# COGS 119
# MATLAB for Experimental Research

Fall 2014 – Week 4

Image Processing in Matlab

# Figures and Images

```
>> help figure
```

figure(): creates a figure window

figure(H): makes H the current figure, forces it to become visible, and raises it above all other figures on the screen. If Figure H does not exist, and H is an integer, a new figure is created with handle H.

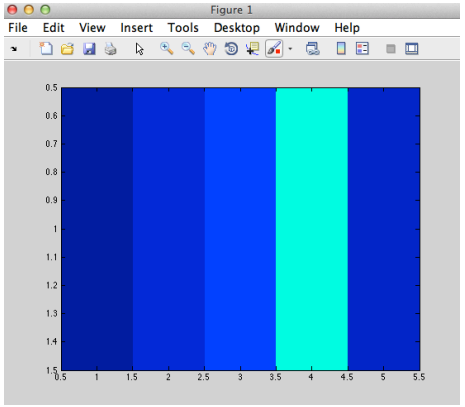
```
>> help image
```

image(X): displays matrix X as an image.

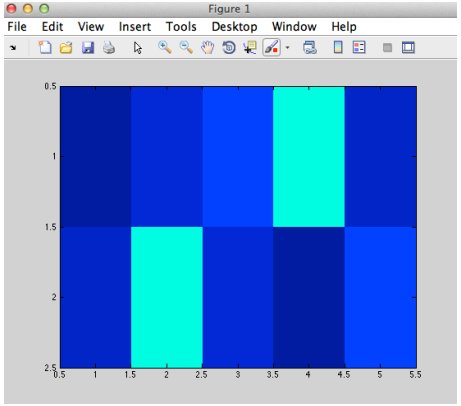
Each element of X specifies the color of a rectilinear patch in the image.

# Figures and Images

```
>> x = [1 5 26 10 4];
>> figure(1)
>> image(x)
```



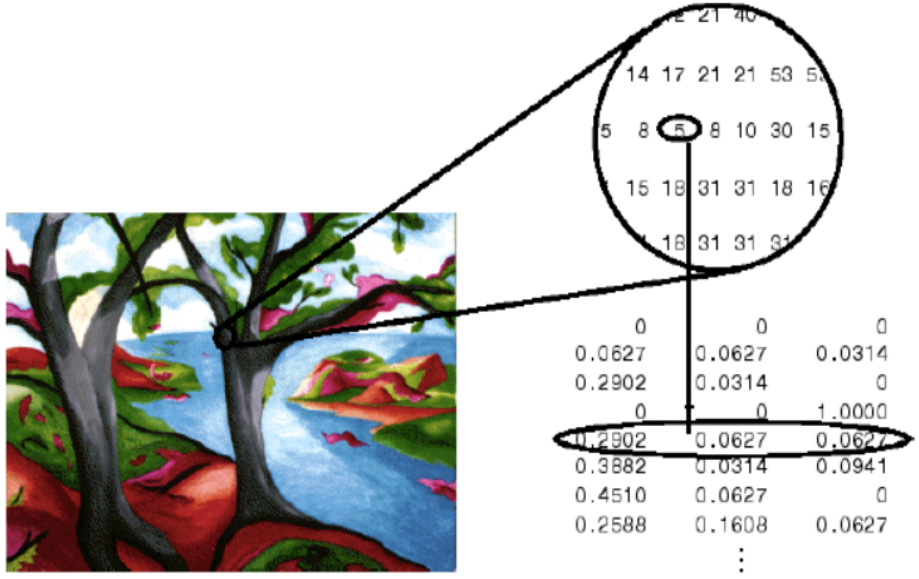

```
>> x =  [1 5 10 26 4 ;  4 26 5 1 10];
>> figure(2)
>> image(x)
```




How do the colors refer to numbers?

# Indexed images in Matlab

- An indexed image consists of a data matrix X and a colormap matrix, which is N x 3 array of class double containing values in the range [0,1].
- Each row of the map specifies red, blue and green components of a single color.
- The color of each image pixel is determined by using the corresponding value of X as an index into the map.
- The value 1 points to the first row in the map, 2 points to the second row, and so on.
- Values of X must be integers.



# Colormap

- You can use defaults or make your own.

```
>> help colormap
```

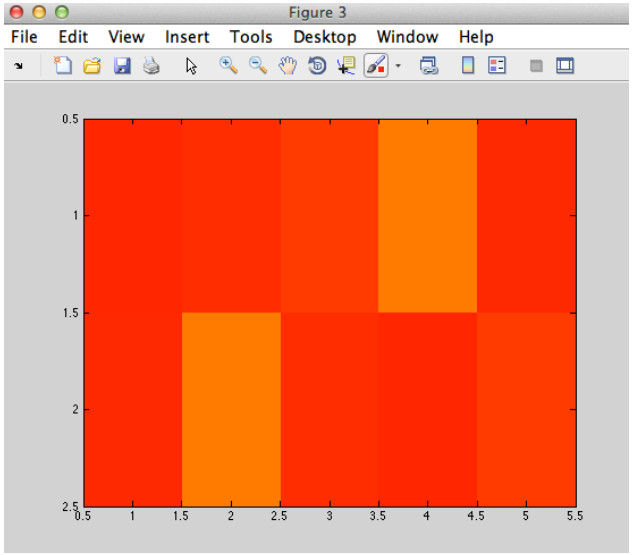
colormap(MAP) sets the current figure's colormap to MAP.

- Some built-in colormaps are:

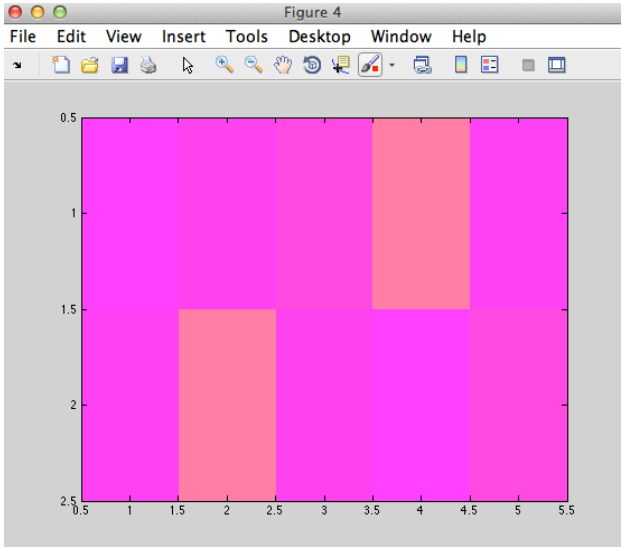

Jet: Default colormap

# Colormap

```
>> figure(3);
>> image(x);
>> colormap('autumn')
```




```
>> figure(4);
>> image(x);
>> colormap('spring')
```



# Colormap

- A colormap is also a matrix.
- For color images, it is an N x 3 matrix where the three columns corresponds to the RGB (red, green, blue) indices.

# Colormap

```
>> c = colormap('spring');
```

c <64x3 double>

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 1 | 0 | 1 |
| 2 | 1 | 0.0159 | 0.9841 |
| 3 | 1 | 0.0317 | 0.9683 |
| 4 | 1 | 0.0476 | 0.9524 |
| 5 | 1 | 0.0635 | 0.9365 |
| 6 | 1 | 0.0794 | 0.9206 |
| 7 | 1 | 0.0952 | 0.9048 |
| 8 | 1 | 0.1111 | 0.8889 |
| 9 | 1 | 0.1270 | 0.8730 |
| 10 | 1 | 0.1429 | 0.8571 |
| 11 | 1 | 0.1587 | 0.8413 |
| 12 | 1 | 0.1746 | 0.8254 |
| 13 | 1 | 0.1905 | 0.8095 |
| 14 | 1 | 0.2063 | 0.7937 |
| 15 | 1 | 0.2222 | 0.7778 |
| 16 | 1 | 0.2381 | 0.7619 |
| 17 | 1 | 0.2540 | 0.7460 |
| 18 | 1 | 0.2698 | 0.7302 |
| 19 | 1 | 0.2857 | 0.7143 |
| 20 | 1 | 0.3016 | 0.6984 |
| 21 | 1 | 0.3175 | 0.6825 |
| 22 | 1 | 0.3333 | 0.6667 |

```
>> c2 = colormap('autumn');
```

c2 <64x3 double>

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 1 | 0.0159 | 0 |
| 3 | 1 | 0.0317 | 0 |
| 4 | 1 | 0.0476 | 0 |
| 5 | 1 | 0.0635 | 0 |
| 6 | 1 | 0.0794 | 0 |
| 7 | 1 | 0.0952 | 0 |
| 8 | 1 | 0.1111 | 0 |
| 9 | 1 | 0.1270 | 0 |
| 10 | 1 | 0.1429 | 0 |
| 11 | 1 | 0.1587 | 0 |
| 12 | 1 | 0.1746 | 0 |
| 13 | 1 | 0.1905 | 0 |
| 14 | 1 | 0.2063 | 0 |
| 15 | 1 | 0.2222 | 0 |
| 16 | 1 | 0.2381 | 0 |
| 17 | 1 | 0.2540 | 0 |
| 18 | 1 | 0.2698 | 0 |
| 19 | 1 | 0.2857 | 0 |
| 20 | 1 | 0.3016 | 0 |
| 21 | 1 | 0.3175 | 0 |
| 22 | 1 | 0.3333 | 0 |

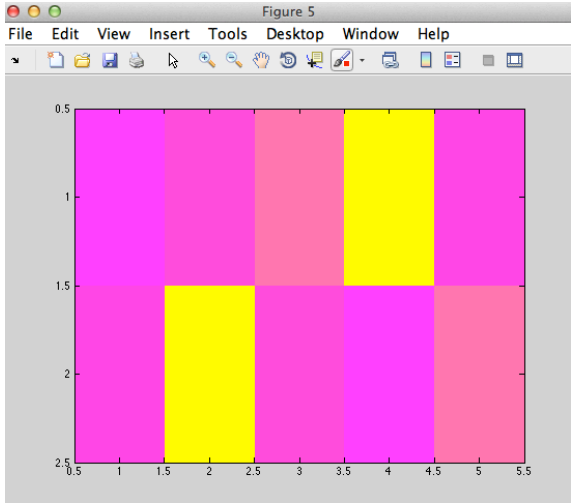
# Colormap

- The reason we don't see much of a color range is, the color map has many possible colors (e.g. 64 colors) so we are using only a subset of the colors.
- But we can easily scale the values of the data matrix to use the full color map.

```
>> help imagesc
```

Scale data and display as image.

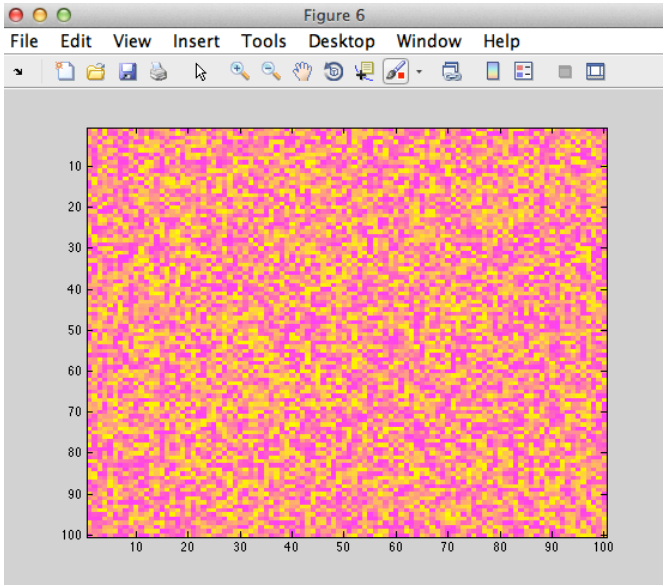
```
>> figure(5)
>> imagesc(x)
>> colormap('spring')
```



# Colormap

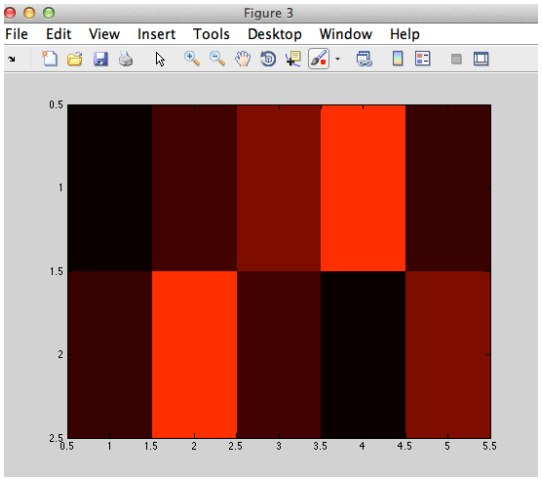
- Alternatively, we can create a new image matrix that uses a large range of values (between 0 and 64) in the first place.

```
>> img = round(rand(100,100)*63)+1;
>> figure(6)
>> image(img)
>> colormap('spring');
```



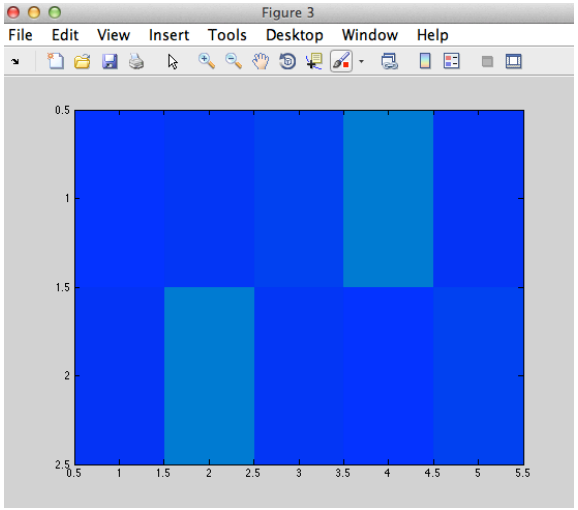
# Colormap

```
>> figure(3)
>> colormap('hot')
```




```
>> colormap(3, 'winter')
```

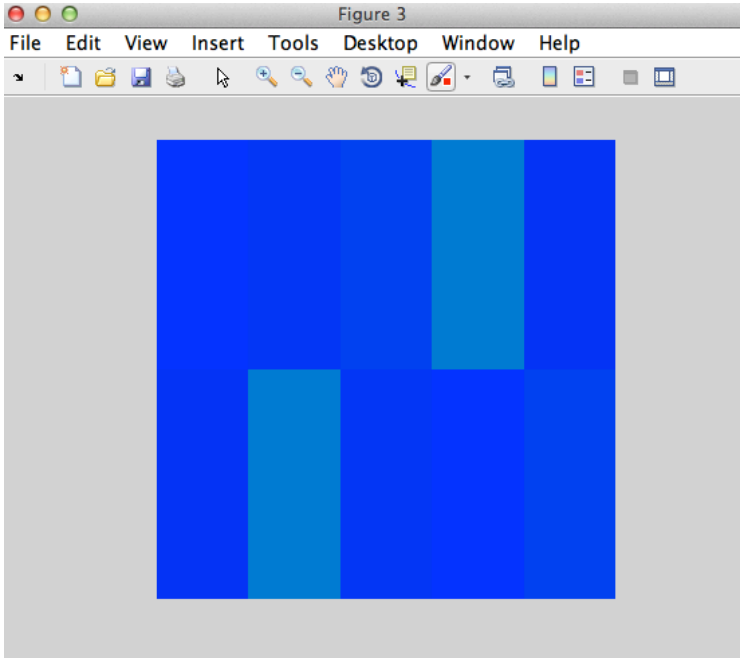
will change the figure whose handle is 3



# Change axis elements

```
>> help axis
```

```
>> figure(3);
>> axis off;
>> axis square;
```

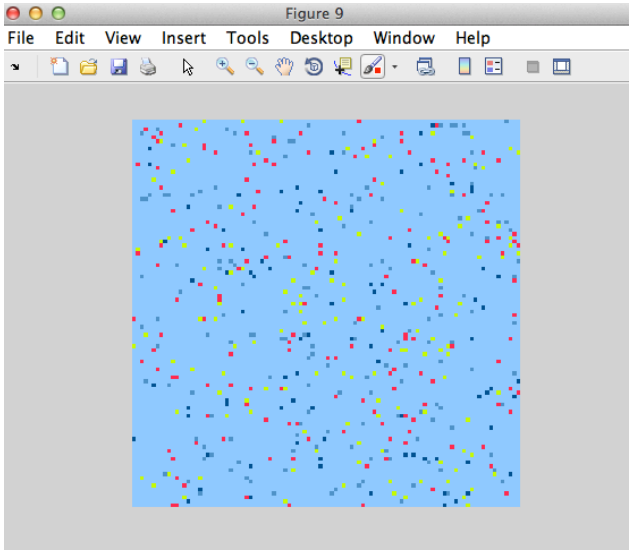


# User-defined colormaps

- We can also make our own colormaps.

```
>> mycmap1 = [0 0 0 ; 0.25 0.25 0.25 ; 0.5 0.5 0.5 ;
              0.75 0.75 0.75 ; 1 1 1];
```

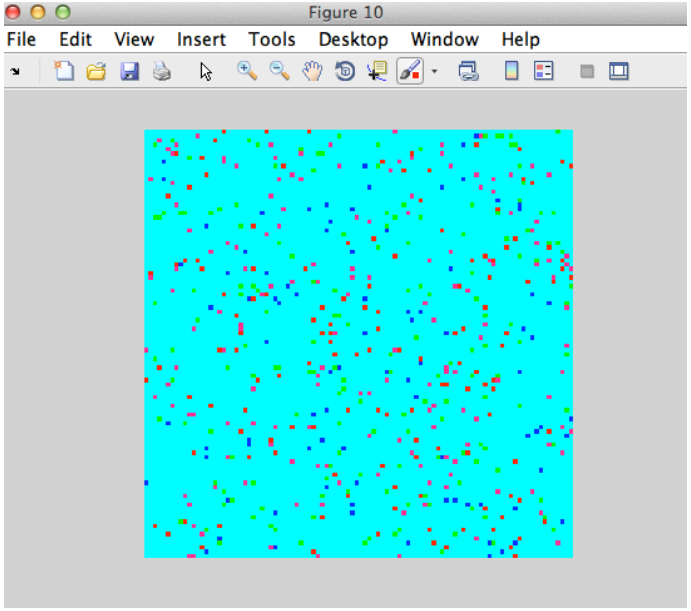
```
>> mycmap1 = reshape(mycmap1, 3 ,5 )';
>> figure(7);
>> axis off;
>> axis square;
>> image(img);
>> colormap(mycmap1);
```



# User-defined colormaps

```
>> mycmap2 = [0 0 1; 1 0 0; 0 1 0; 1 0 0.5 ;
                0 1 1];
>> figure(8);
>> axis off;
>> axis square;
>> image(img);
>> colormap(mycmap2);

>> close(figure(8));
```




Closes the figure window

# Let's play with colormaps

New .m file called mycolormaps.m

```
clear all; close all;
colormap(gray(256));
myimg = reshape(1:256,16,16);
image(myimg);
axis square;
axis off;
pause
figure(1);
for i = 1:200
      cmaps = rand(256,3);
      colormap(cmaps);
      drawnow
end
```

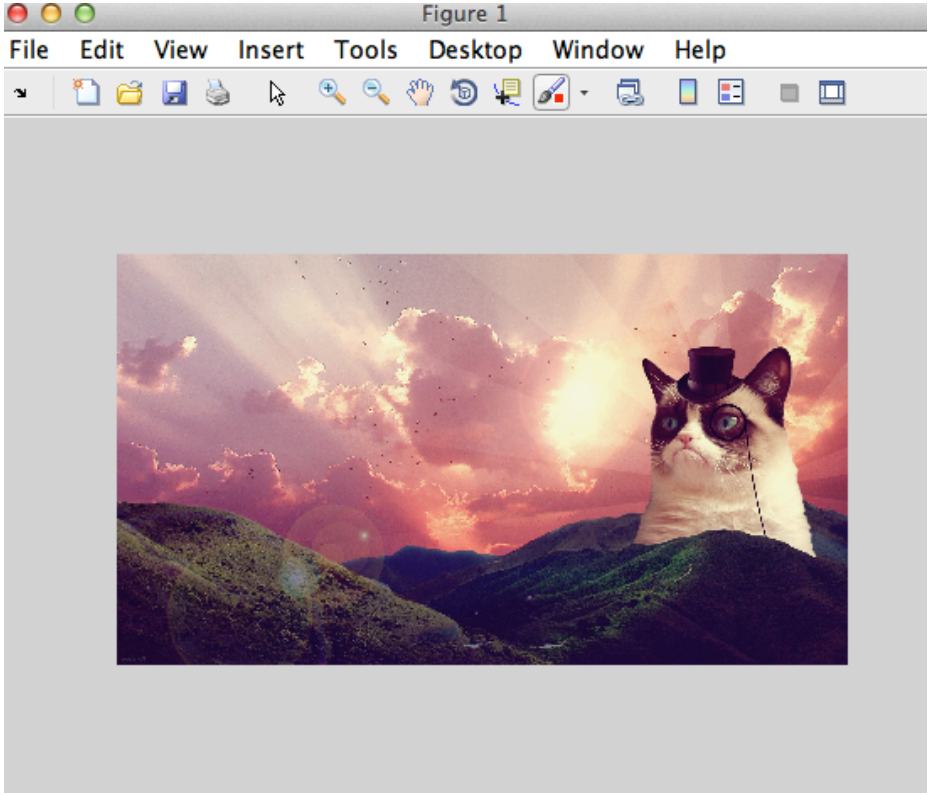
What does the code do?

# RGB or True-color images

- An RGB image is stored as an M x N x 3 array that defines red, green, and blue components for each individual voxel.
- The color of each pixel is determined by the combination of the red, green, and blue intensities stored in each color plane at the pixel's location.
- RGB images don't use a separate palette or colormap.
- Graphics file formats store RGB images as 24 bit images, where red, green and blue components are 8 bits each.
- This yields a potential of 16 million colors.

# Loading images

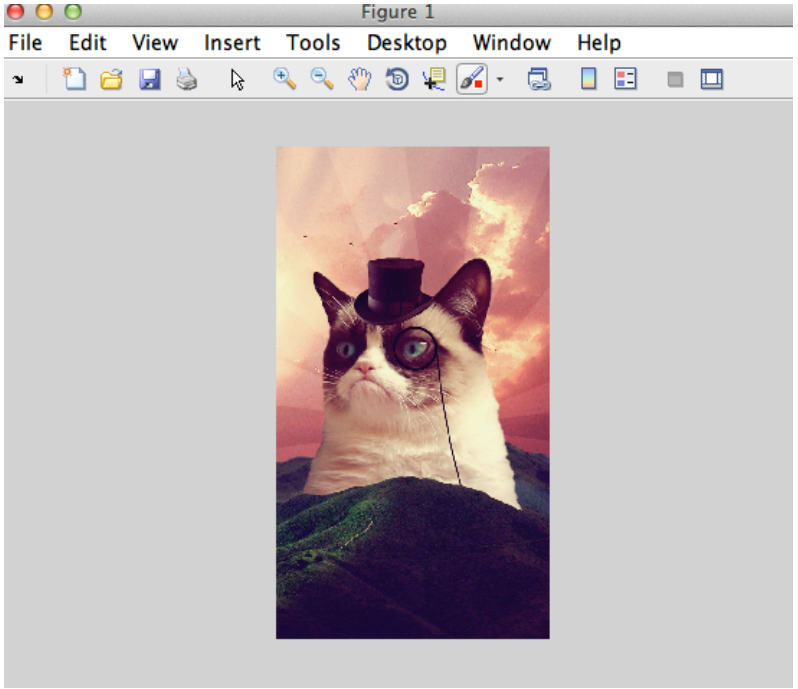
```
>> g1 = imread('grumpycat.jpg');
>> image(g1);
>> axis off;
>> axis image;
```



# Operations with images

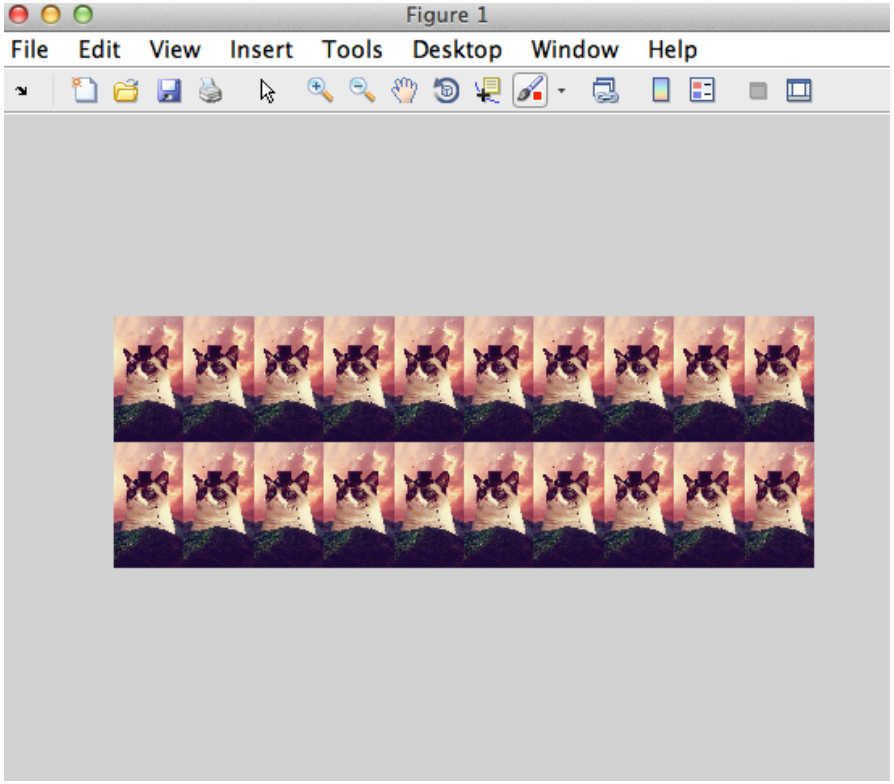
- You can do all matrix operations with images after they are loaded.

```
>> g3 = g1(:,550:800,:);
>> image(g3)
>> axis image
>> axis off
```



# Operations with images

```
g4 = repmat(g3,2,10);
>> image(g4)
>> axis image
>> axis off
```



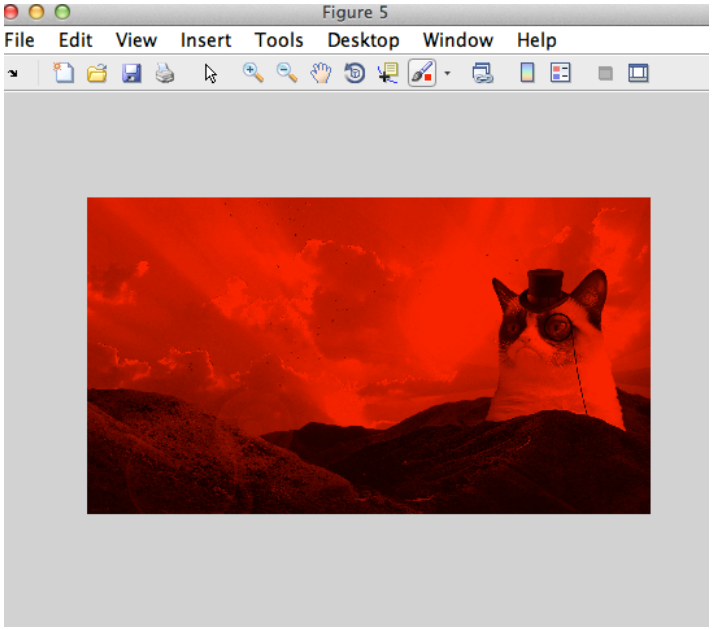
# RDB images classes

- An RGB array can be of class double, uint8 or uint16.
- Double:
  - Each color component is a value between 0 and 1.
  - A pixel whose color components are (0,0,0) is displayed as black.
  - A pixel whose color components are (1,1,1) is displayed as white.

- Unsigned integer
  - Colors will be represented by integers 0 to 255 (for a total of 256 which 2^8)
  - The three color components for each pixel are stored along the third dimension of the data array.
  - For example, the red, green, and blue color components of the pixel (24,78) in g1 are stored in g1(24,78,1), g1(24,78,2), and g1(24,78,3), respectively.

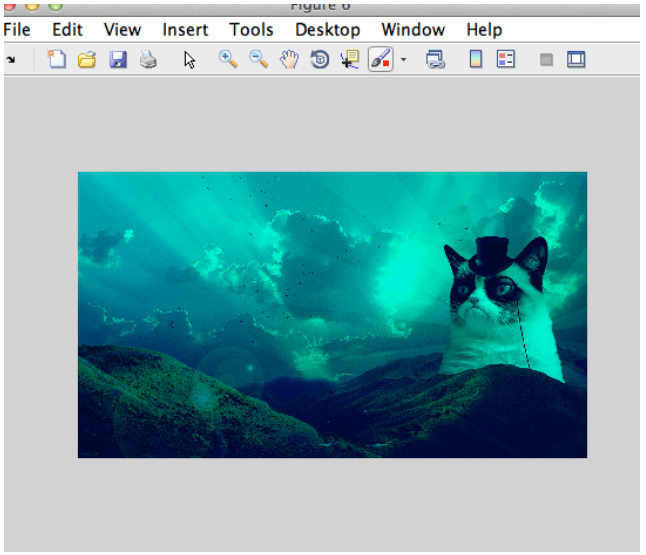
# Manipulate colors

```
>> g5 = g1;
>> g5(:,:,3) = 0;    % no blues
>> g5(:,:,2) = 0;   % now no greens either
>> figure(5); image(g5);
>> axis off; axis image;
```

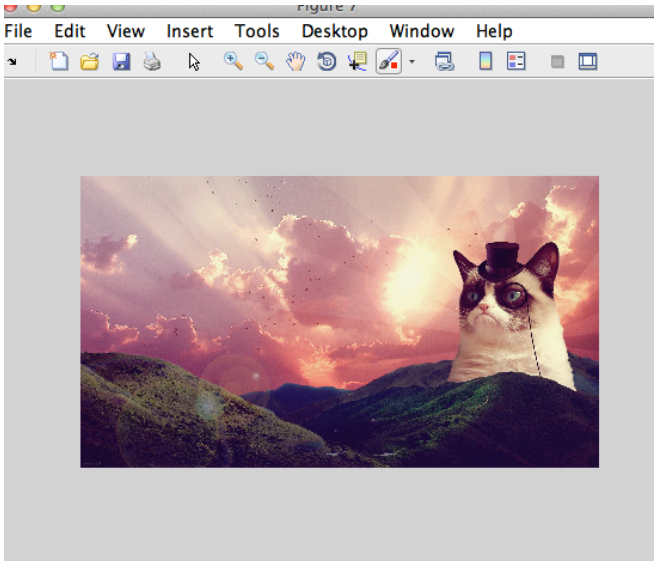


# Manipulate colors

```
>> g6 = g1;
>>  g6(:,:,1) = 0;  % no red
>> figure(6); image(g6);
>> axis off; axis image;
```

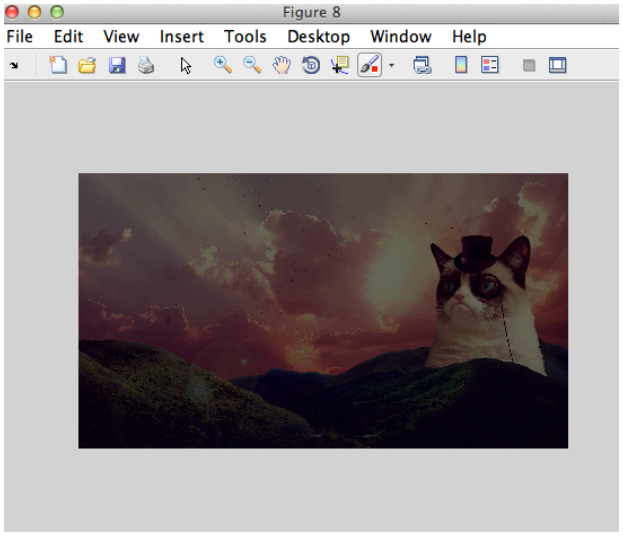



```
>> g7 = g5+g6;
>> figure(7); image(g7);
>> axis off; axis image;
```

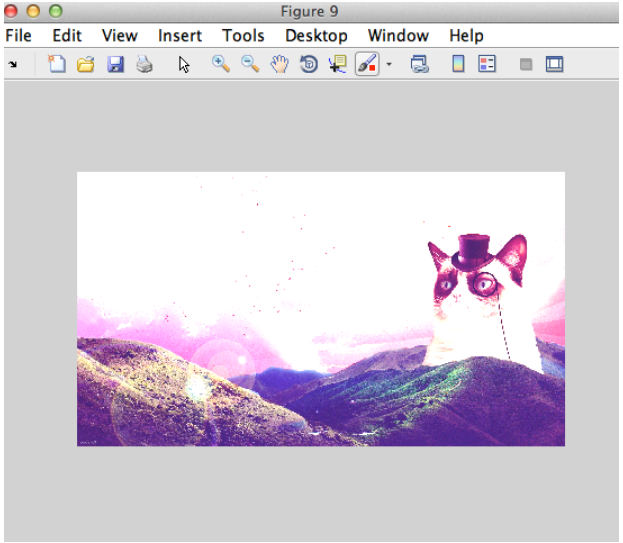


# Manipulate brightness

```
>> g8 = g1/3;
>> figure(8); image(g8);
>> axis off; axis image;
```

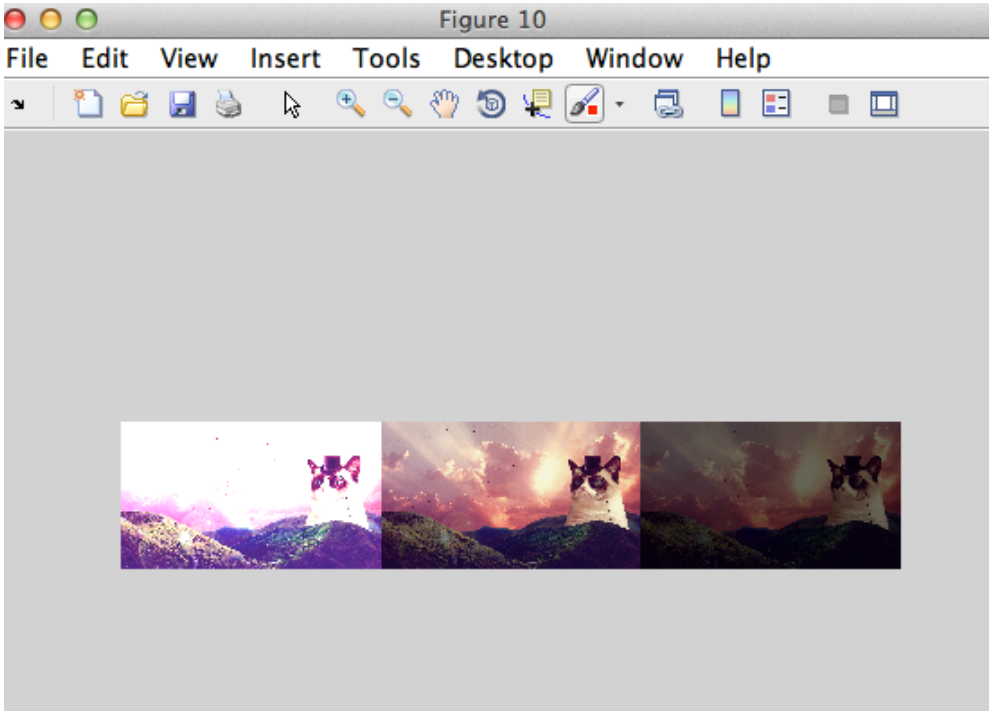



```
>> g9 = g1*3;
>> figure(9); image(g9);
>> axis off; axis image;
```



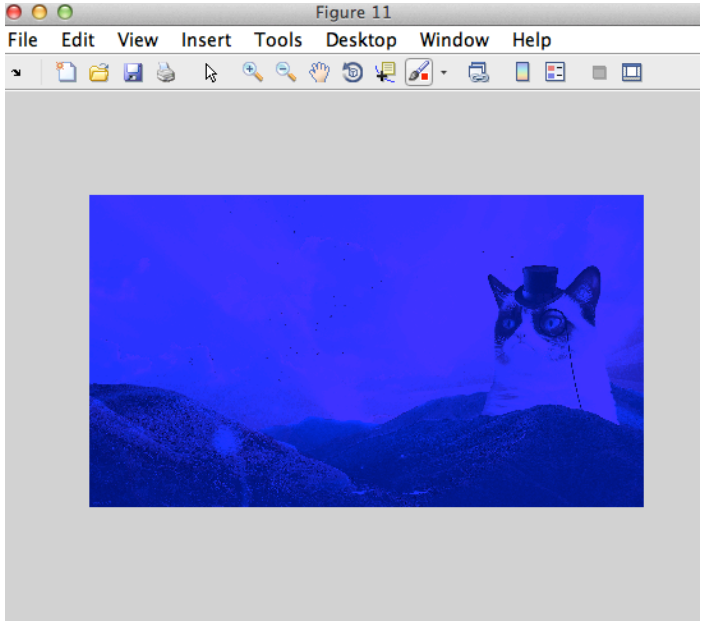
# Manipulate brightness

```
>> g10 = [g9 g1 g8];
>> figure(10); image(g10);
>> axis image; axis off;
```



# Manipulate both color and brightness

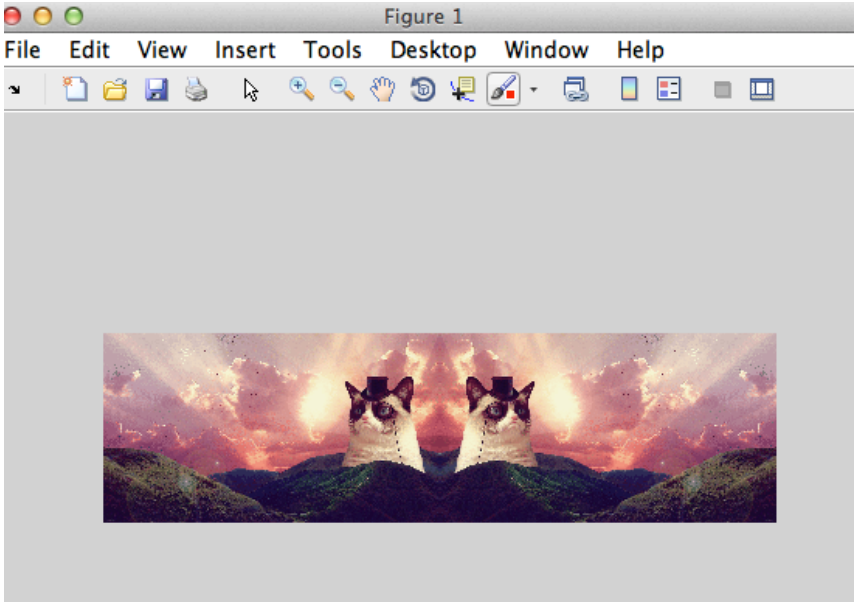
```
>> g11 = g1;
>> g11(:,:,3) = g1(:,:,3) * 3;
>> g11(:,:,2) = 0;  % no greens
>> g11(:,:,1) = g1(:,:,1)/5;
>> figure(11); image(g11);
>> axis off; axis image;
```




Exercise

# Convert RGB image to indexed image

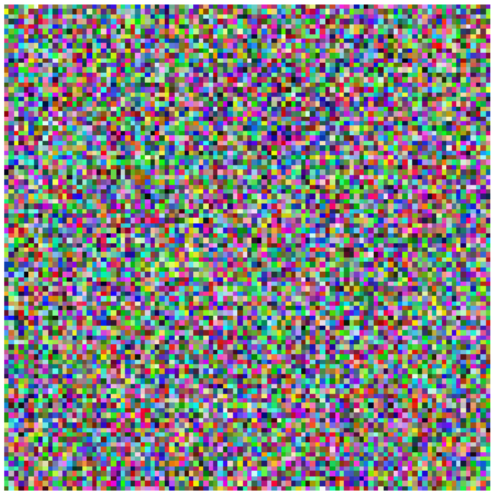
```
>> g1 = imread('grumpycat.jpg');
>> [indg1 cmapg1] = rgb2ind(g1,256);
>> indg1flip = fliplr(indg1);
>> indg1both = [indg1 indg1flip];
>> figure;imagesc(indg1both); axis off; axis image
>> colormap('gray');
>> colormap(cmapg1);
```



# Convert indexed image to RGB image

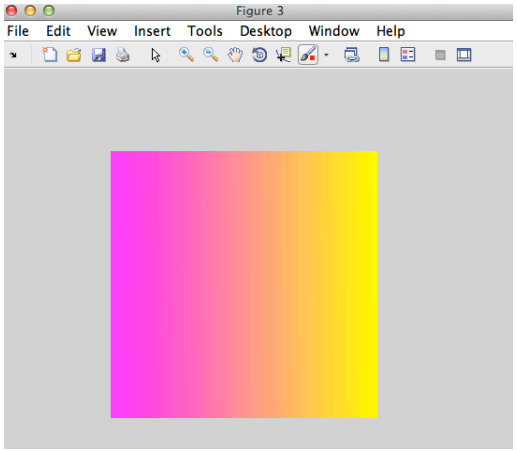
```
>> myimg = reshape(1:10000,100,100);
>> cmap = rand(10000,3);
>> figure; image(myimg); axis off; axis square;
>> colormap(cmap)

>> newrgbimg = ind2rgb(myimg,cmap);
>> figure(2);image(newrgbimg);
>> axis square; axis off;
```

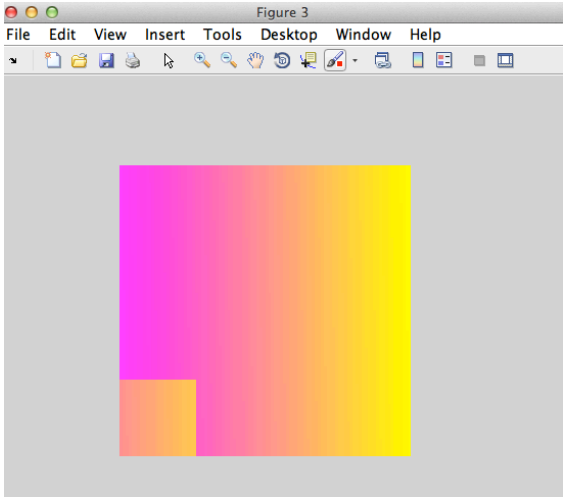


# hold on

```
>>figure(3);
>> hold on;
>> imagesc(myimg)
>> axis square ; axis off
>> colormap('spring');
```

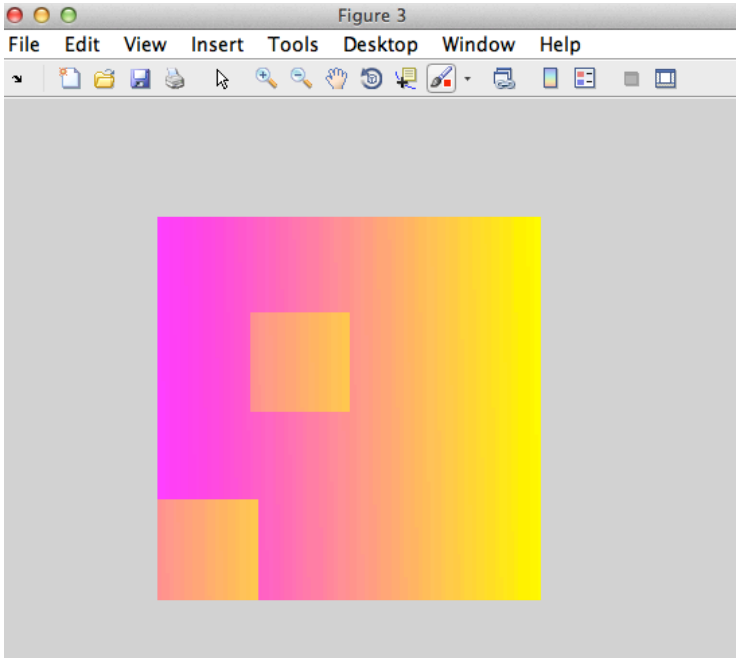



```
>> myimg2 = myimg(50:75, 50:75);
>> imagesc(myimg2);
```



# hold on

```
>> imagesc(25,50,myimg2);
>> hold off;
```

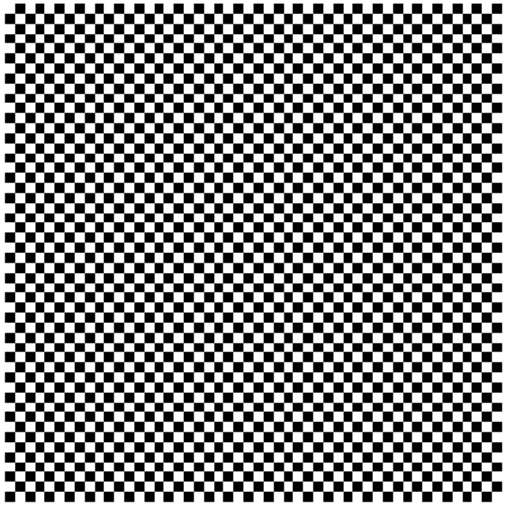


# Exercise

- Create matrix A, which is a 50 x 50 matrix that has a randomly generated number on the odd rows and odd columns and zeros elsewhere (hint: one way to do it us use **rand** and **eye**).

- It should look like the following when displayed using figure and **imagesc** (axis off, axis square) (hint: you may need to change the colormap).

- Save the image (from the figure window, or using saveas) as YourLastName_checkerboard.jpg.



# Exercise

```
>> x = eye(2);
>> x = rand() * x;
>> y = repmat(x, 25, 25);
>> figure;
>> imagesc(y);
>> axis off; axis square;
>> colormap('gray')
```