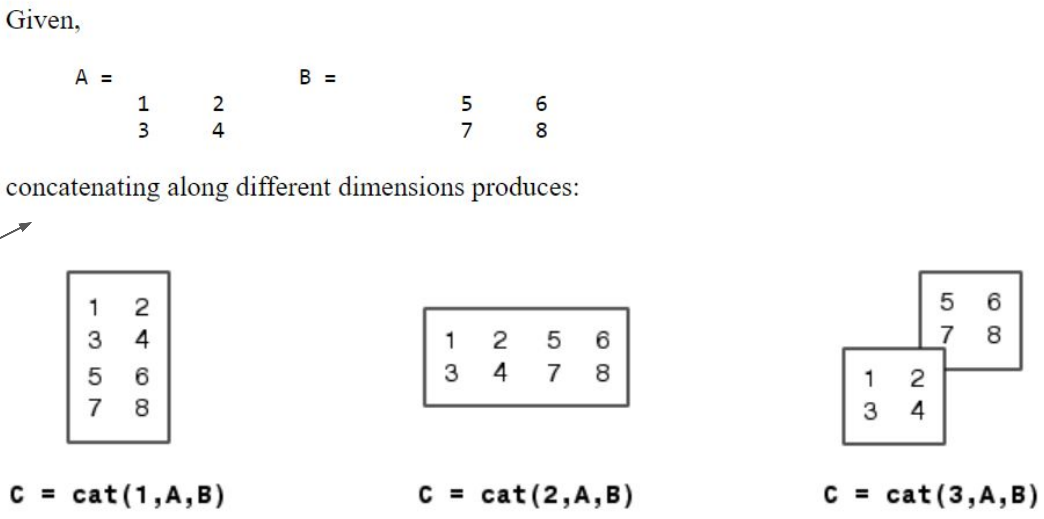


Reviewing Lab 11 Template- Shearing



```
%* ----- ----- -----
%*     shear transformation matrix
%* ----- ----- -----
x_factor =
y_factor =
shearMatx =

for c=1:y   %* column index
  for r=1:x %* row index

    %* What is pxVal_1? Use display() to find the value
    pxVal_1 = img(r, c, 1);

    %* Apply shear transformation here

    %* assign the pixel value to the new pixel indices
    R(row, col) = pxVal_1;

  end
end

%*concatenate 3 channels together
A = cat(3, R, G, B);

%* Update the file name according to the HW instruction
imwrite(A, 'file_name.png')
```

MATLAB’s let's us concatenate arrays in 1D, 2D, or for image processing 3D using the built in: cat(dim,A,B,...N)



Given,

```
A =                 B =
     1    2              5    6
     3    4              7    8
```

concatenating along different dimensions produces:

Acknowledgement

The lab slides you see are not made by one person. All the TA/TFs served for this course have contributed their effort and time to the slides. Below are the leading TFs for each semester:

- 2021 FA - Karla Negrete (GTA)
- 2022 SP - Justin Grahovac
- 2022 FA - Kelli Boyer, Yisrael Wealcatch, Noelle Ray (GTA)
- 2024 SP - Riyaz Rehman, Mahamoudou Bah and Michael Mullaney