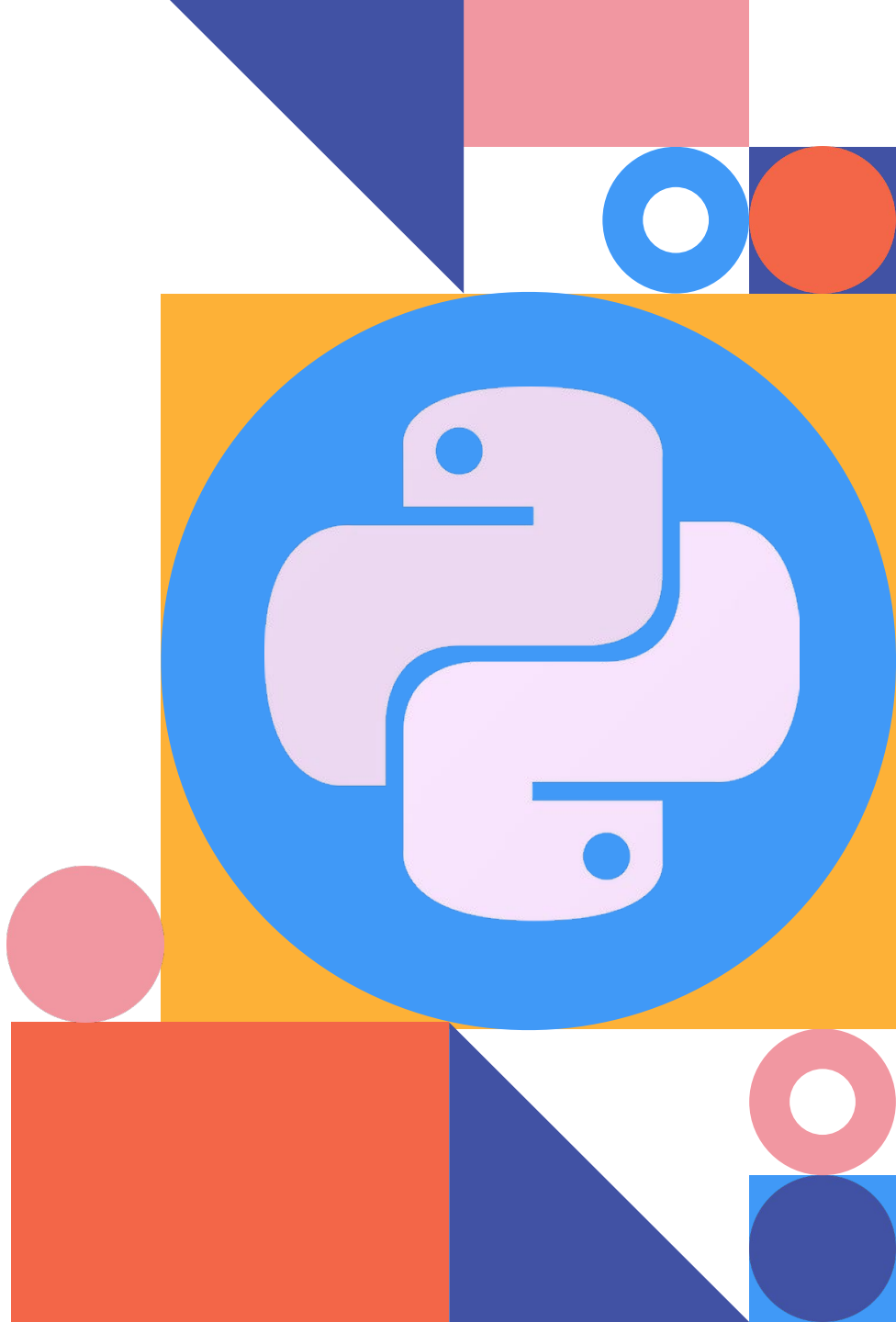
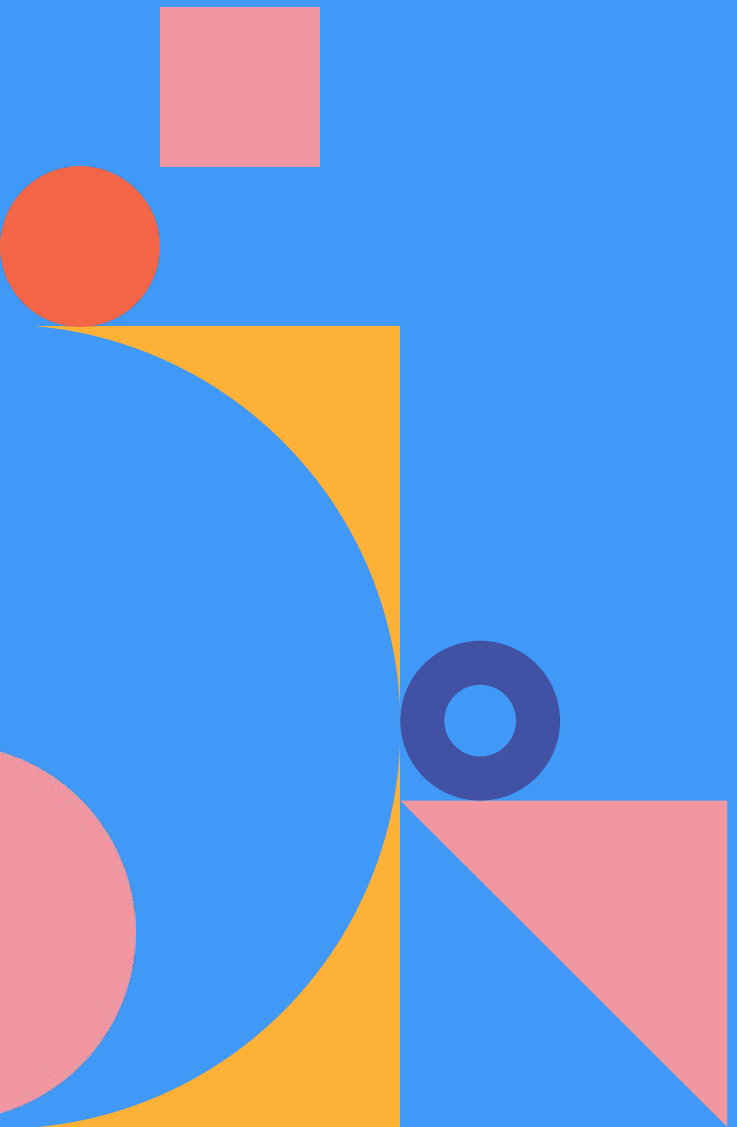
Image Processing I

Yan-Fu Kuo
Dept. of Biomechatronics Engineering
National Taiwan University

→



Contents



01 Digital Image

Image Processing in Python

Scikit-image

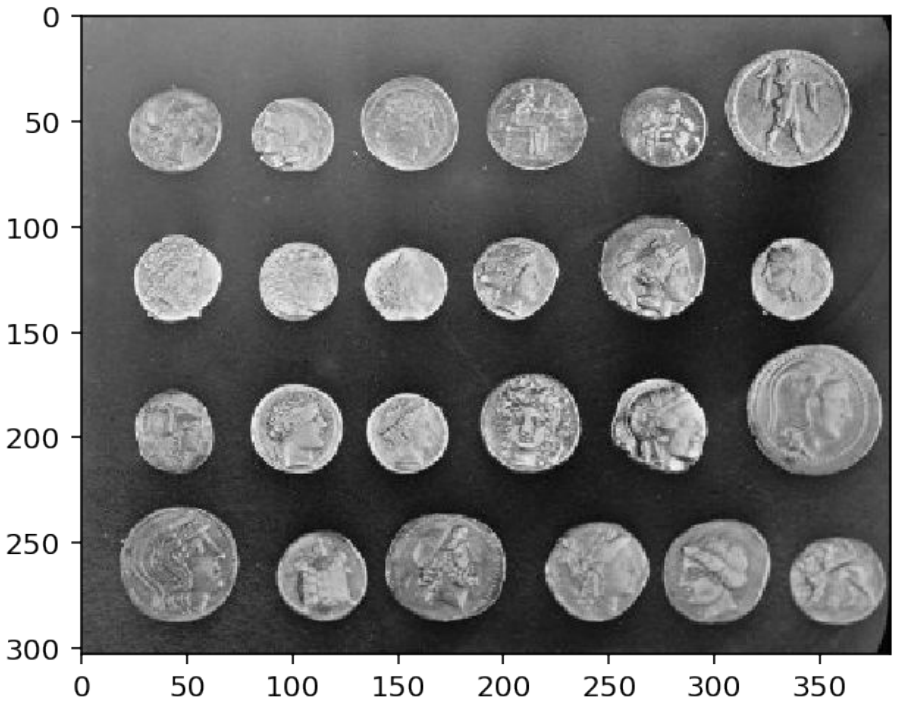
- Simple and efficient tools for image processing and computer vision techniques.
- Built on the top of NumPy, SciPy, and matplotlib.
- Open source, commercially usable – BSD license.

```
$ pip install scikit-image
```

Digital Image in skimage

How are images stored in Python with the skimage library?

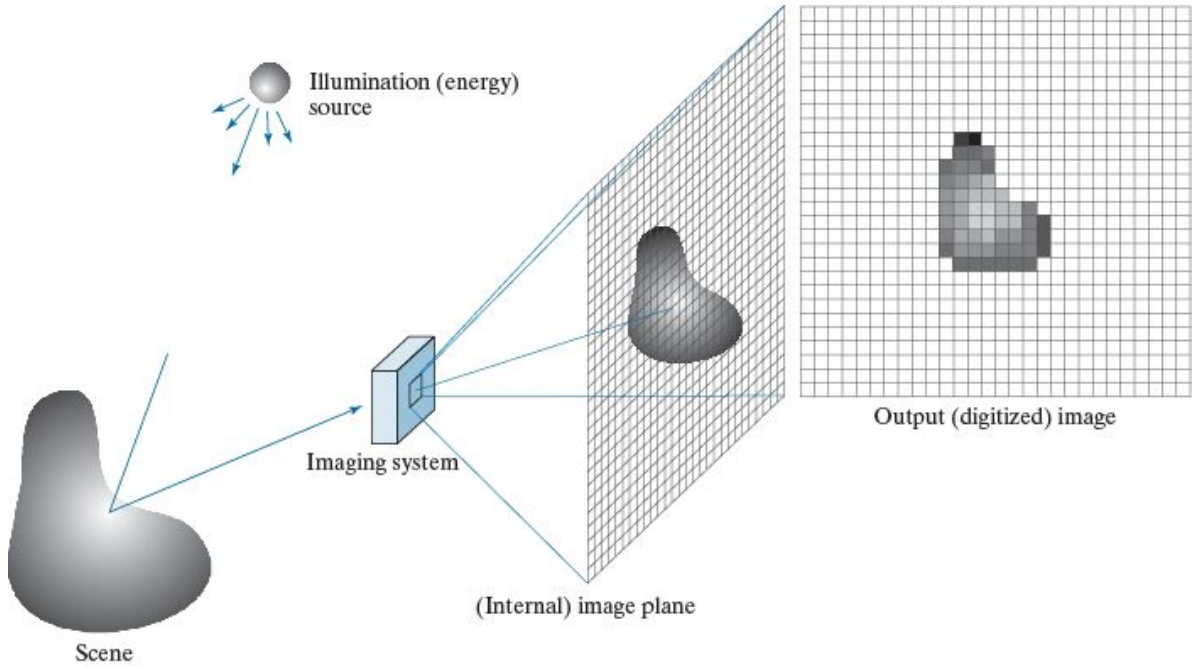
```
from skimage import data
import matplotlib.pyplot as plt

coins = data.coins()

print('Type: ', type(coins))
print('dtype: ', coins.dtype)
print('shape: ', coins.shape)

plt.imshow(coins, cmap='gray')
plt.show()
```

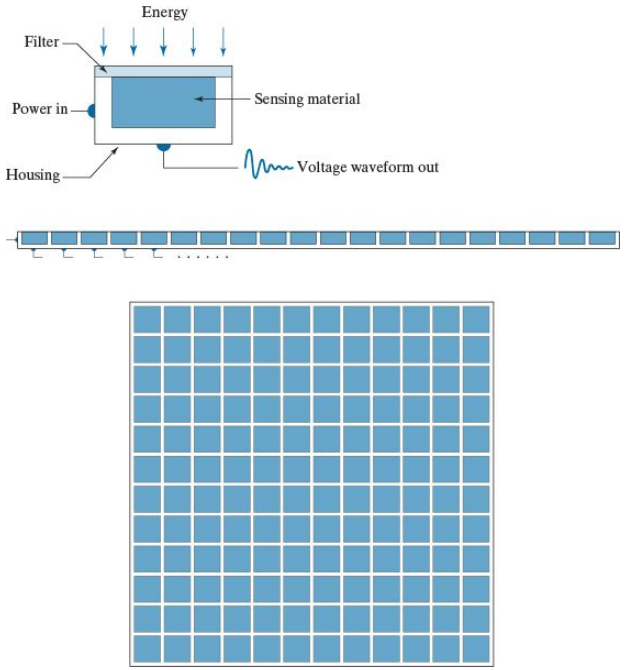


Digital Image

146	155	144	130	145	151	143	127	131	150	158	150	131	119	133	145	148
142	153	150	128	131	151	161	146	126	136	155	151	132	132	134	129	121
131	141	142	130	128	148	167	157	134	143	164	163	150	139	126	105	96
122	123	125	127	130	135	152	150	143	165	184	173	156	128	106	79	82
130	123	107	118	150	154	145	151	173	191	196	179	145	118	102	92	79
127	120	125	143	153	161	165	174	203	216	211	192	145	102	107	112	100
121	123	152	178	169	178	201	214	225	212	200	192	146	100	121	127	120
132	153	188	214	203	204	237	249	224	181	168	172	140	133	144	139	133
175	200	223	242	242	233	243	243	206	163	154	148	126	153	165	156	148
222	229	238	255	255	246	213	192	173	158	151	132	116	152	166	165	157

Digital Image Acquisition





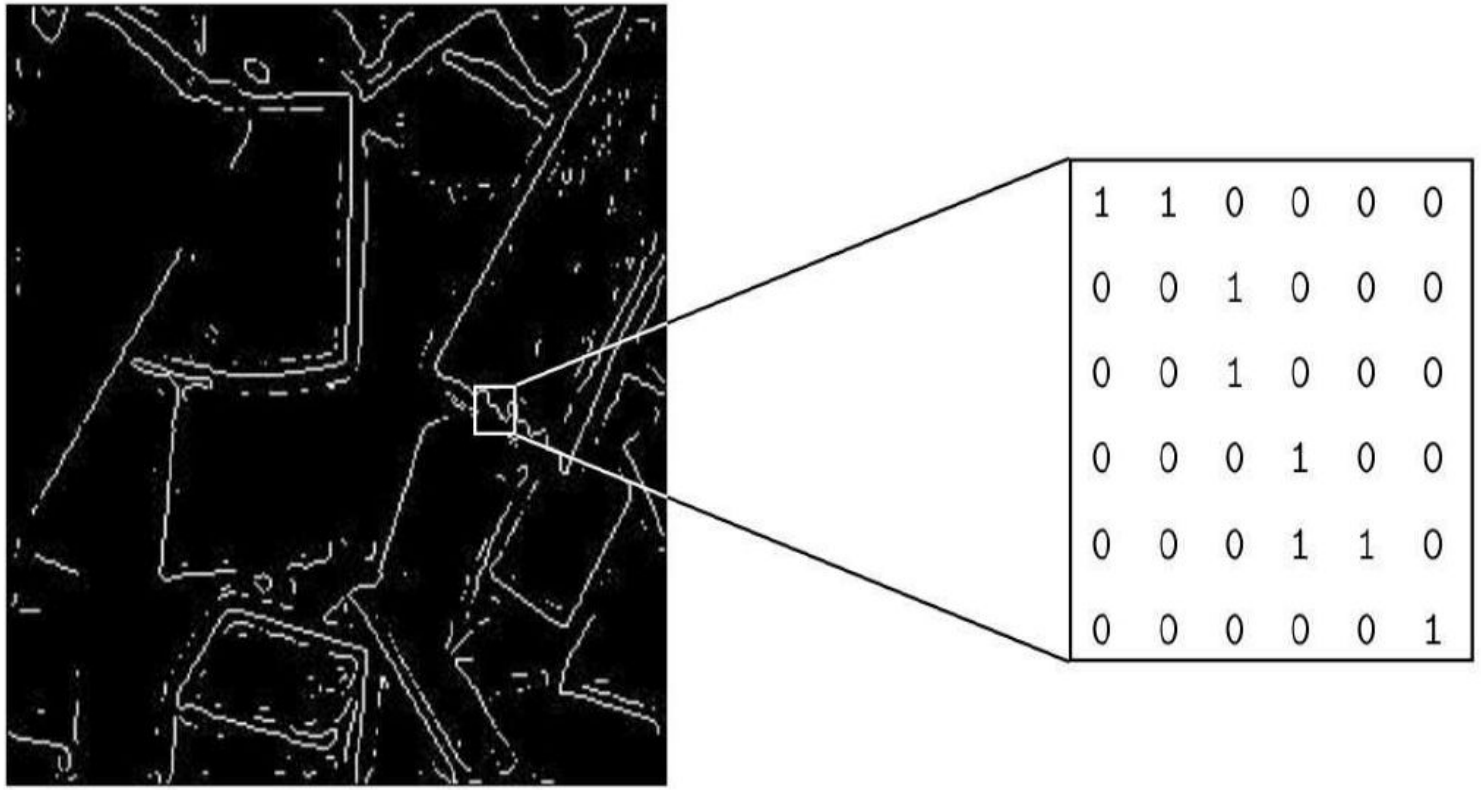
Types of Digital Image

Binary

Grayscale

RGB

Binary Image



Grayscale Image

230	229	232	234	235	232	148
237	236	236	234	233	234	152
255	255	255	251	230	236	161
99	90	67	37	94	247	130
222	152	255	129	129	246	132
154	199	255	150	189	241	147
216	132	162	163	170	239	122

RGB (Color) Image

49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

Red

64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Green

66	80	77	80	87	77
81	93	96	99	86	85
83	83	91	94	92	88
135	128	126	112	107	106
141	129	129	117	115	101
95	99	109	108	112	109
84	93	107	101	105	102

Blue

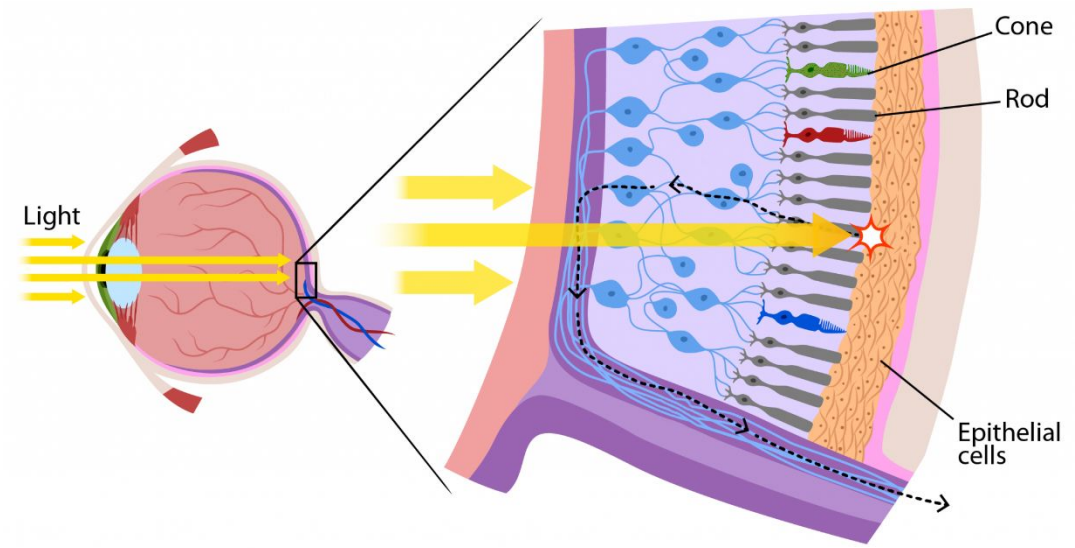
Quiz

Which one is G, which one is B, and which one is R?

The Rainbow Tunnel, Toronto, Ontario

Quiz

RGB is an acronym for "Red Green Blue," and it refers how colors are composed. But why red, green, and blue?



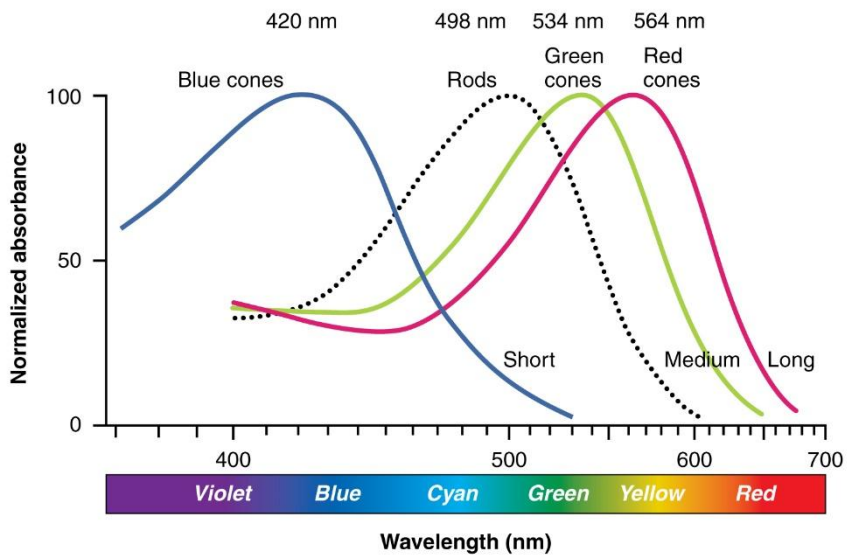


Image Processing

```
cat[10:110, 10:110, :] = [255, 0, 0]
```

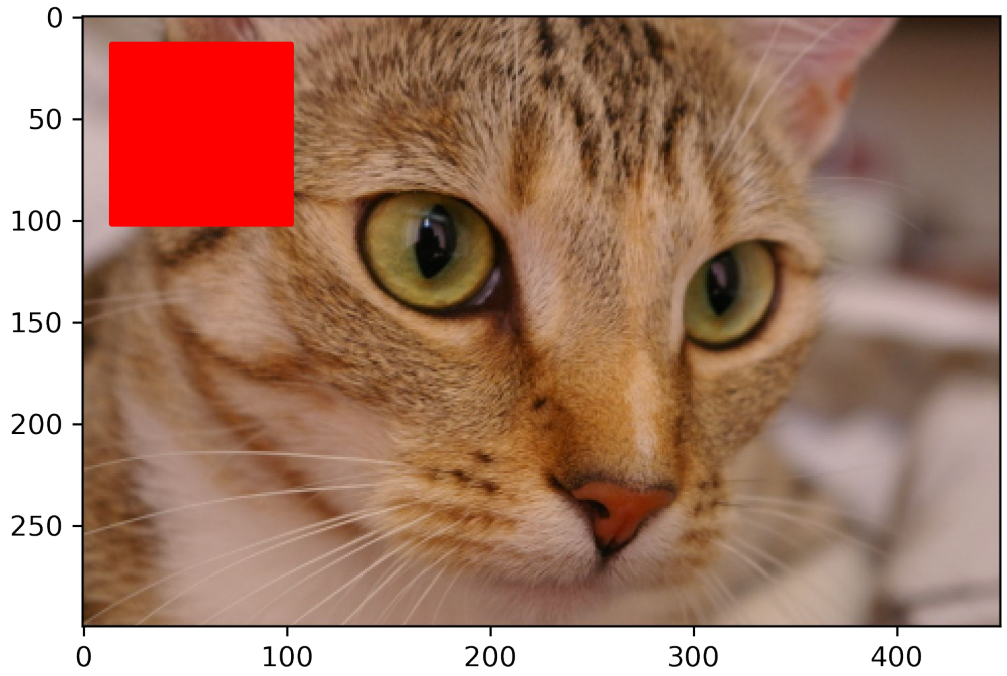


Image I/O

Reading an image

```
from skimage import io

deer = io.imread('deer.jpg')
gray_deer = io.imread('deer.jpg', as_gray=True)
```

Writing an image

```
io.imsave('deer1.jpg', deer)
io.imsave('deer2.png', gray_deer)
```

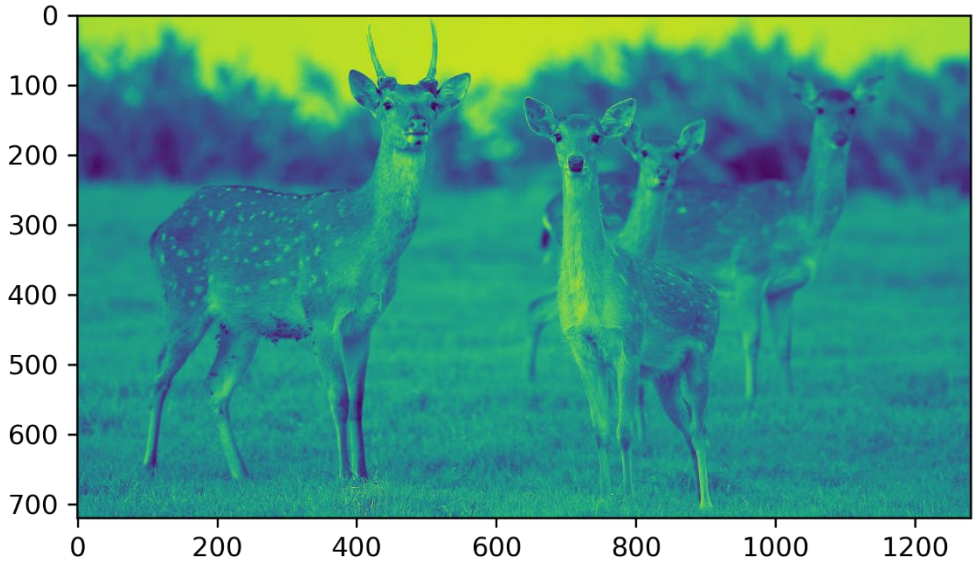
`imsave()` function automatically determines the type of the file, based on the file extension we provide.

Displaying Images

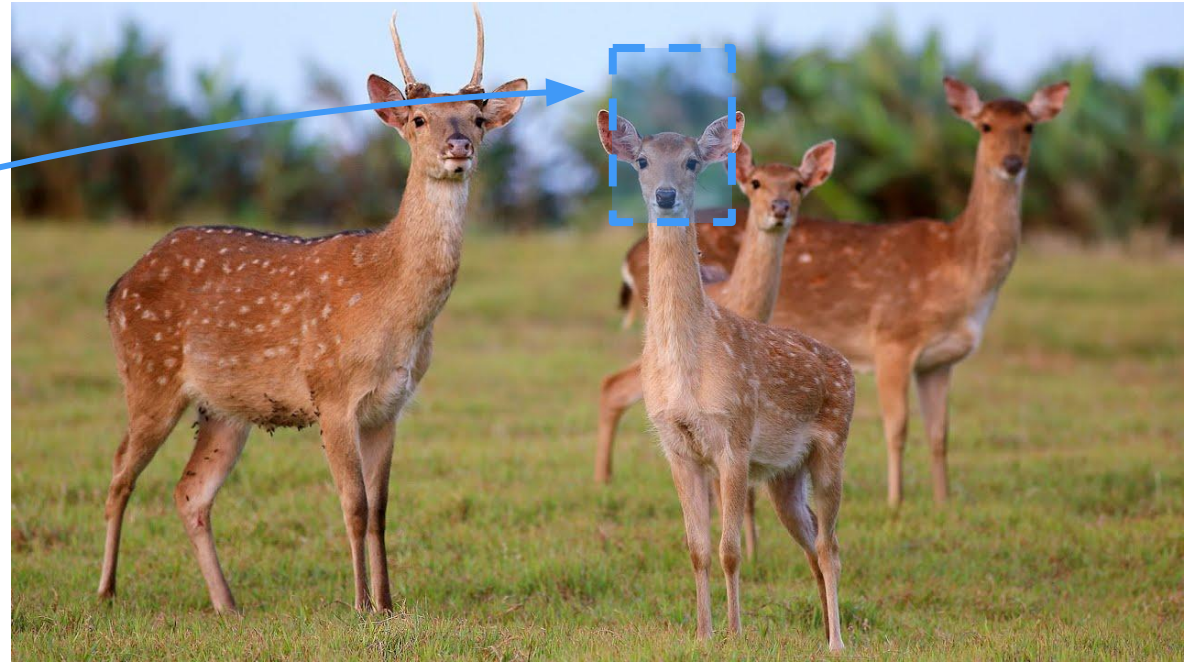
```
import matplotlib.pyplot as plt
plt.imshow(deer)
plt.show()
```

```
plt.imshow(gray_deer)
plt.show()
```

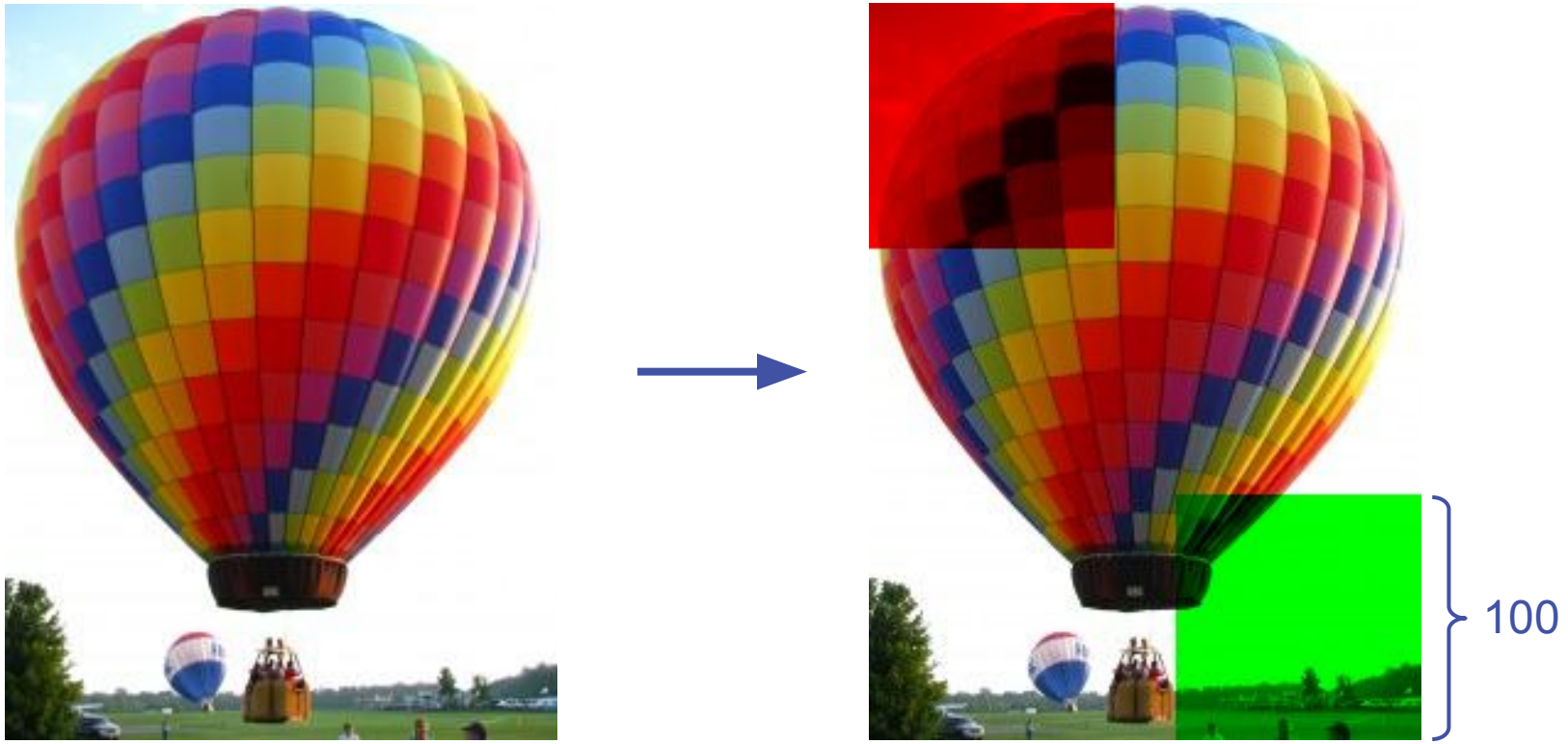
```
plt.imshow(gray_deer, cmap='gray')
plt.show()
```



Exercise 1 (5 mins)



Exercise 2 (5 mins)



Exercise – 3 GRB to Grayscale (5 mins)

The relative luminance of an image is the intensity of light coming from each point. Different colors contribute differently to the luminance: it's very hard to have a bright, pure blue, for example. So, starting from an RGB image, the luminance is given by:

① $$Y = 0.2126R + 0.7152G + 0.0722B$$

② $$Y = \frac{1}{3}R + \frac{1}{3}G + \frac{1}{3}B$$

Compare your results to that obtained with `skimage.color.rgb2gray`. ③

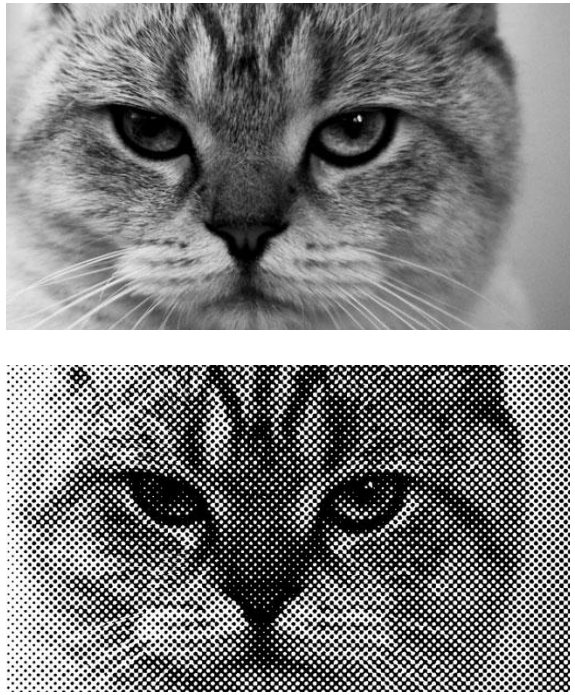
Change the coefficients to 1/3 (i.e., take the mean of the red, green, and blue channels, to see how that approach compares with rgb2gray).

02 Mathematical Tools

Image Processing

Any form of signal processing for which the input is an image

Halftone



Deblur

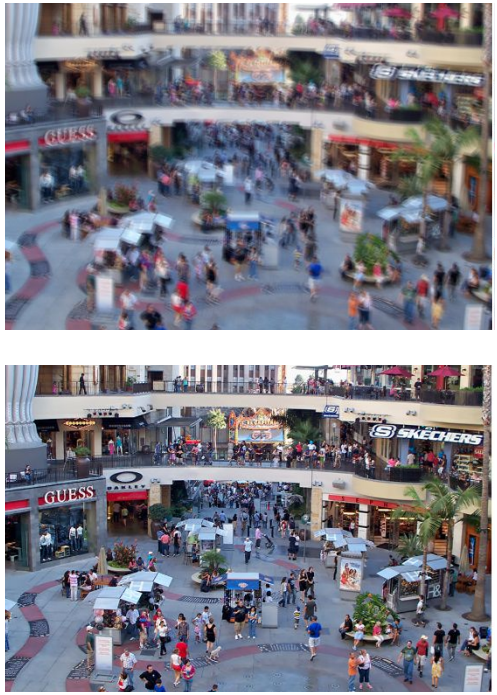


Image Arithmetic

Name	Operator	Equivalent func.
Addition	+	np.add
Subtraction	-	np.subtrat
Multiplication	*	np.multiply
Division	/	np.divide
Modulus	%	np.mod
Exponentiation	**	np.power
Floor Division	//	np.floor_divide

Image Addition

```
camera = data.camera()
cmaera50 = camera + 10
```

Changing the pixels values adjust the brightness of the image.

Any problem with “`image+10`”?
Try “`image+50`”.

Integers can overflow

Integer data types are 8-, 16- or 32-bytes, signed or unsigned.

The value of an integer overflow if it is added or multiplied with a large number.

Strategy: convert integer to float.

```
from skimage import img_as_float
camera_float = img_as_float(camera)
print(camera.max(), camera_float.max())
```

Image Multiplication

```
camera_float_15 = camera_float*1.5
plt.imshow(camera)
plt.imshow(camera_float_15)
plt.imshow(camera_float_15, vmax=1)
```

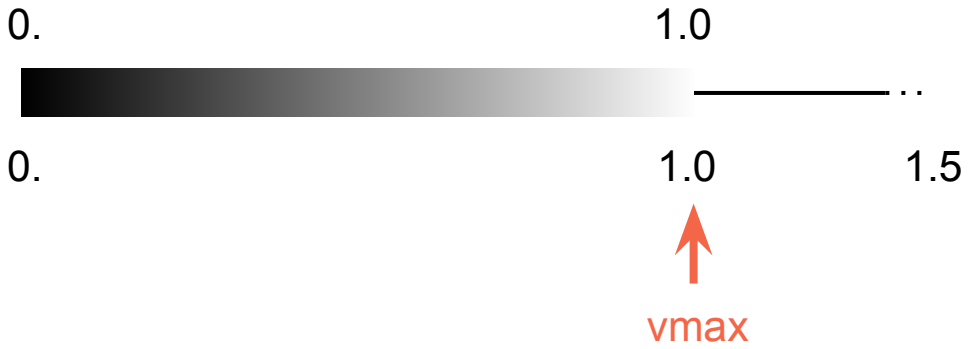


Image Addition

Image Subtraction

Detecting changes between two images, or levelling uneven sections of an image such as half an image having a shadow on it.

-

Geometric Transformation

Moving the coordinates (not the gray-levels) of the pixels in an image.

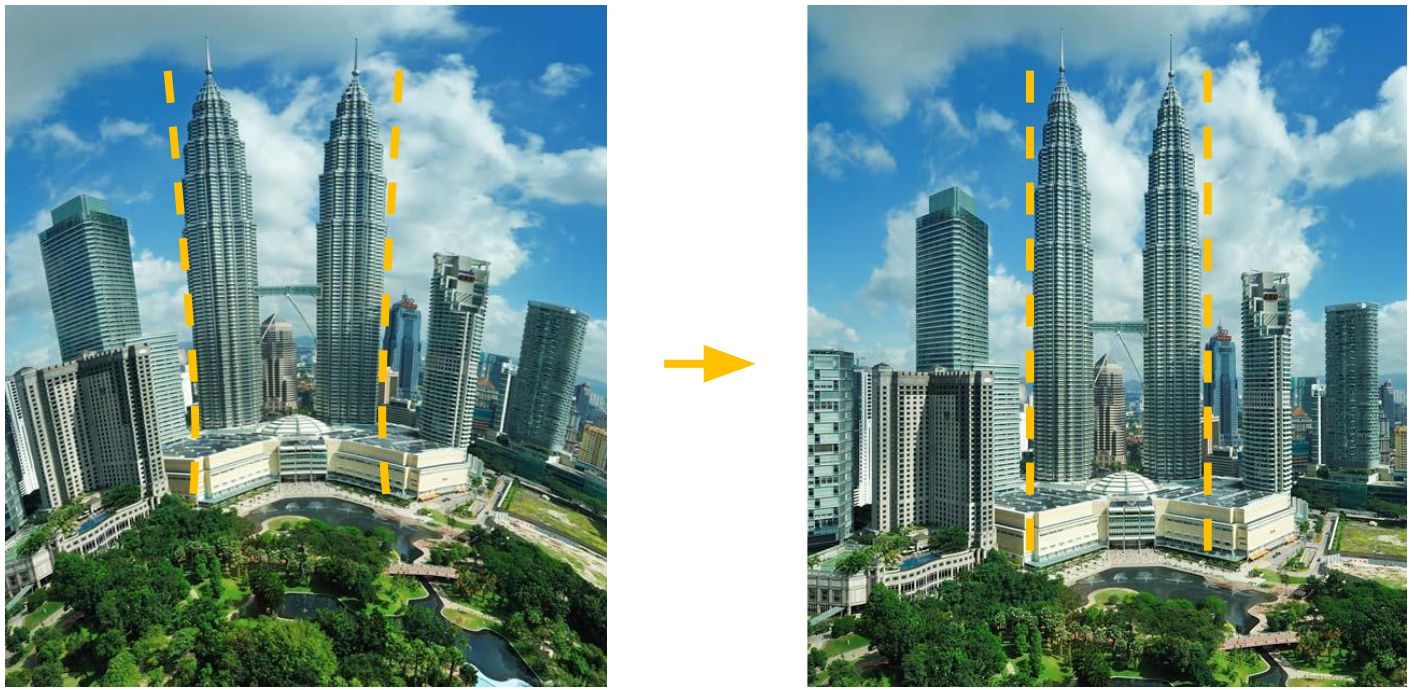


Image Rotation

```
from skimage import transform, data
camera = data.camera()
rotate_camera = transform.rotate(camera, 30, resize=False, order=1)
rotate_camera = transform.rotate(camera, 30, resize=False, order=1)
rotate_camera_resize = transform.rotate(camera, 30, resize=True, order=1)
```

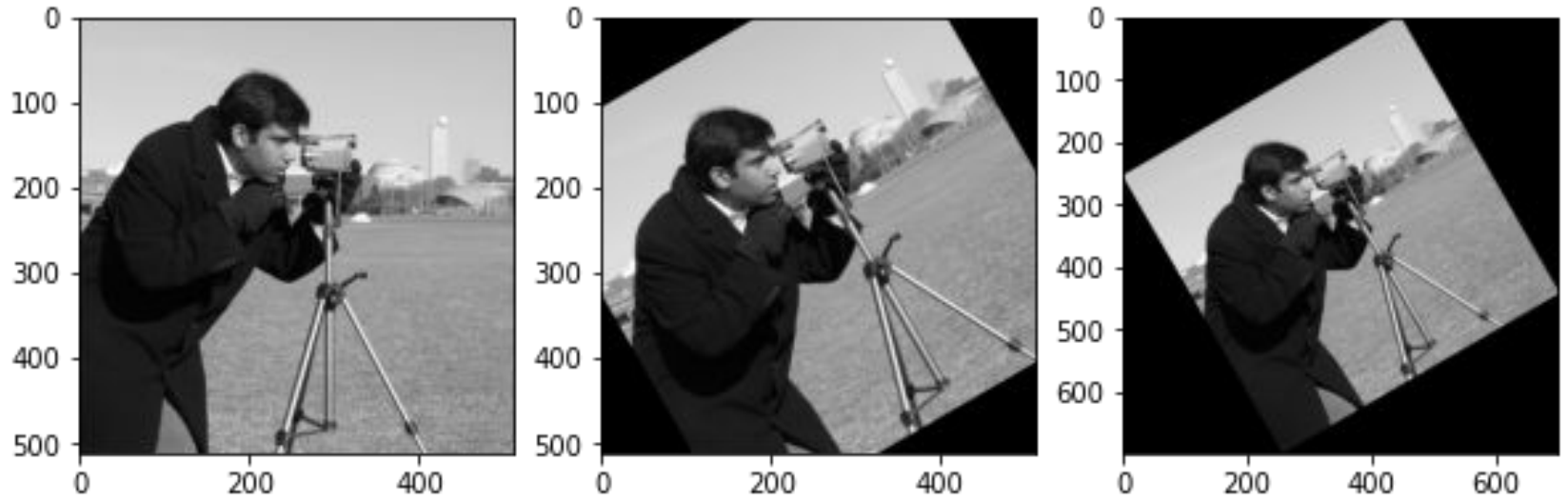
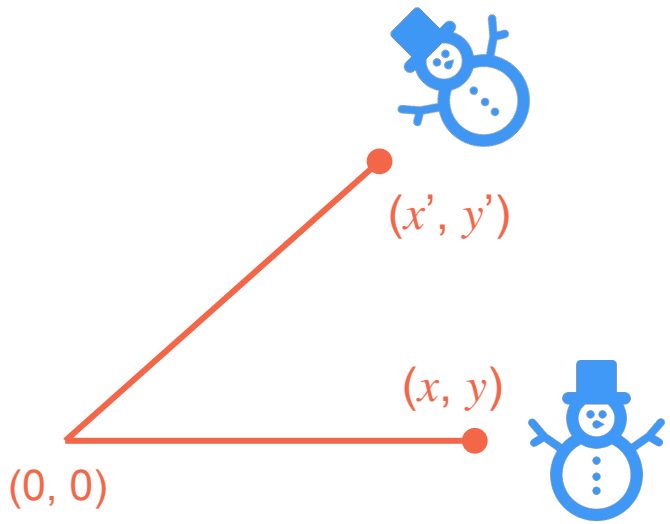


Image Rotation

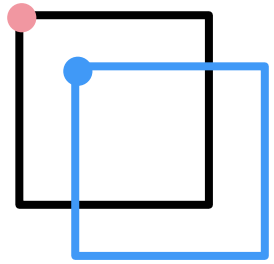
In two dimensions, rotation of a point (x, y) for an angle θ "counter-clockwise" can be written as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



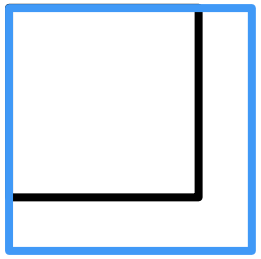
Affine Transformation Matrices

Translation



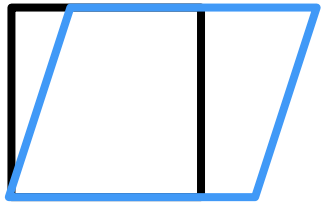
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scale



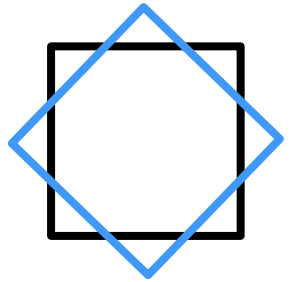
$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear



$$\begin{bmatrix} 1 & h_x & 0 \\ h_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation



$$\begin{bmatrix} cos\theta & sin\theta & 0 \\ -sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

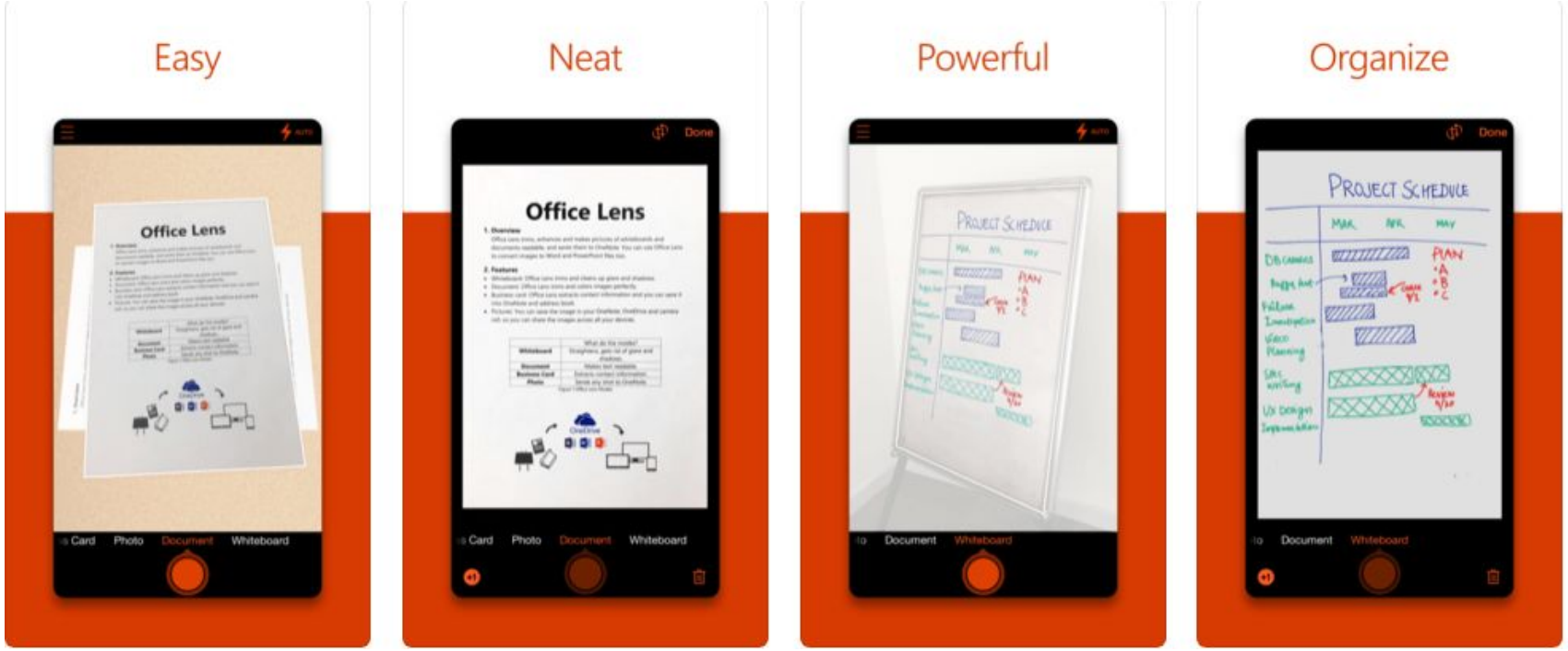
transform.AffineTransform()

```
tform = transform.AffineTransform(
            scale=(2, 4),
            rotation=-np.pi/3,
            shear=np.pi/6,
            translation=(-400, 200)
        )
```

$$\begin{bmatrix} 1 & 2 & -400 \\ -1.732 & 3.464 & 200 \\ 0 & 0 & 1 \end{bmatrix}$$

```
tf_camera = transform.warp(camera, tform)
```

Projective Transformation

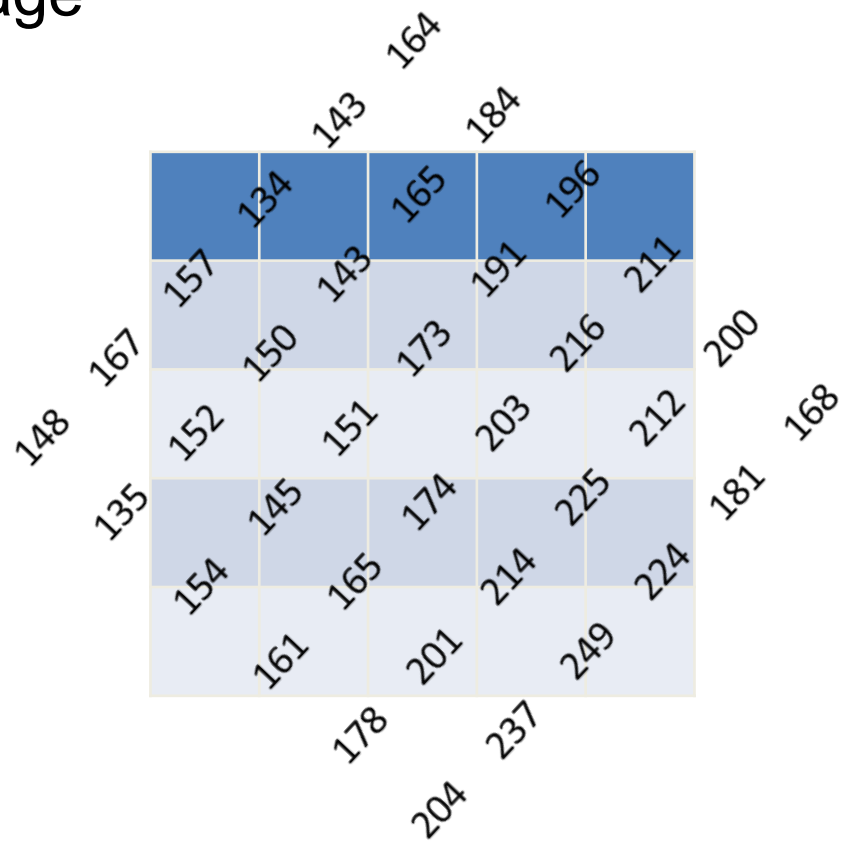


Try `transform.ProjectiveTransform()`

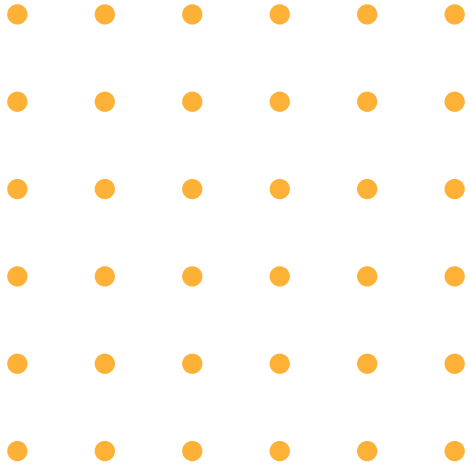
Exercise | Rotation

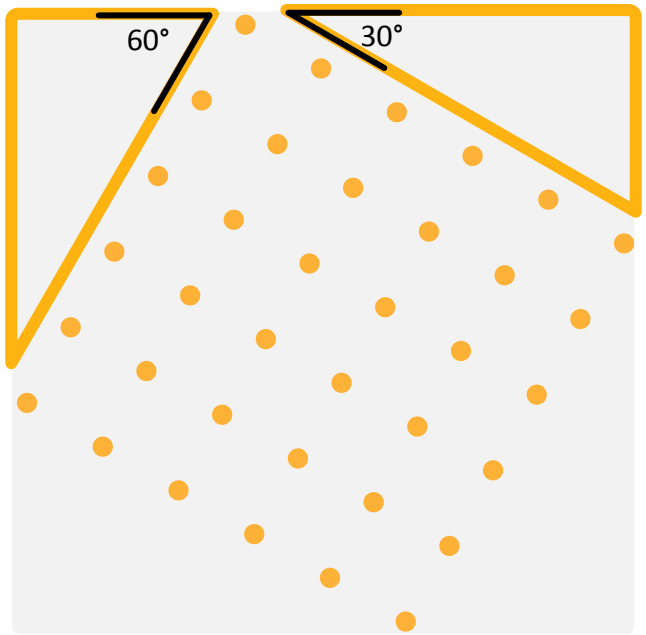
Implement a function to rotate an image

148	167	157	134	143	164
135	152	150	143	165	184
154	145	151	173	191	196
161	165	174	203	216	211
178	201	214	225	212	200
204	237	249	224	181	168



Solution | step1

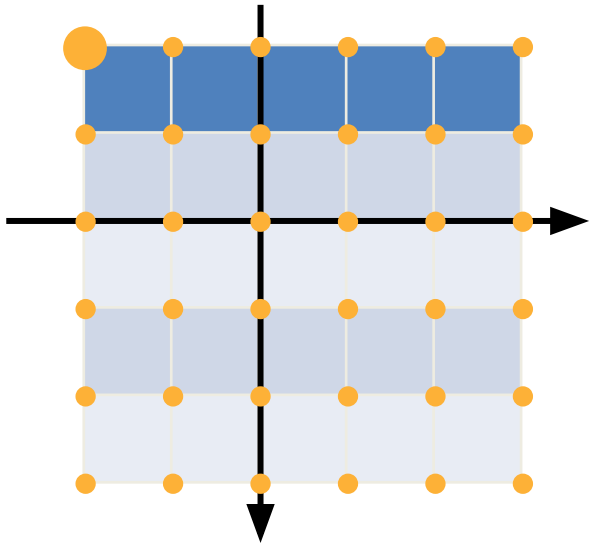


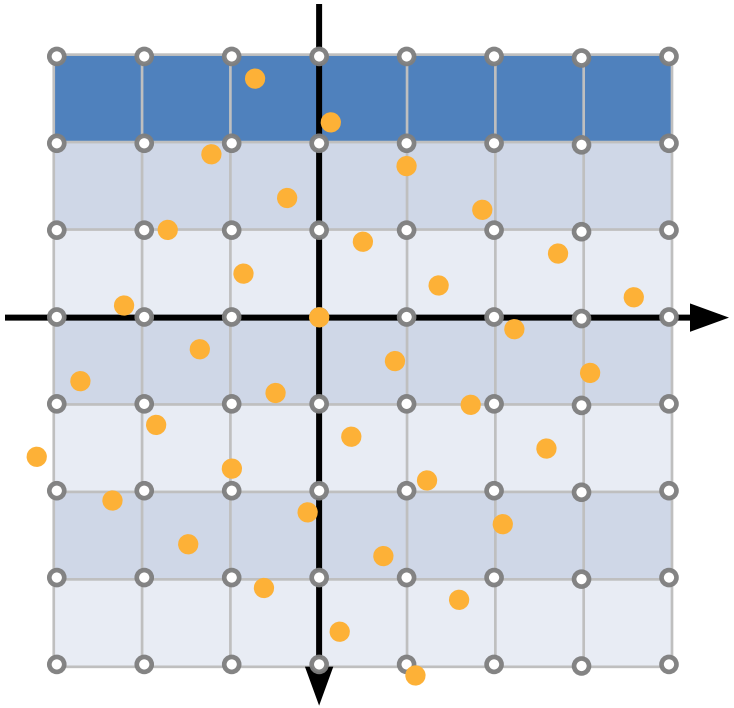


Solution | step2

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} -0.732 \\ -2.732 \end{bmatrix} = \begin{bmatrix} cos30 & -sin30 \\ sin30 & cos30 \end{bmatrix} \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$





Solution | step2

$$\begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}^{-1} \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$