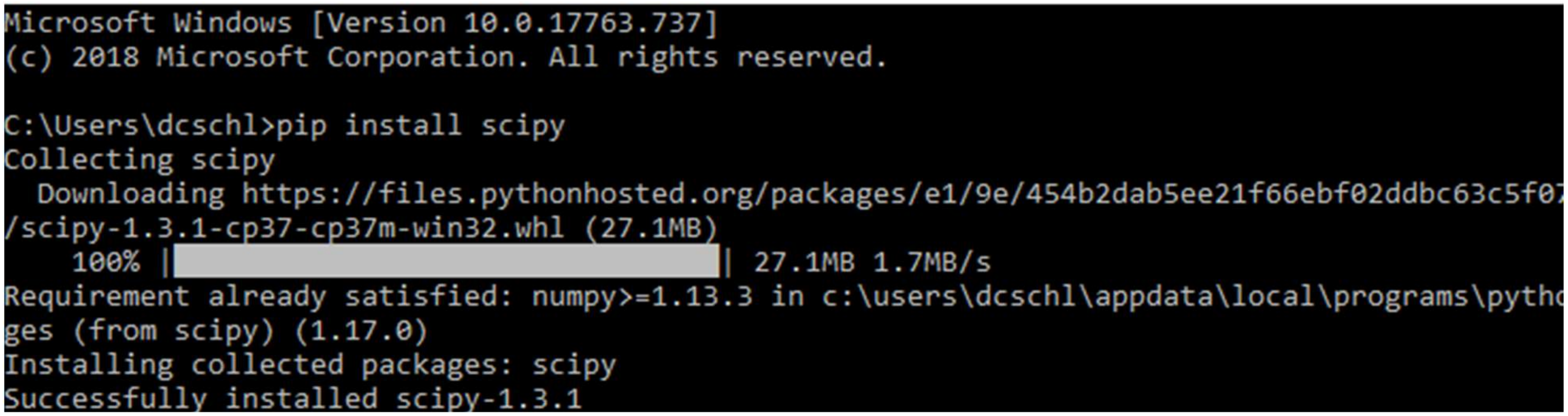
Image Processing

Installing Python Packages

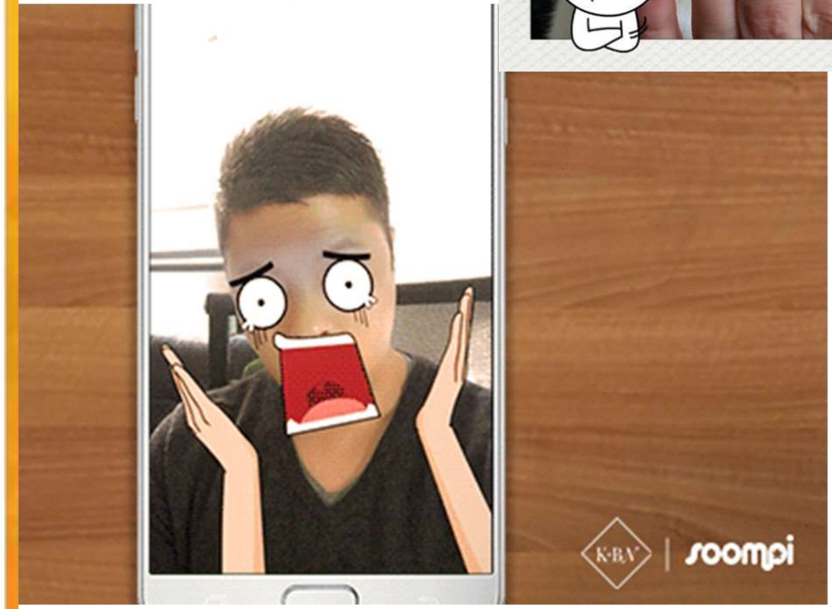
- Python comes with built-in functions
- However, you need to manually install additional packages
  - In Assignment 0, the instructions asked you to install, imageio, numpy, etc
- In this lecture we will need “imageio”
  - To install “imageio” (or any other packages), go to cmd.exe
    - Type “`pip install imageio`”

- Provided you have your internet connected
- **`pip`** will download the package and install it for you



```
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\dcschl>pip install scipy
Collecting scipy
  Downloading https://files.pythonhosted.org/packages/e1/9e/454b2dab5ee21f66ebf02ddbc63c5f07
/scipy-1.3.1-cp37-cp37m-win32.whl (27.1MB)
    100% |████████████████████████████████| 27.1MB 1.7MB/s
Requirement already satisfied: numpy>=1.13.3 in c:\users\dcschl\appdata\local\programs\pytho
ges (from scipy) (1.17.0)
Installing collected packages: scipy
Successfully installed scipy-1.3.1
```

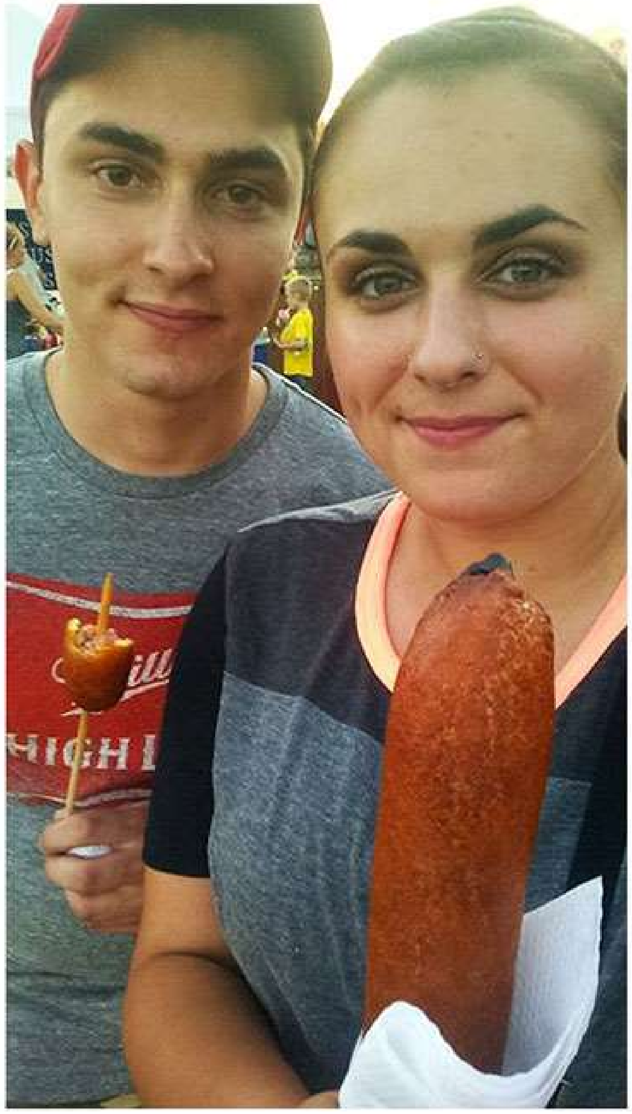
We have all these photo apps



https://www.everydayfamily.com/slideshow/10-hilariously-awful-photoshop-fails/

M[redacted]il.com>
to me

Hey, just wondering if you could edit this photo of me and my boyfriend. I was hoping you could make his corn dog whole again... with no bites taken out... thanks!



James Fridman <fjamie013@gmail.com>

Done.

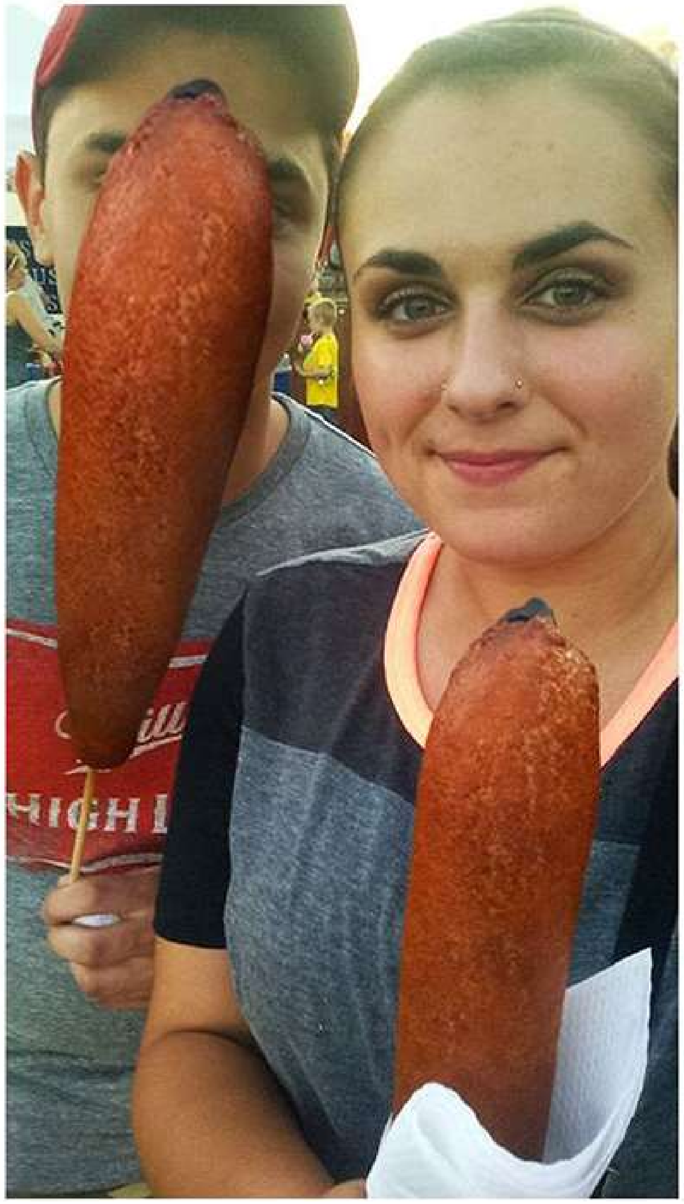


Image Processing

- To load an image, you can use the package “imageio”

```
import imageio
import matplotlib.pyplot as plt

cat_pic = imageio.imread('cute cat.jpg')

plt.imshow(cat_pic)
plt.show()
```

```
>>> type(cat_pic)
<class 'imageio.core.util.Array'>
>>> cat_pic.shape
(600, 600, 3)
>>>
```

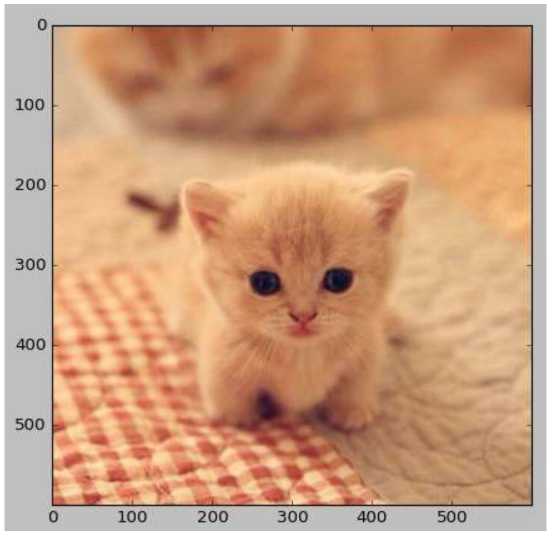
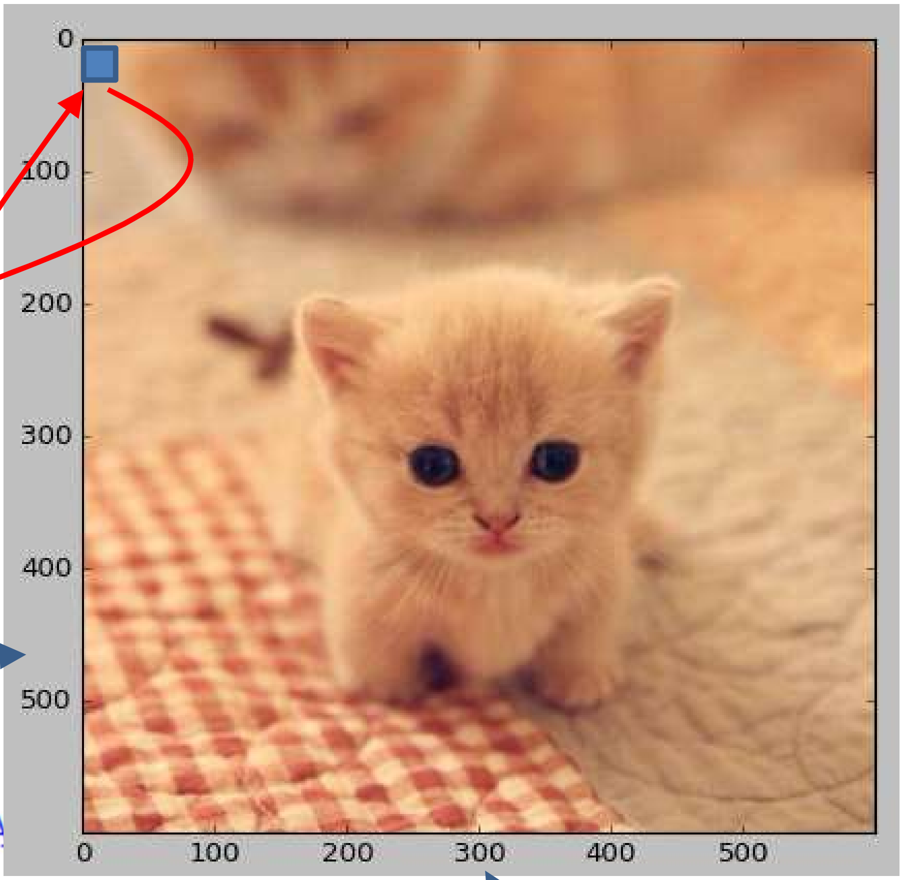
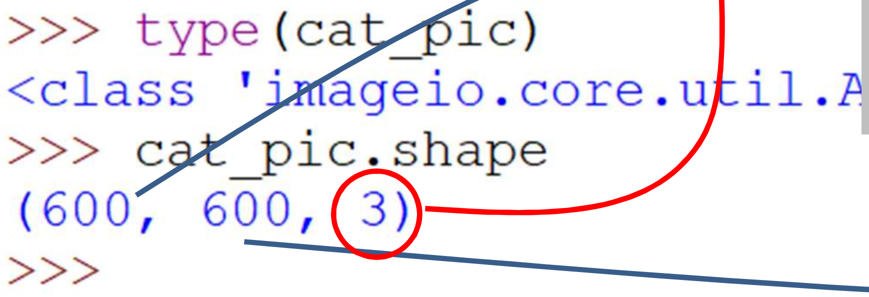


Image Processing

- 600 x 600 pixel
  - And each pixel has three values of R, G and B
  - [R, G, B]

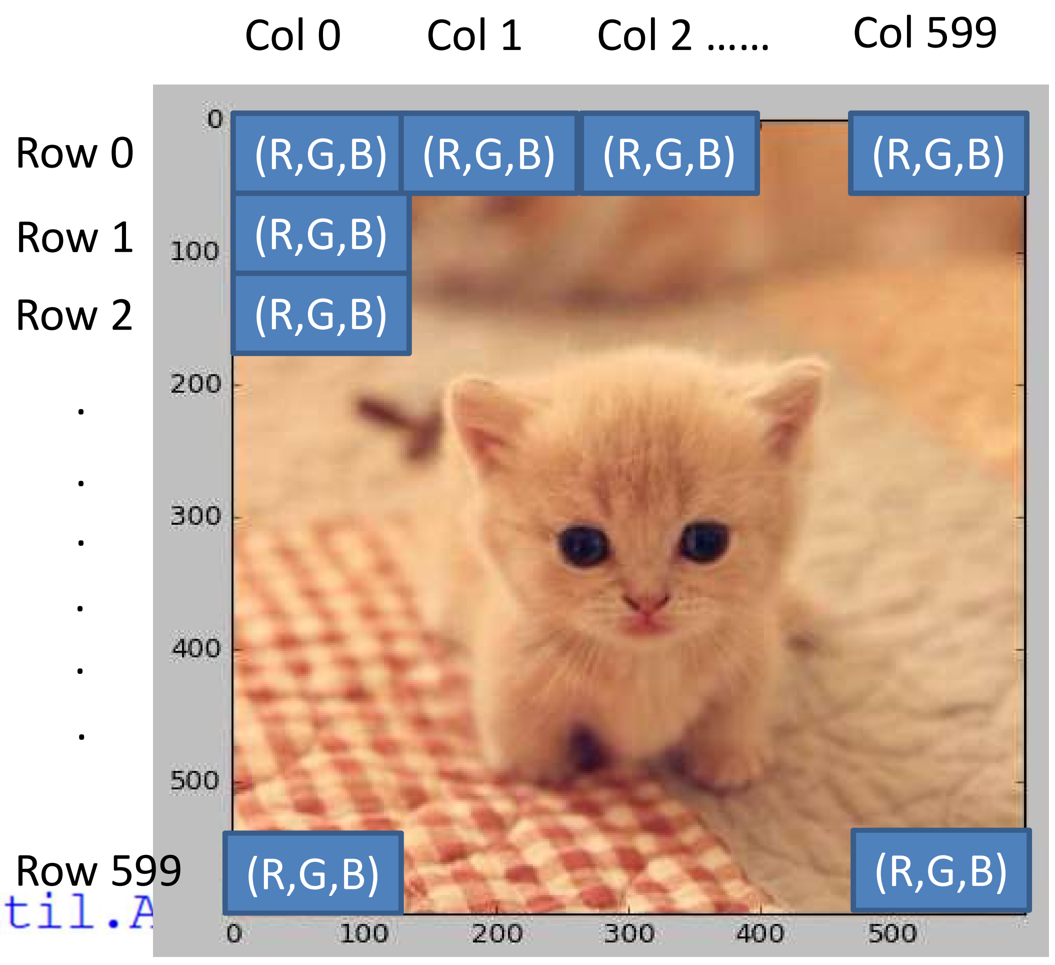


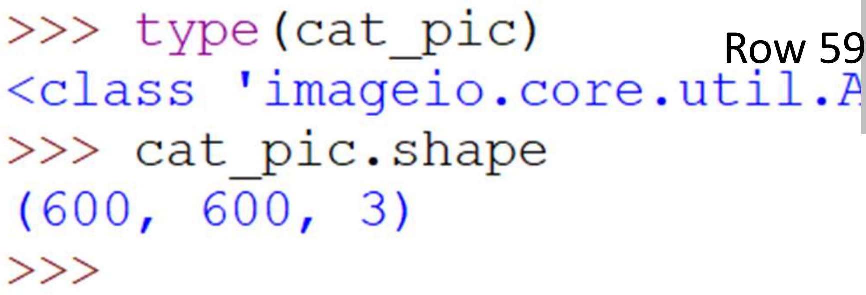


```
>>> type(cat_pic)
<class 'imageio.core.util.A
>>> cat_pic.shape
(600, 600, 3)
>>>
```

Image Processing

- 600 x 600 pixel
  - [R, G, B]
  - 0 <= R,G,B <=255



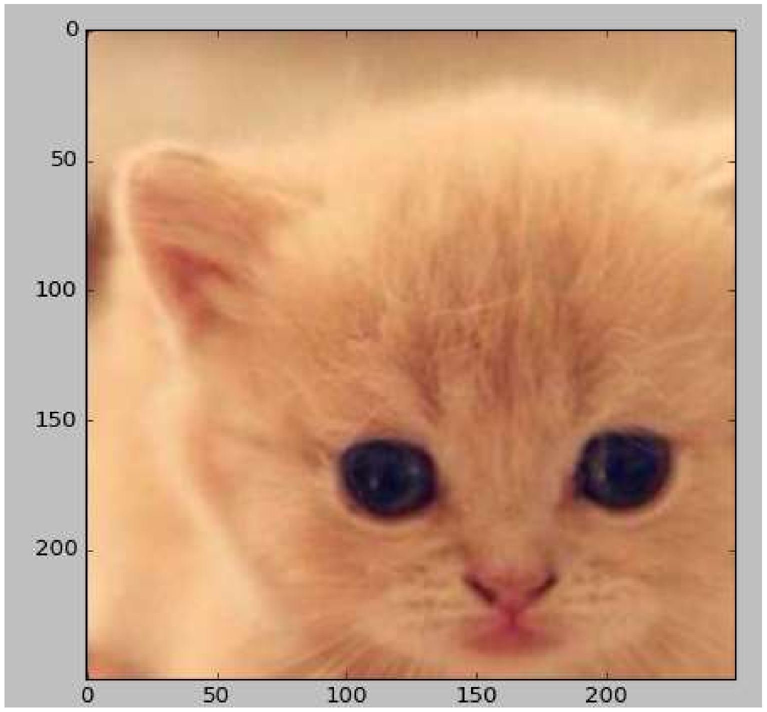


```
>>> type(cat_pic)
<class 'imageio.core.util.A
>>> cat_pic.shape
(600, 600, 3)
>>>
```

Image Processing

- Remember sub-matrix, string slicing, etc.?

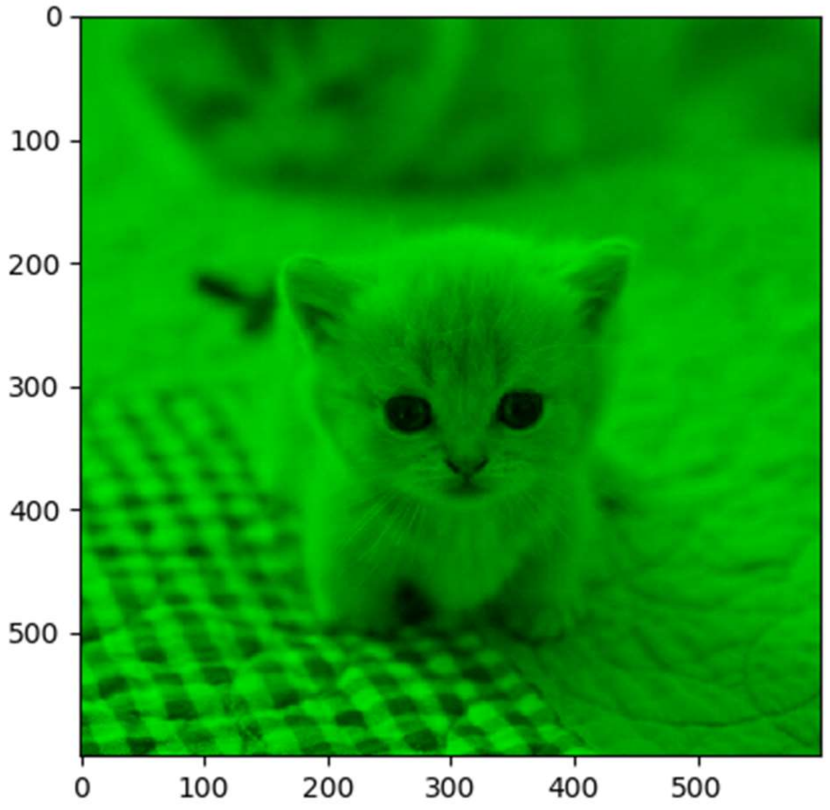
```
cat_pic2 = cat_pic[150:400,150:400]
plt.imshow(cat_pic2)
plt.show()
```



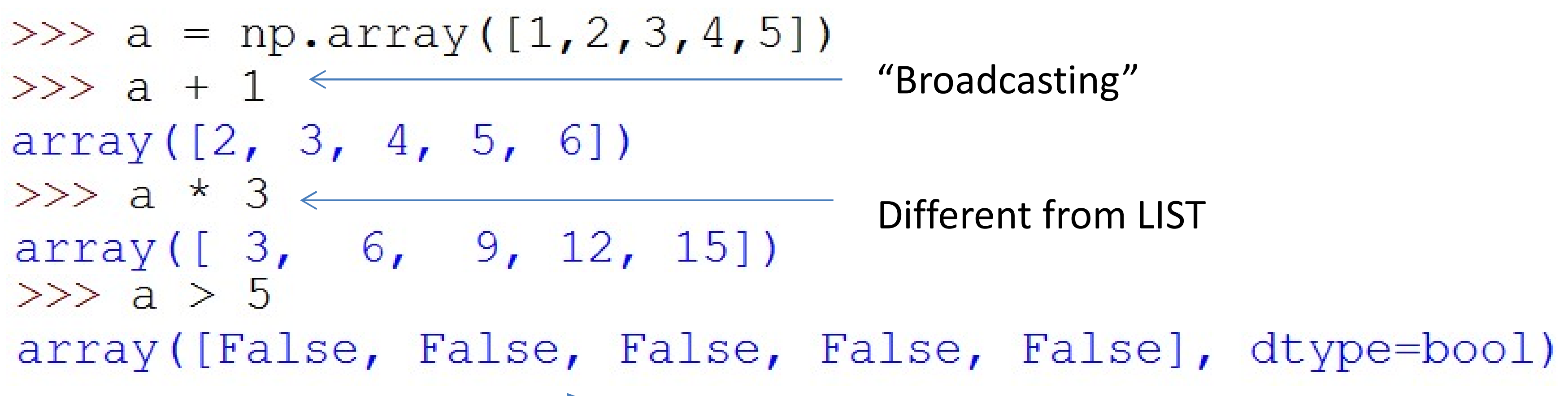
Broadcasting

```
cat_pic2 = cat_pic * [0, 1, 0]
plt.imshow(cat_pic2)
plt.show()
```

- every pixel multiply by
  - [R,G,B] x [0,1,0] =
  - [R x 0, G x 1, B x 0]
  - [0, G, 0]



Array Broadcasting



```
>>> a = np.array([1,2,3,4,5])
>>> a + 1
array([2, 3, 4, 5, 6])
>>> a * 3
array([ 3,  6,  9, 12, 15])
>>> a > 5
array([False, False, False, False, False], dtype=bool)
```

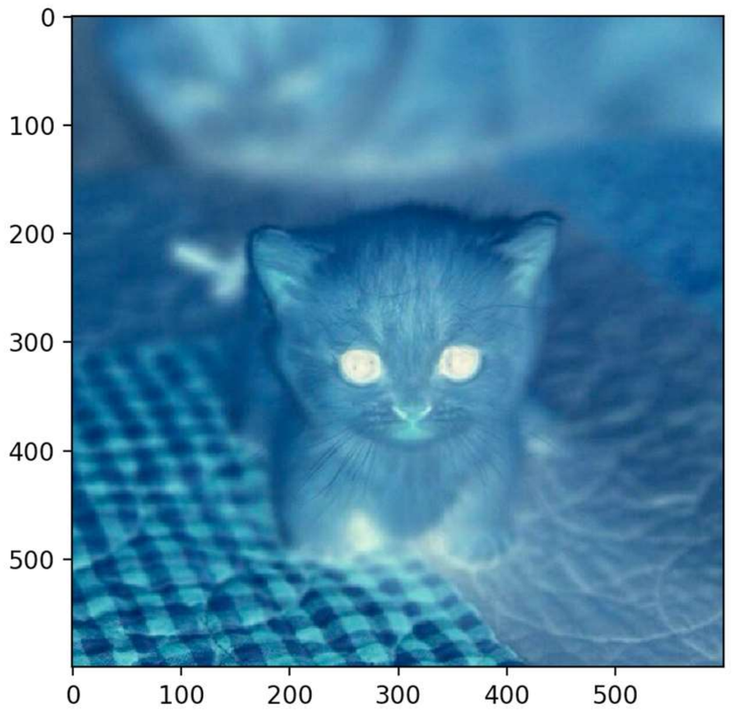
“Broadcasting”

Different from LIST

Create another array with the Boolean results

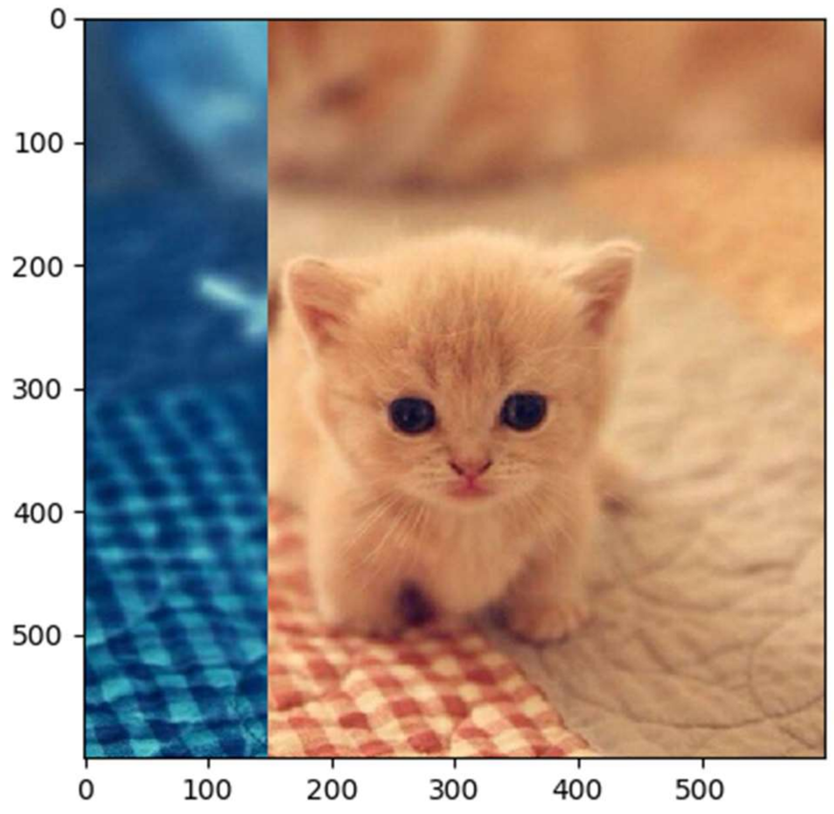
Negative Image

```
cat_pic2 = 255 - cat_pic
plt.imshow(cat_pic2)
plt.show()
```

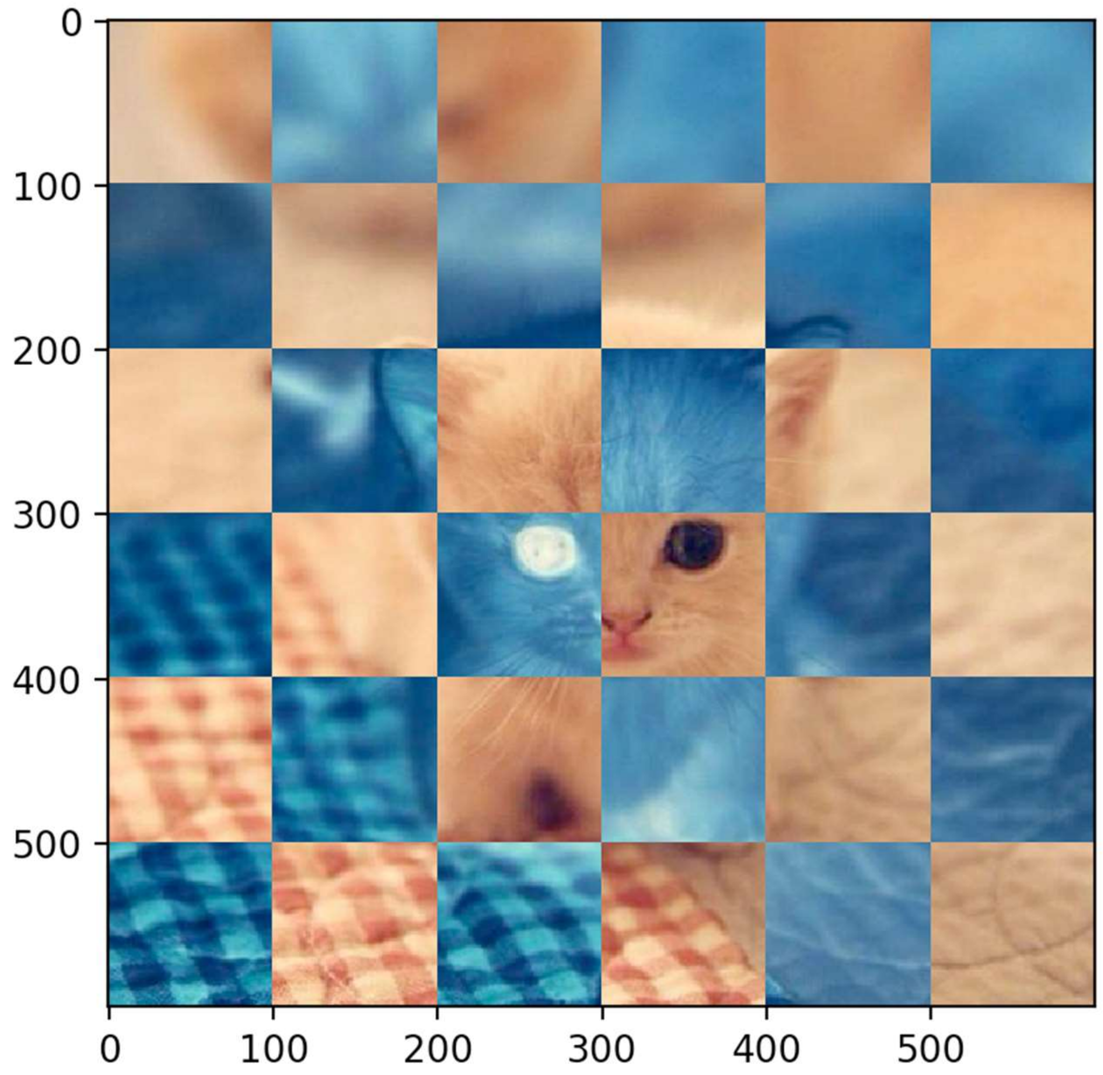


2D Array looping

```
for i in range(cat_pic.shape[0]):
    for j in range(cat_pic.shape[1]):
        if j < cat_pic.shape[1]/4:
            cat_pic[i][j] = 255 - cat_pic[i][j]

plt.imshow(cat_pic)
plt.show()
```

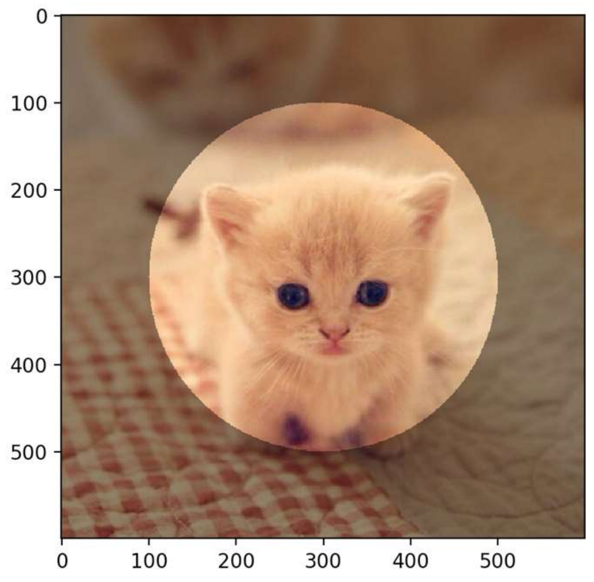


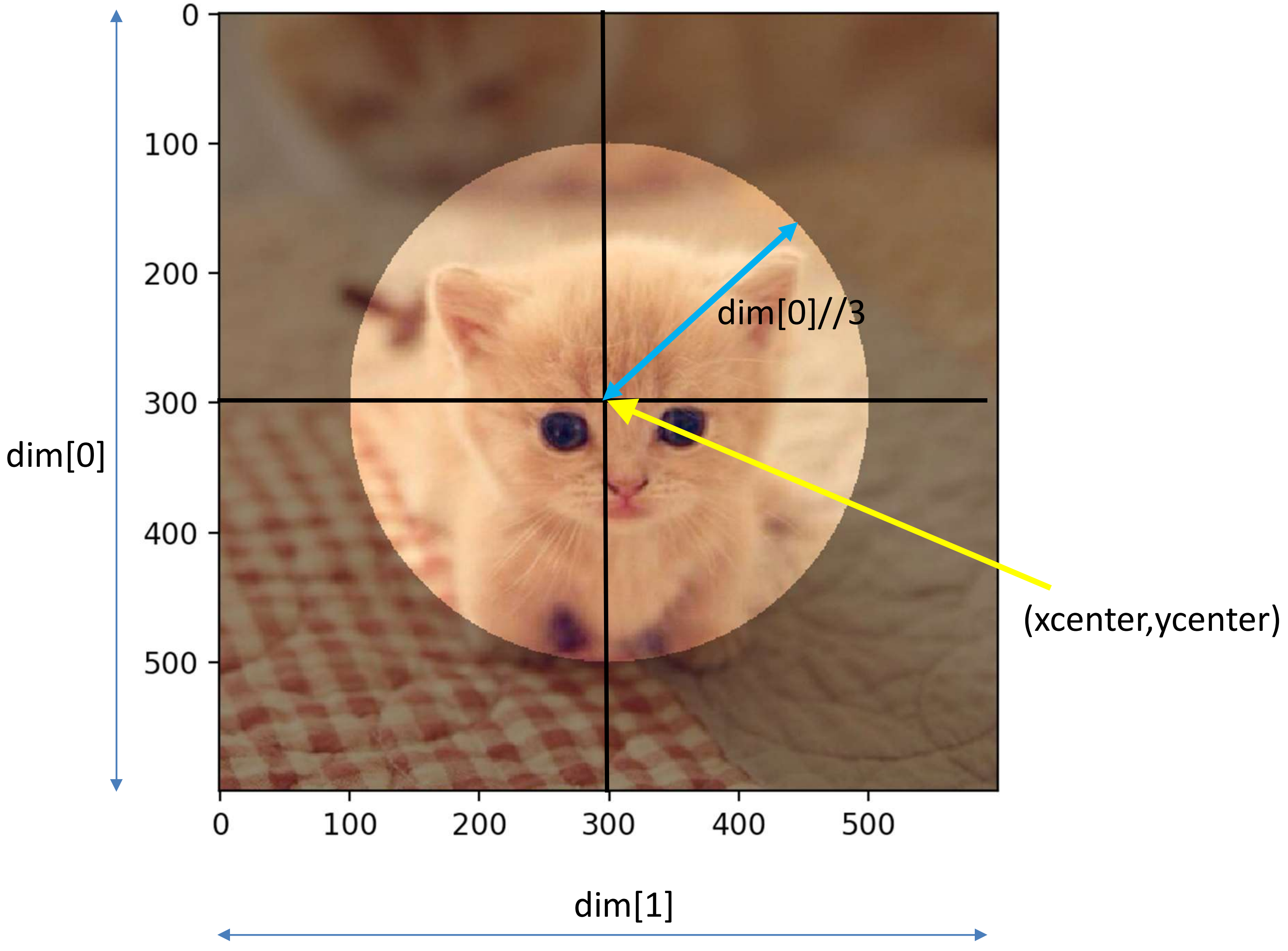
How to....?



Making a Mask

```
dim = cat_pic.shape
xcenter = dim[1]//2
ycenter = dim[0]//2

for i in range(cat_pic.shape[0]):
    for j in range(cat_pic.shape[1]):
        if (i-xcenter)**2 + (j-ycenter)**2 > (dim[0]//3)**2:
            cat_pic[i][j] = cat_pic[i][j]*0.3
```



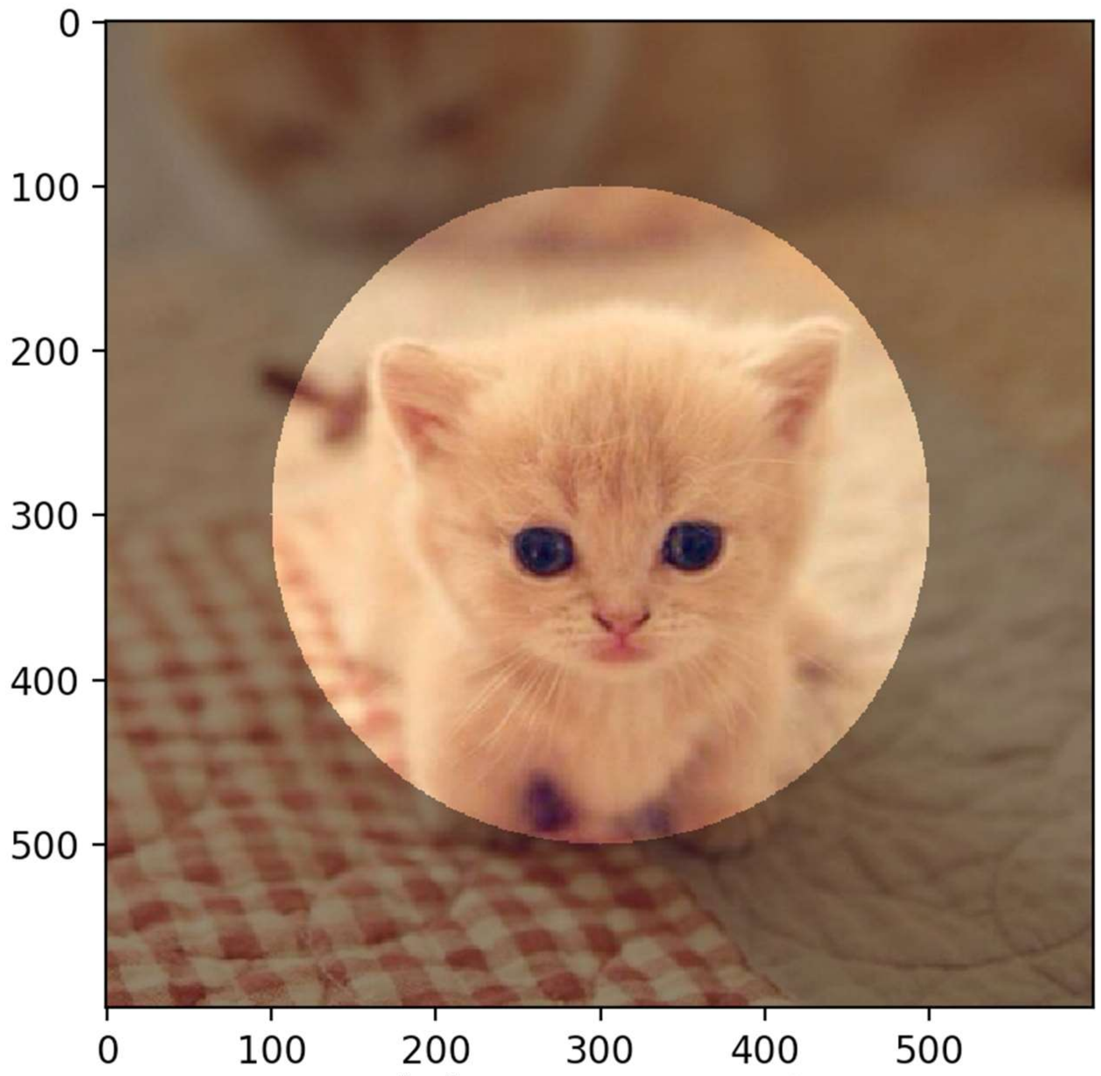

0
100
200
300
400
500
0
100
200
300
400
500
dim[0]
dim[1]
dim[0]//3
(xcenter,ycenter)

Making a Mask

If the pixel is out of the circle

```
dim = cat_pic.shape
xcenter = dim[1]//2
ycenter = dim[0]//2

for i in range(cat_pic.shape[0]):
    for j in range(cat_pic.shape[1]):
        if (i-xcenter)**2 + (j-ycenter)**2 > (dim[0]//3)**2:
            cat_pic[i][j] = cat_pic[i][j]*0.3
```

Each color of the pixel is divided by 2



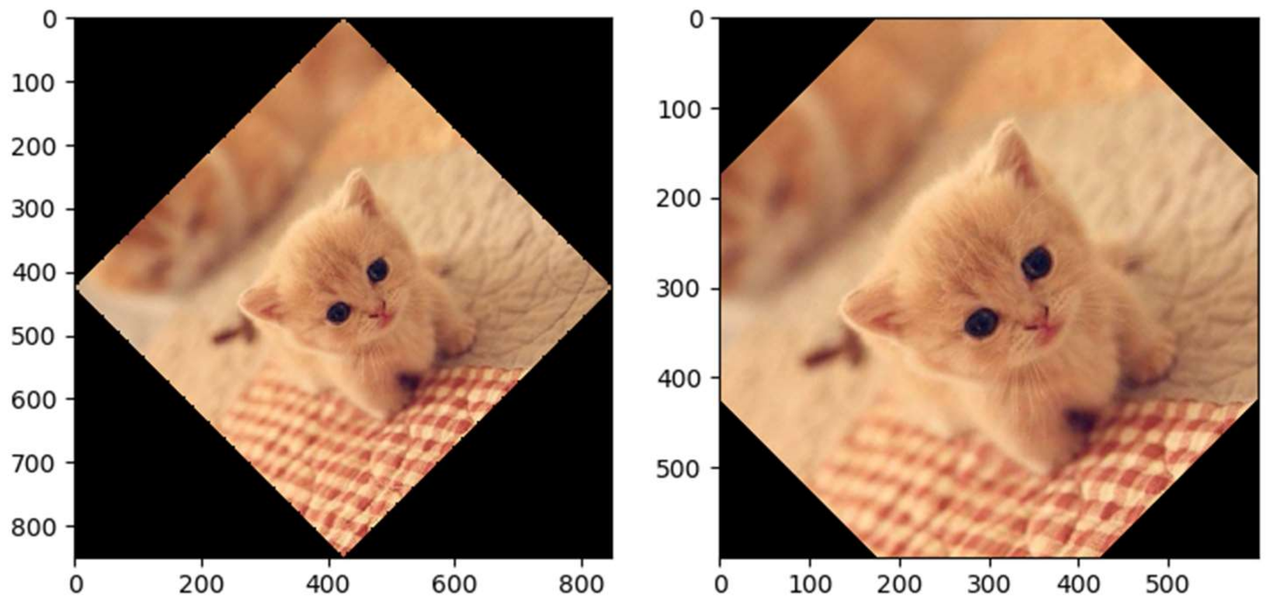
Your picture array

- Any time you want to save an image:

```
imageio.imsave('file name.png', cat_pic)
```

Rotating an Image

```
from scipy import ndimage
rcat1 = ndimage.rotate(cat_pic,45)
rcat2 = ndimage.rotate(cat_pic,45,reshape=False)
plt.subplot(121)
plt.imshow(rcat1)
plt.subplot(122)
plt.imshow(rcat2)
plt.show()
```



Applying Filters

```
from scipy import misc,ndimage
import matplotlib.pyplot as plt
import numpy as np

cat_pic = misc.imread('cute cat.jpg')
blurred_cat_pic = ndimage.gaussian_filter(
    cat_pic, sigma=(9,9,1))

plt.imshow(blurred_cat_pic)
plt.show()
```

Blending on x

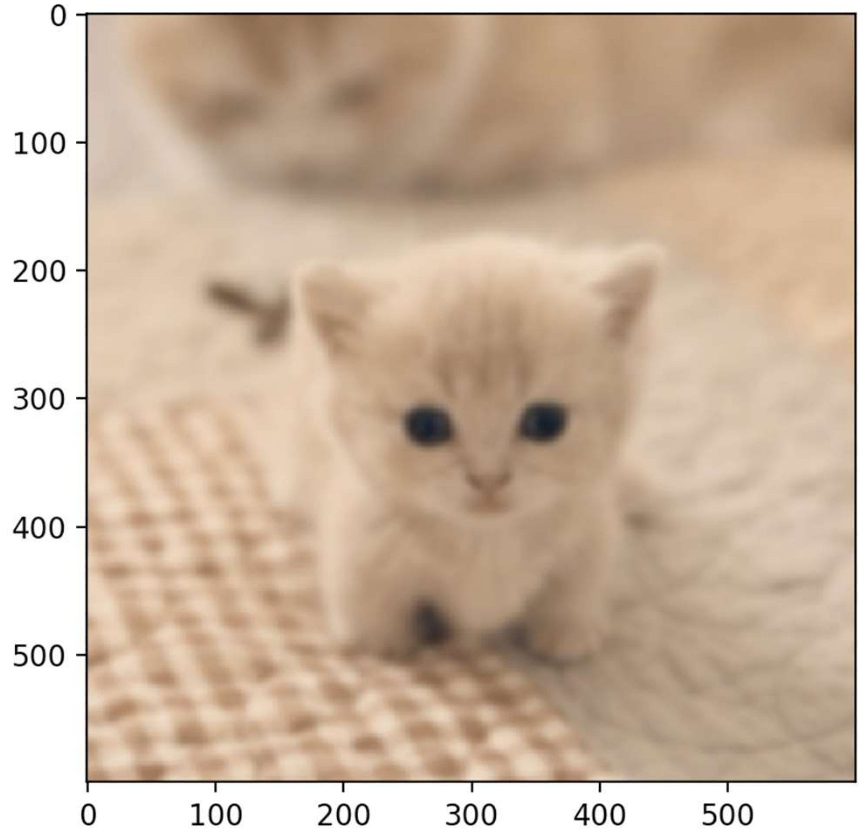
NO Blending on colors

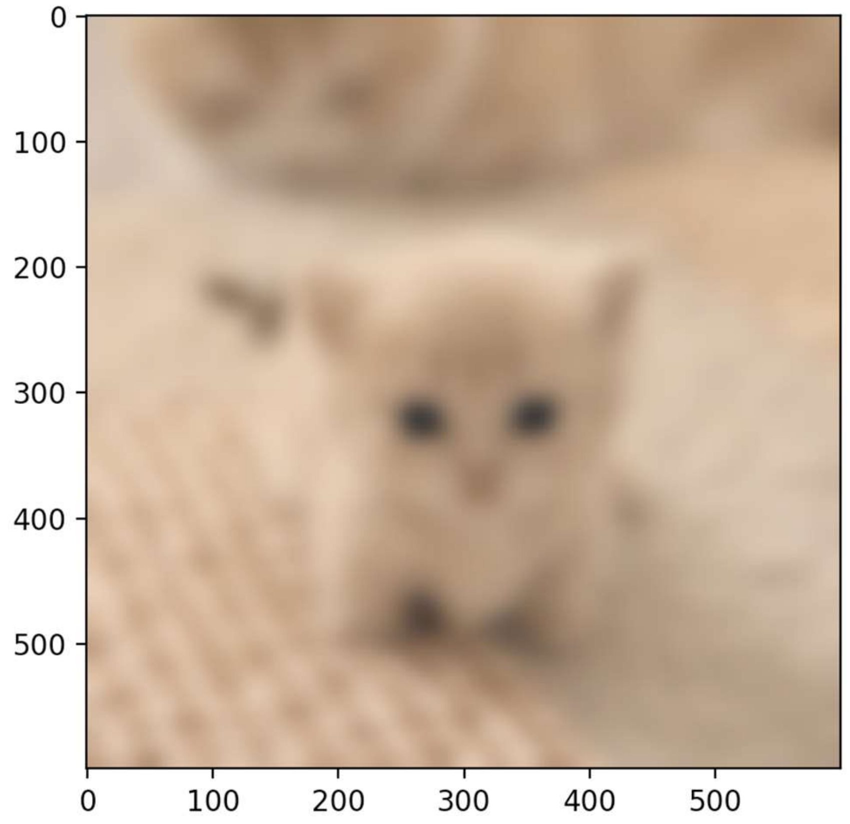
Blending on y

Applying Filters

```
blurred_cat = ndimage.gaussian_filter(cat_pic,sigma=(9,9,1))
```

- sigma = (3,3,1)
- sigma = (9,9,1)

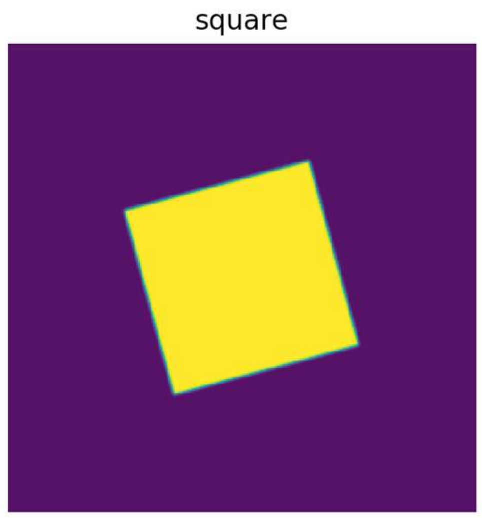




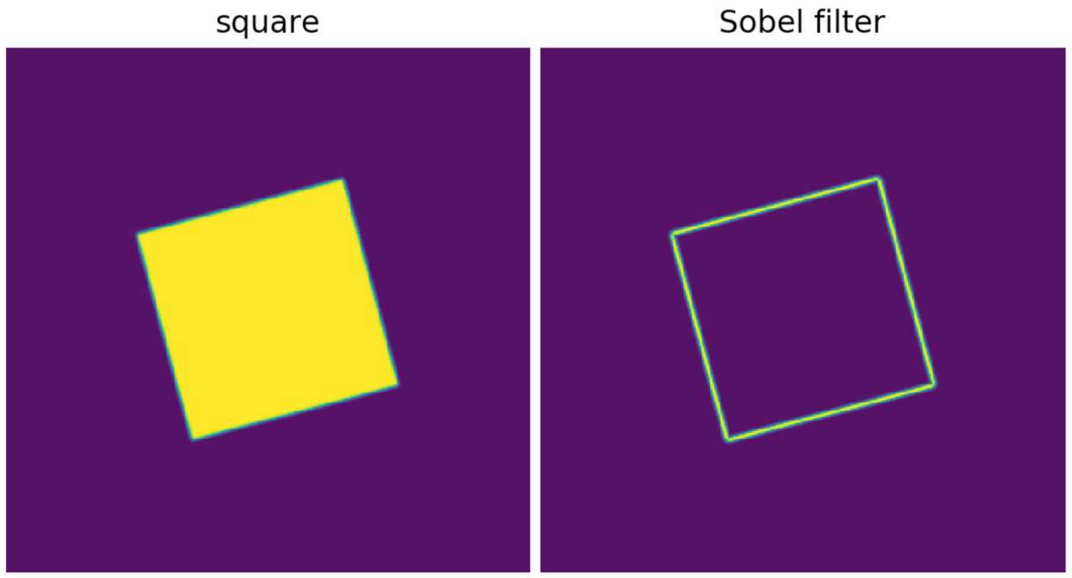
Edge Detection

- Anyhow generate an image

```
img = np.zeros((256, 256))
img[64:-64, 64:-64] = 1

img = ndimage.rotate(img, 15, mode='constant')
img = ndimage.gaussian_filter(img, 1)
```

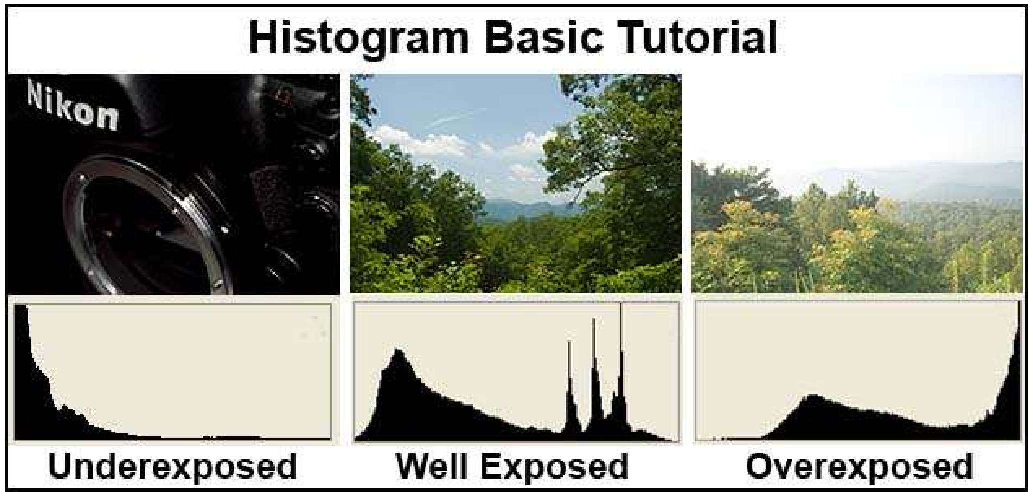


```python
# Applying Sobel filter to the image
sx = ndimage.sobel(img, axis=0, mode='constant')
sy = ndimage.sobel(img, axis=1, mode='constant')
sob = np.hypot(sx, sy)

plt.subplot(121)
plt.imshow(img)
plt.axis('off')
plt.title('square')

plt.subplot(122)
plt.imshow(sob)
plt.axis('off')
plt.title('Sobel filter')
```



More in Numpy and Scipy

- Fourier Transform
- Uniform filter
- Histogram
- Laplace… etc



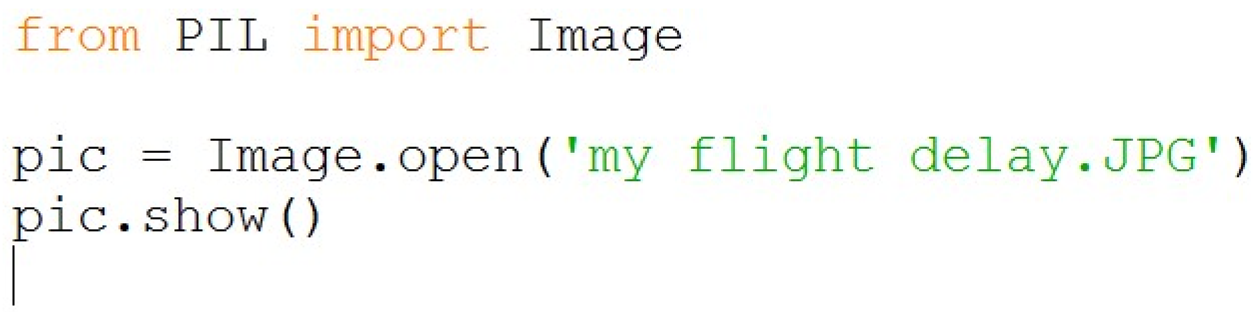
- More on
  - https://docs.scipy.org/doc/scipy/reference/ndimage.html

PILLOW

A Fork in PIL

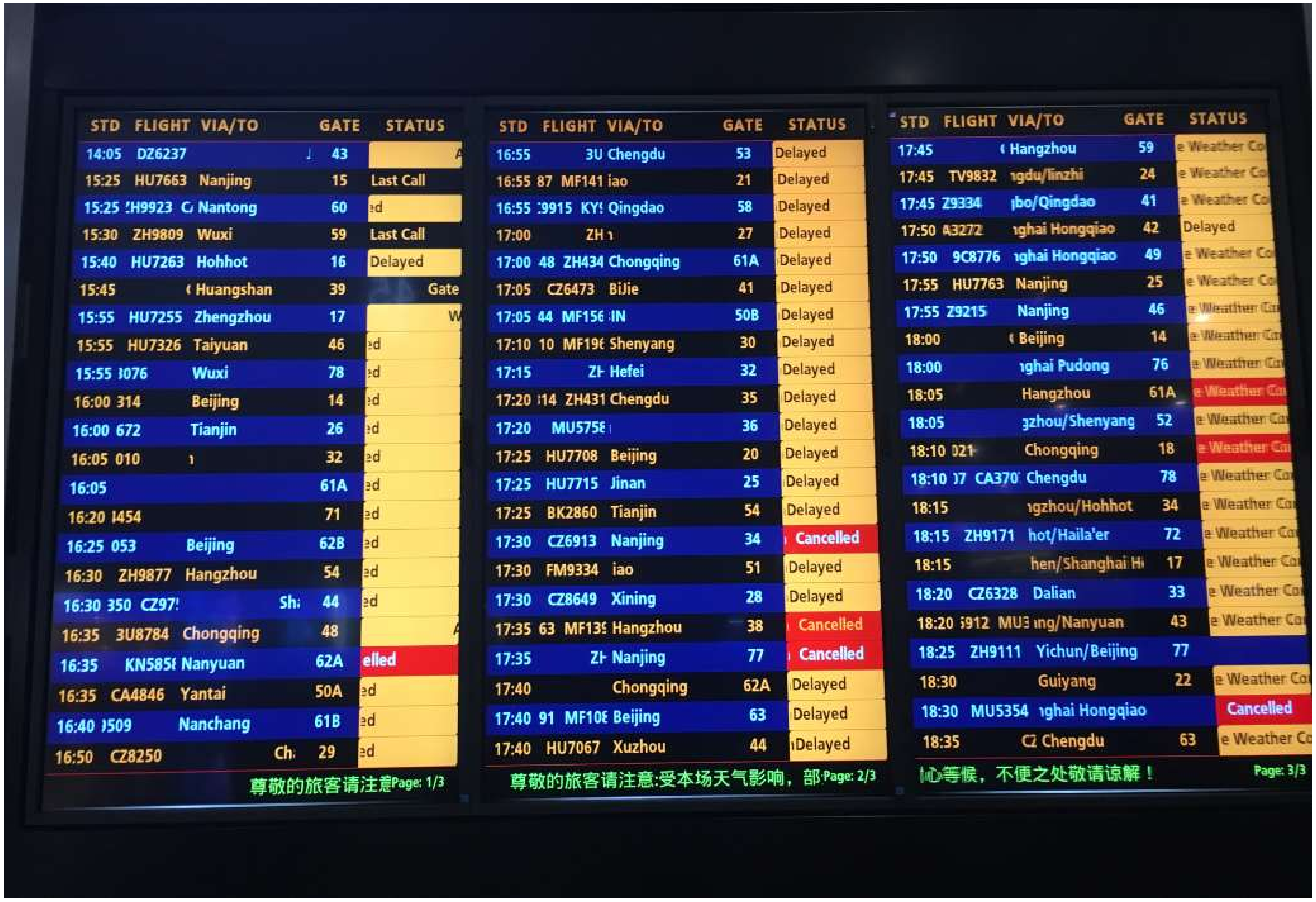
PILLOW is a fork of PIL

- PIL stands for Python Imaging Library



```python
from PIL import Image

pic = Image.open('my flight delay.JPG')
pic.show()
```

Let’s get the secret out



STD	FLIGHT	VIA/TO	GATE	STATUS
14:05	DZ6237		43	
15:25	HU7663	Nanjing	15	Last Call
15:25	ZH9923	C Nantong	60	ed
15:30	ZH9809	Wuxi	59	Last Call
15:40	HU7263	Hohhot	16	Delayed
15:45		Huangshan	39	Gate
15:55	HU7255	Zhengzhou	17	W
15:55	HU7326	Taiyuan	46	ed
15:55	3076	Wuxi	78	ed
16:00	314	Beijing	14	ed
16:00	672	Tianjin	26	ed
16:05	010		32	ed
16:05			61A	ed
16:20	1454		71	ed
16:25	053	Beijing	62B	ed
16:30	ZH9877	Hangzhou	54	ed
16:30	350 CZ97	Sh	44	ed
16:35	3U8784	Chongqing	48	
16:35	KN585	Nanyuan	62A	elled
16:35	CA4846	Yantai	50A	ed
16:40	3509	Nanchang	61B	ed
16:50	CZ8250	Ch	29	ed

尊敬的旅客请注意 Page: 1/3

STD	FLIGHT	VIA/TO	GATE	STATUS
16:55	3U	Chengdu	53	Delayed
16:55	87 MF141	iao	21	Delayed
16:55	9915 KY	Qingdao	58	Delayed
17:00	ZH		27	Delayed
17:00	48 ZH434	Chongqing	61A	Delayed
17:05	CZ6473	BiJie	41	Delayed
17:05	44 MF156	IN	50B	Delayed
17:10	10 MF19	Shenyang	30	Delayed
17:15	ZH	Hefei	32	Delayed
17:20	14 ZH431	Chengdu	35	Delayed
17:20	MU575		36	Delayed
17:25	HU7708	Beijing	20	Delayed
17:25	HU7715	Jinan	25	Delayed
17:25	BK2860	Tianjin	54	Delayed
17:30	CZ6913	Nanjing	34	Cancelled
17:30	FM9334	iao	51	Delayed
17:30	CZ8649	Xining	28	Delayed
17:35	63 MF13	Hangzhou	38	Cancelled
17:35	ZH	Nanjing	77	Cancelled
17:40		Chongqing	62A	Delayed
17:40	91 MF10	Beijing	63	Delayed
17:40	HU7067	Xuzhou	44	Delayed

尊敬的旅客请注意:受本场天气影响，部 Page: 2/3

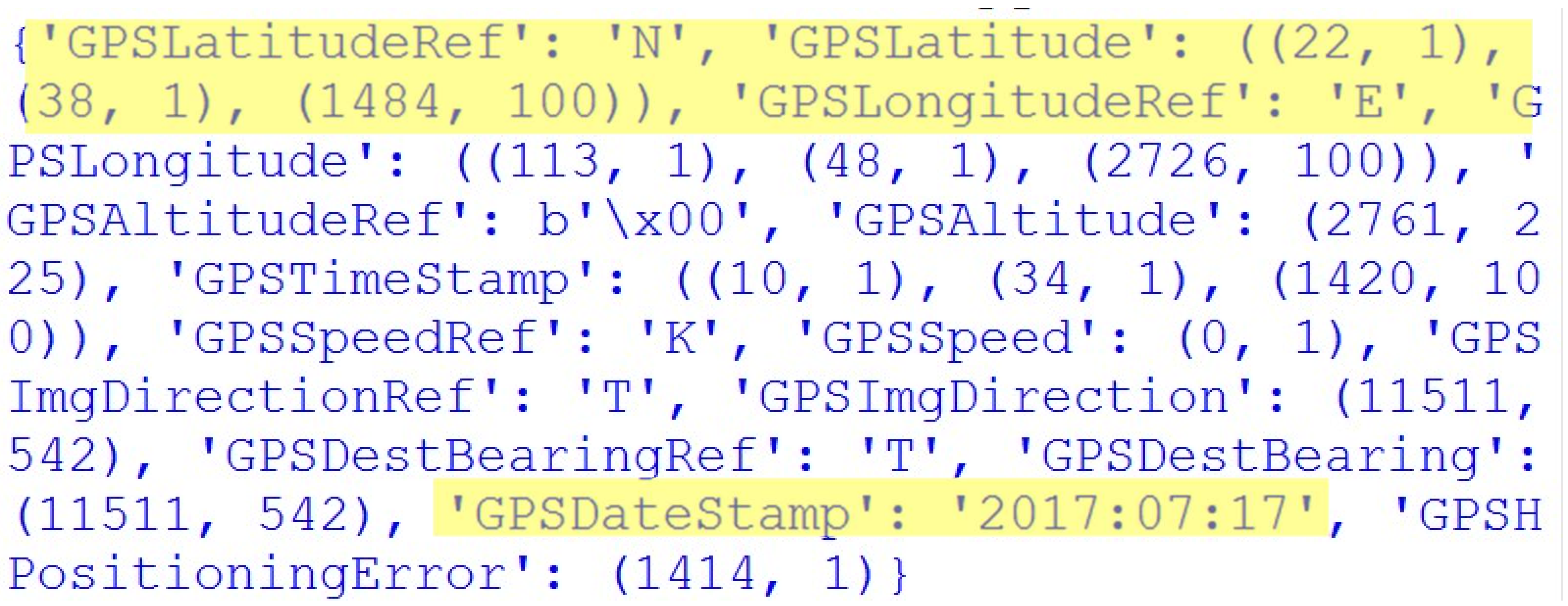
STD	FLIGHT	VIA/TO	GATE	STATUS
17:45		Hangzhou	59	e Weather Co
17:45	TV9832	gdu/linzhi	24	e Weather Co
17:45	Z9334	bo/Qingdao	41	e Weather Co
17:50	A3272	ghai Hongqiao	42	Delayed
17:50	9C8776	ghai Hongqiao	49	e Weather Co
17:55	HU7763	Nanjing	25	e Weather Co
17:55	Z9215	Nanjing	46	e Weather Co
18:00		Beijing	14	e Weather Co
18:00		ghai Pudong	76	e Weather Co
18:05		Hangzhou	61A	e Weather Co
18:05		gzhou/Shenyang	52	e Weather Co
18:10	021	Chongqing	18	e Weather Co
18:10	37 CA370	Chengdu	78	e Weather Co
18:15		gzhou/Hohhot	34	e Weather Co
18:15	ZH9171	hot/Haila'er	72	e Weather Co
18:15		hen/Shanghai H	17	e Weather Co
18:20	CZ6328	Dalian	33	e Weather Co
18:20	5912 MU3	ng/Nanyuan	43	e Weather Co
18:25	ZH9111	Yichun/Beijing	77	
18:30		Guiyang	22	e Weather Co
18:30	MU5354	ghai Hongqiao		Cancelled
18:35	CZ	Chengdu	63	e Weather Co

心等候，不便之处敬请谅解！ Page: 3/3

```python
from PIL import Image
from PIL.ExifTags import TAGS, GPSTAGS

pic = Image.open('my flight delay.JPG')

def get_exif_data(image):
    exif_data = {}
    info = image._getexif()
    if info:
        for tag, value in info.items():
            decoded = TAGS.get(tag, tag)
            if decoded == "GPSInfo":
                gps_data = {}
                for t in value:
                    sub_decoded = GPSTAGS.get(t, t)
                    gps_data[sub_decoded] = value[t]
                exif_data[decoded] = gps_data
            else:
                exif_data[decoded] = value

    return exif_data

print(get_exif_data(pic)['GPSInfo'])
pic.show()
```

PILLOW

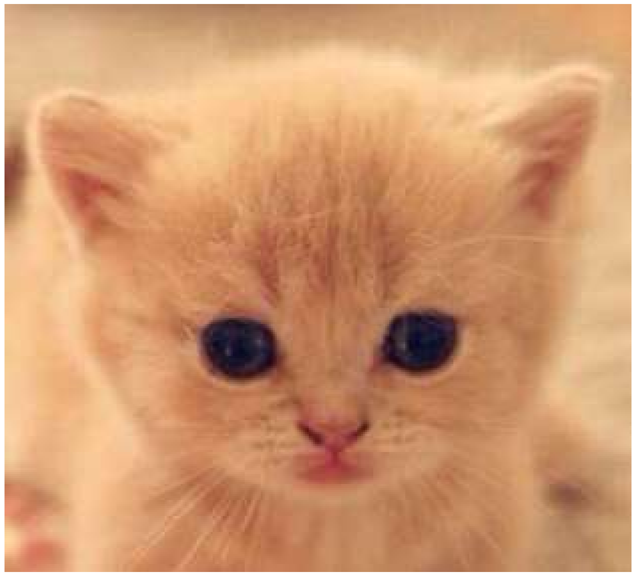
- Cannot escape!

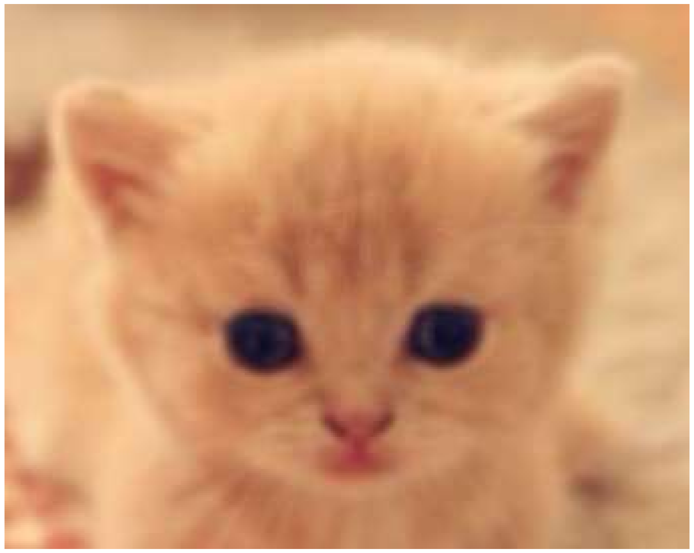


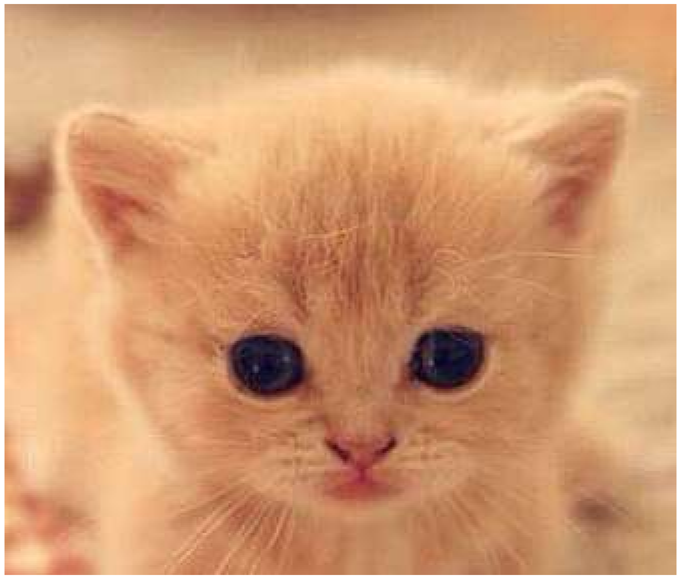
```
{'GPSLatitudeRef': 'N', 'GPSLatitude': ((22, 1),
(38, 1), (1484, 100)), 'GPSLongitudeRef': 'E', 'G
PSLongitude': ((113, 1), (48, 1), (2726, 100)), '
GPSAltitudeRef': b'\x00', 'GPSAltitude': (2761, 2
25), 'GPSTimeStamp': ((10, 1), (34, 1), (1420, 10
0)), 'GPSSpeedRef': 'K', 'GPSSpeed': (0, 1), 'GPS
ImgDirectionRef': 'T', 'GPSImgDirection': (11511,
542), 'GPSDestBearingRef': 'T', 'GPSDestBearing':
(11511, 542), 'GPSDateStamp': '2017:07:17', 'GPSH
PositioningError': (1414, 1)}
```

PILLOW

```
from PIL import Image
from PIL import ImageFilter
pic = Image.open('cute cat.jpg')

pic.show()

blurred_pic =  pic.filter(ImageFilter.BLUR)
blurred_pic.show()

sharpen_pic =  pic.filter(ImageFilter.SHARPEN)
sharpen_pic.show()
```


Original


Blurred

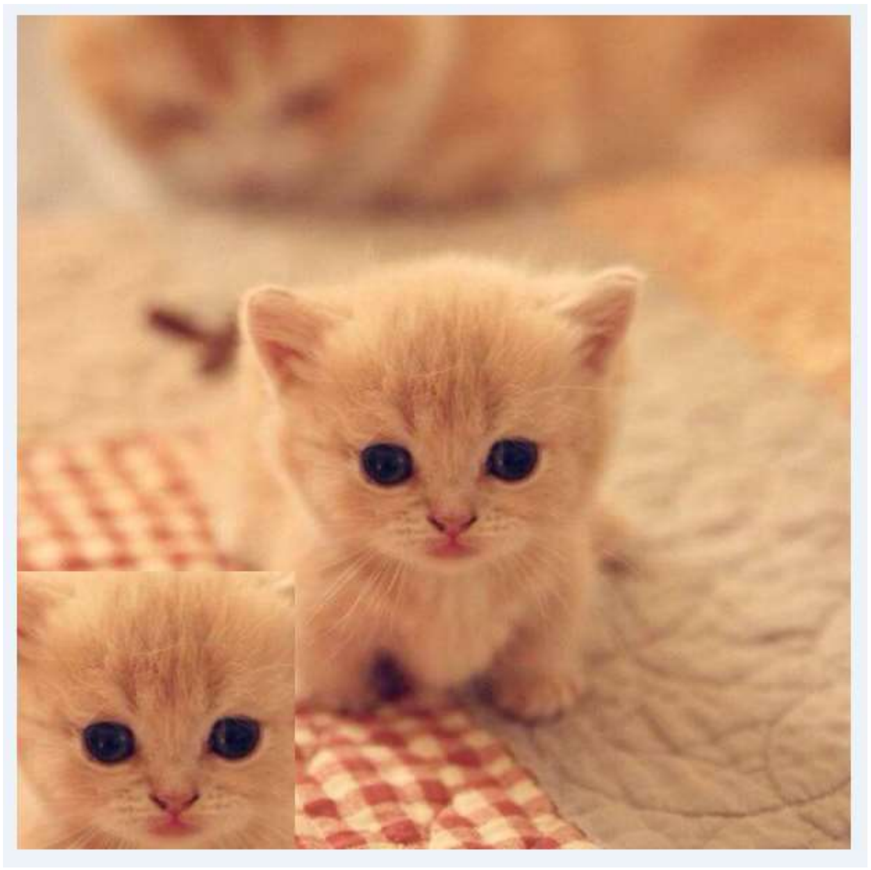

Sharpen

Copy And Paste

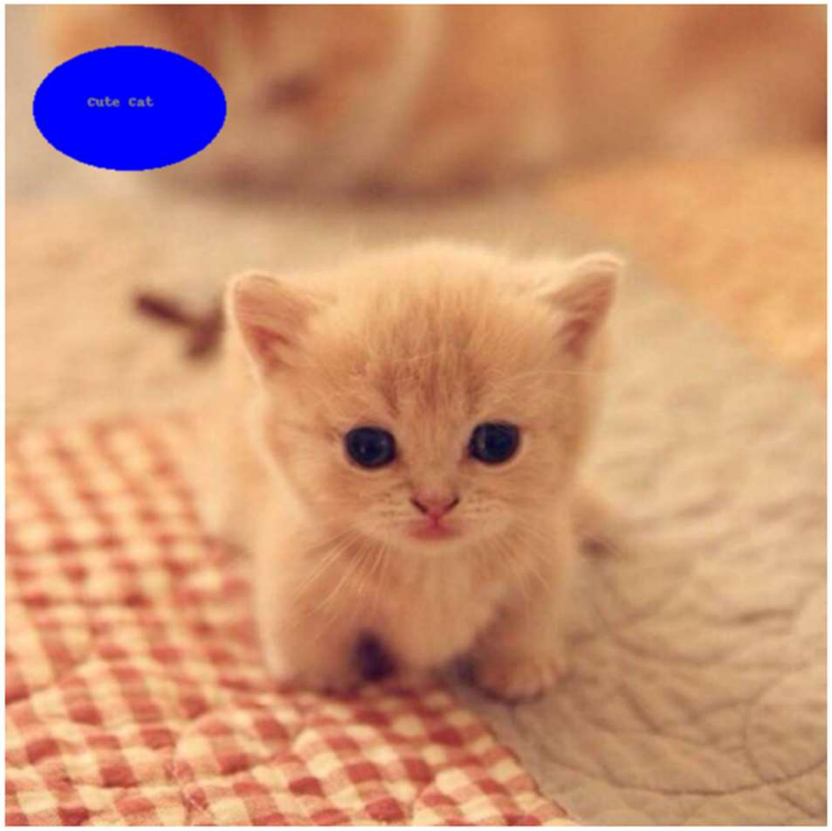
```
from PIL import Image

pic = Image.open('cute cat.JPG')
part = pic.crop((200,200,400,400))

pic.paste(part,(0,400))
pic.show()
```

Copy (crop) the part of the picture

Paste it on the position (0,400)



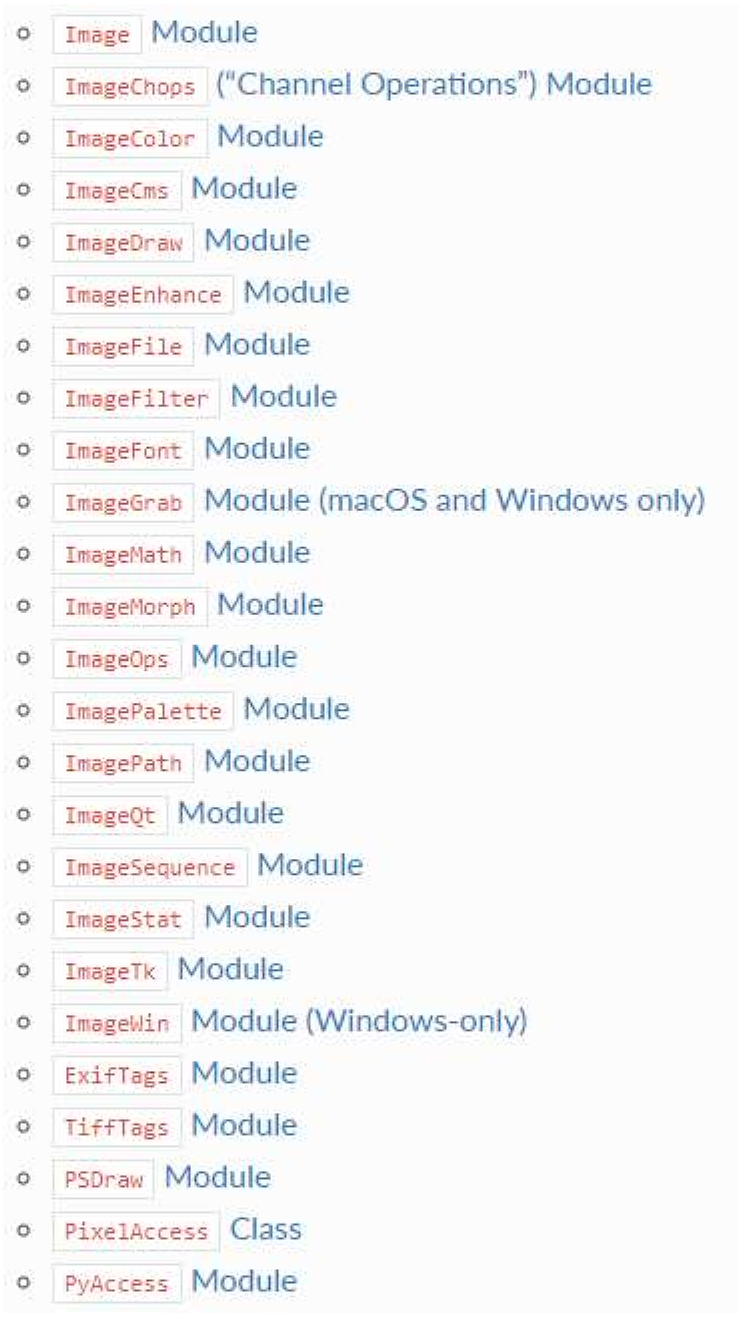
```python
from PIL import Image, ImageDraw, ImageFont

pic = Image.open('cute cat.JPG')

draw = ImageDraw.Draw(pic)
draw.ellipse((20, 30, 160, 120), fill='blue')
draw.text((60,65),'Cute Cat', fill = 'gray')

pic.show()
```



Other operations

- resize
- rotation/flipping
- traspose
- Drawing shapes
- etc. etc..

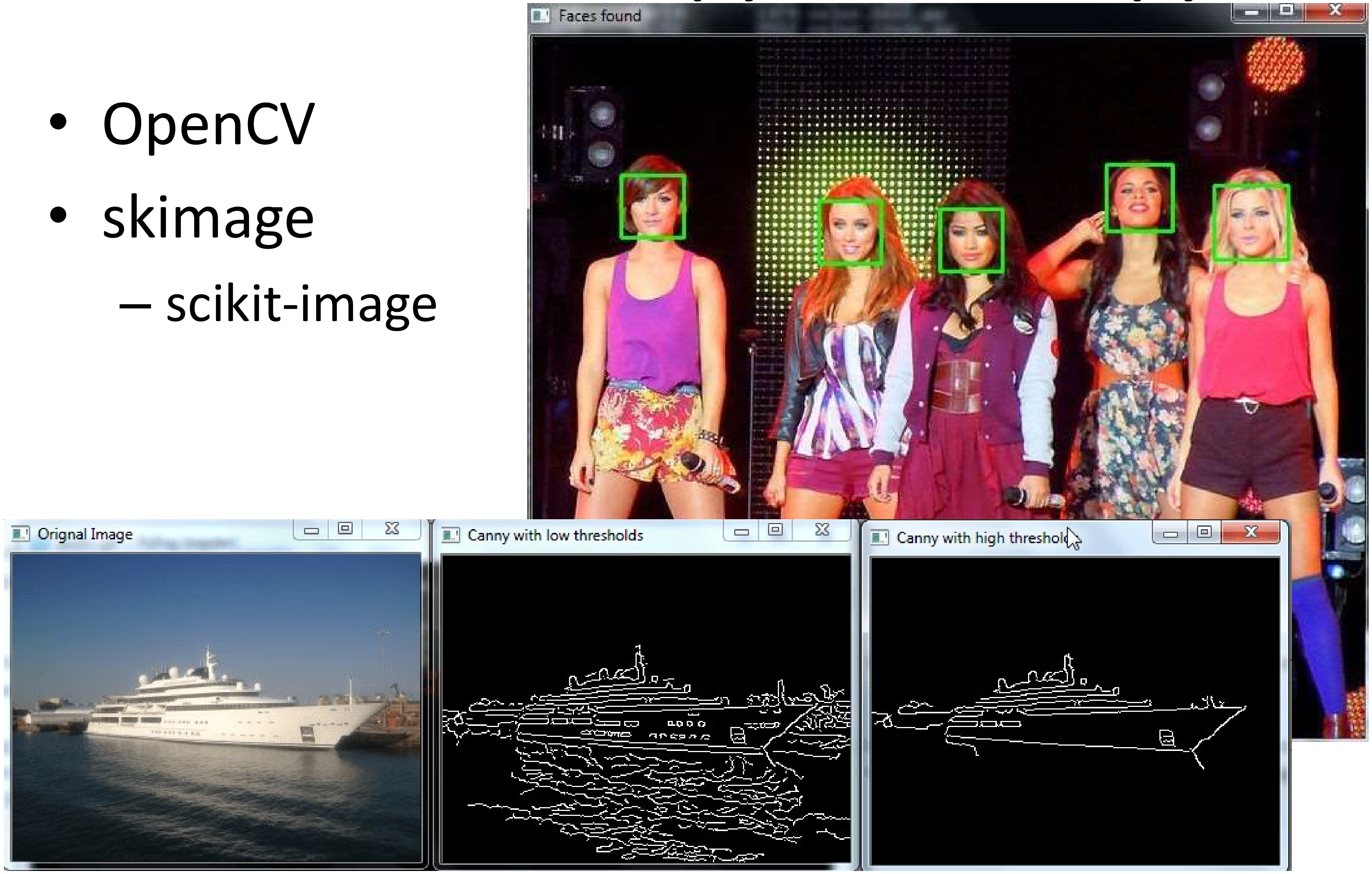
https://pillow.readthedocs.io/en/stable/



- `Image` Module
- `ImageChops` ("Channel Operations") Module
- `ImageColor` Module
- `ImageCms` Module
- `ImageDraw` Module
- `ImageEnhance` Module
- `ImageFile` Module
- `ImageFilter` Module
- `ImageFont` Module
- `ImageGrab` Module (macOS and Windows only)
- `ImageMath` Module
- `ImageMorph` Module
- `ImageOps` Module
- `ImagePalette` Module
- `ImagePath` Module
- `ImageQt` Module
- `ImageSequence` Module
- `ImageStat` Module
- `ImageTk` Module
- `ImageWin` Module (Windows-only)
- `ExifTags` Module
- `TiffTags` Module
- `PSDraw` Module
- `PixelAccess` Class
- `PyAccess` Module

Other Than Scipy and Numpy

- OpenCV
- skimage
  - scikit-image


Faces found
Orignal Image
Canny with low thresholds
Canny with high threshol