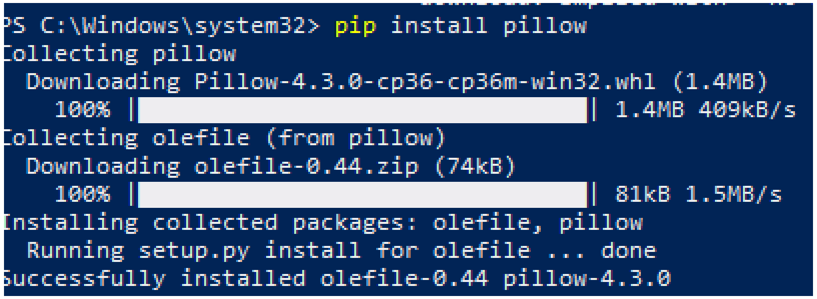
MANIPULATING IMAGES

Clifford Addai Twum
Riadh Ben Hassine

PILLOW MODULE

- Install via pip
  - pip install pillow



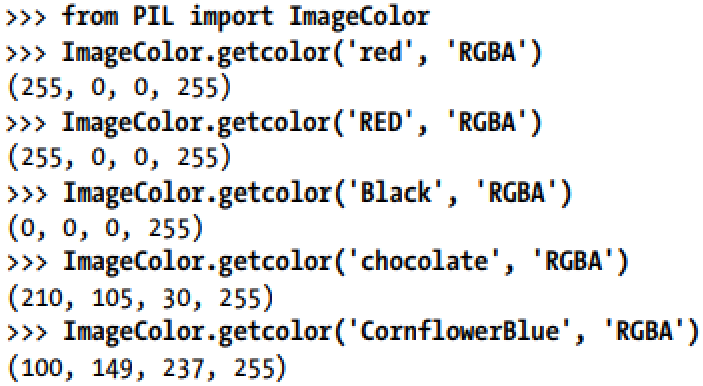
```
PS C:\Windows\system32> pip install pillow
Collecting pillow
  Downloading Pillow-4.3.0-cp36-cp36m-win32.whl (1.4MB)
    100% |████████████████████████████████| 1.4MB 409kB/s
Collecting olefile (from pillow)
  Downloading olefile-0.44.zip (74kB)
    100% |████████████████████████████████| 81kB 1.5MB/s
Installing collected packages: olefile, pillow
  Running setup.py install for olefile ... done
Successfully installed olefile-0.44 pillow-4.3.0
```

- Import with PIL
  - from PIL import *
  - from PIL import ImageColor
  - Backward compatible with Python Imaging Library

RGBA

- RGBA is a color space defined by:
  - **R**ed **G**reen **B**lue **A**lpha

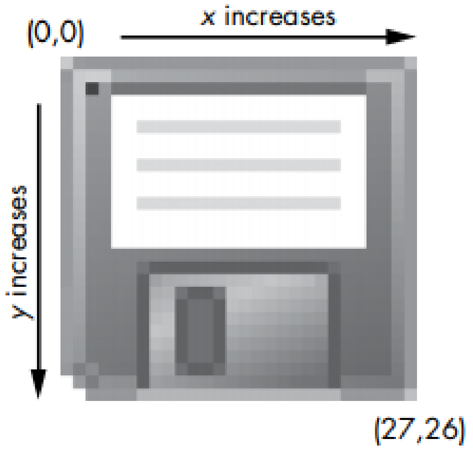
- ImageColor.getcolor()
  - Returns the value of a color
  - Tuple



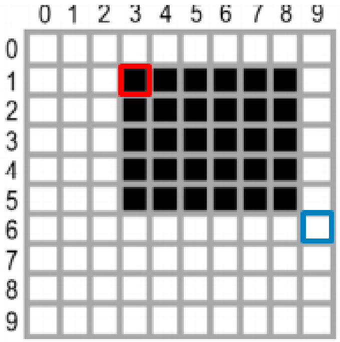
```
>>> from PIL import ImageColor
>>> ImageColor.getcolor('red', 'RGBA')
(255, 0, 0, 255)
>>> ImageColor.getcolor('RED', 'RGBA')
(255, 0, 0, 255)
>>> ImageColor.getcolor('Black', 'RGBA')
(0, 0, 0, 255)
>>> ImageColor.getcolor('chocolate', 'RGBA')
(210, 105, 30, 255)
>>> ImageColor.getcolor('CornflowerBlue', 'RGBA')
(100, 149, 237, 255)
```

COORDINATES AND BOX TUPLES

- Pixels are defined by x and y
  - (0, 0) is the origin at the top left



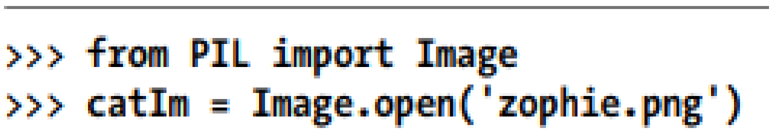
- Box tuples
  - Used to select an rectangle area
  - Coordinates left top followed by width and height
  - Ex. (3, 1, 9, 6)



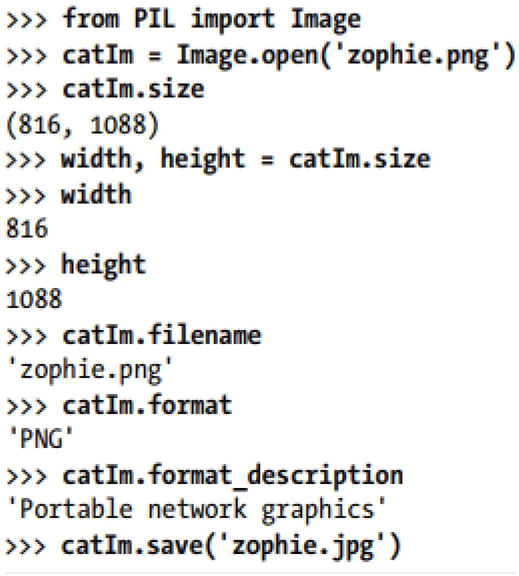
MANIPULATING IMAGES WITH PILLOW

- Import Image
- Open an image
  - Returns a image object

- Information about the image
  - Size, filename, format
- Save image in another format



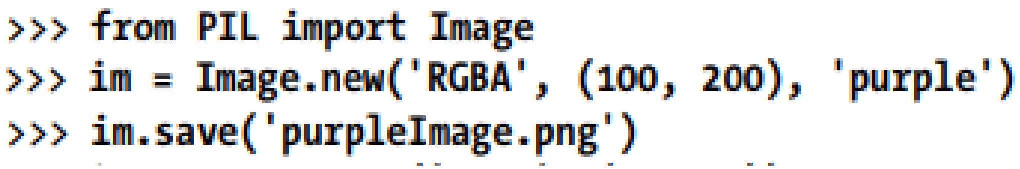
```
>>> from PIL import Image
>>> catIm = Image.open('zophie.png')
```



```
>>> from PIL import Image
>>> catIm = Image.open('zophie.png')
>>> catIm.size
(816, 1088)
>>> width, height = catIm.size
>>> width
816
>>> height
1088
>>> catIm.filename
'zophie.png'
>>> catIm.format
'PNG'
>>> catIm.format_description
'Portable network graphics'
>>> catIm.save('zophie.jpg')
```

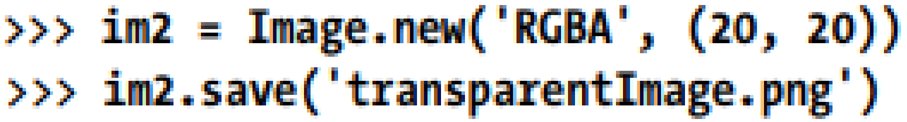
MANIPULATING IMAGES WITH PILLOW

- Create a new image
  - Color space, size and color
- Save the image



```
>>> from PIL import Image
>>> im = Image.new('RGBA', (100, 200), 'purple')
>>> im.save('purpleImage.png')
```

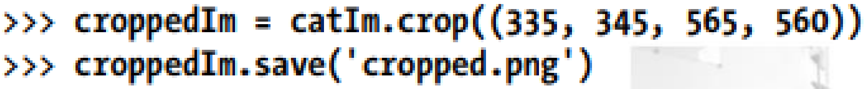
- Transparant image?
  - Don’t give a color value



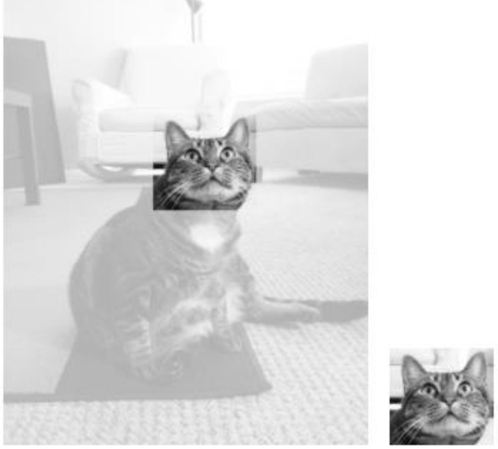
```
>>> im2 = Image.new('RGBA', (20, 20))
>>> im2.save('transparentImage.png')
```

CROPPING, COPYING AND PASTING

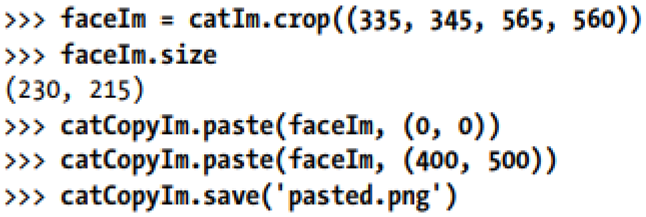
- Cropping an image
  - Using a box tuple
  - Save as a new image



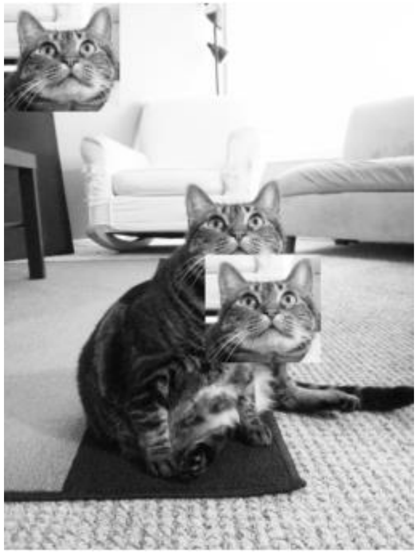
```
>>> croppedIm = catIm.crop((335, 345, 565, 560))
>>> croppedIm.save('cropped.png')
```



- Copy and paste
  - Save the crop in a variable
  - Paste by using coordinates

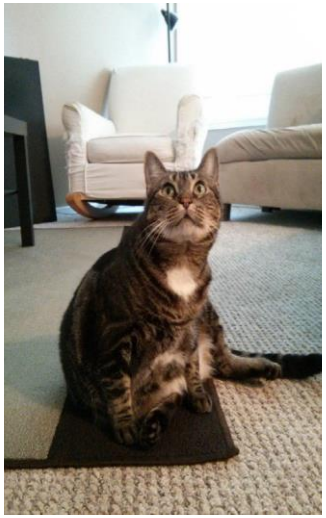


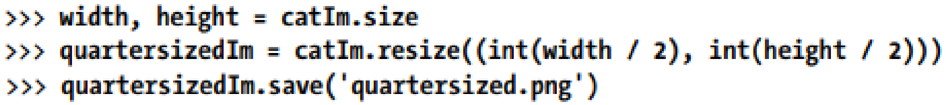
```
>>> faceIm = catIm.crop((335, 345, 565, 560))
>>> faceIm.size
(230, 215)
>>> catCopyIm.paste(faceIm, (0, 0))
>>> catCopyIm.paste(faceIm, (400, 500))
>>> catCopyIm.save('pasted.png')
```



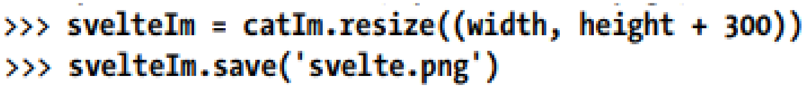
RESIZING

- Save the size in variables
  - Size is a tuple ➔ width, height
- Resize
  - Change the size with .resize
  - Aspect ratio ➔ Divide or multiply the width and height





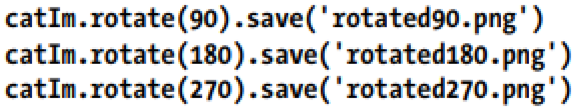
```
>>> width, height = catIm.size
>>> quartersizedIm = catIm.resize((int(width / 2), int(height / 2)))
>>> quartersizedIm.save('quartersized.png')
```



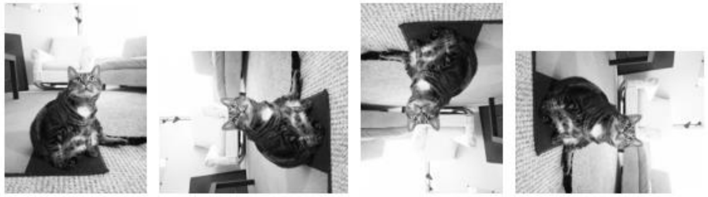
```
>>> svelteIm = catIm.resize((width, height + 300))
>>> svelteIm.save('svelte.png')
```

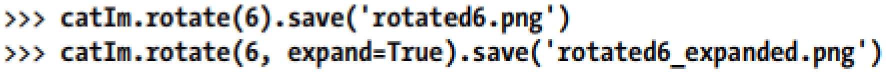
ROTATING

- Rotate the image
  - Save it with .save
  - Show it directly with .show
- Custom rotation
  - Rotating 6 degrees ➔ cuts off parts of the image
  - Expand=true ➔ Expands the image to fit the whole image

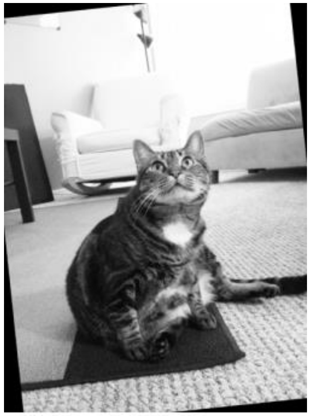


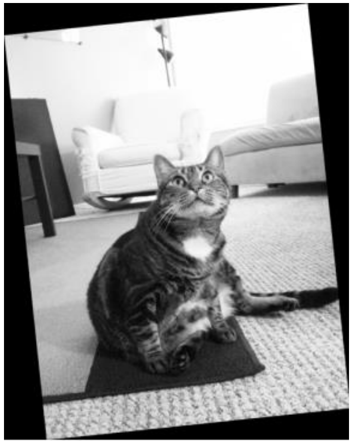
```
catIm.rotate(90).save('rotated90.png')
catIm.rotate(180).save('rotated180.png')
catIm.rotate(270).save('rotated270.png')
```





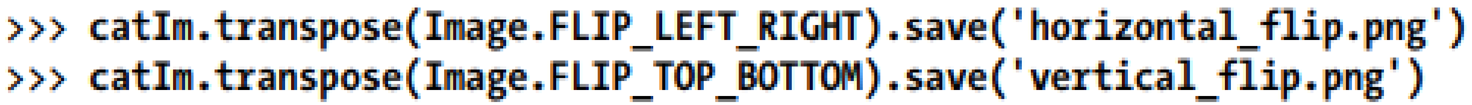
```
>>> catIm.rotate(6).save('rotated6.png')
>>> catIm.rotate(6, expand=True).save('rotated6_expanded.png')
```



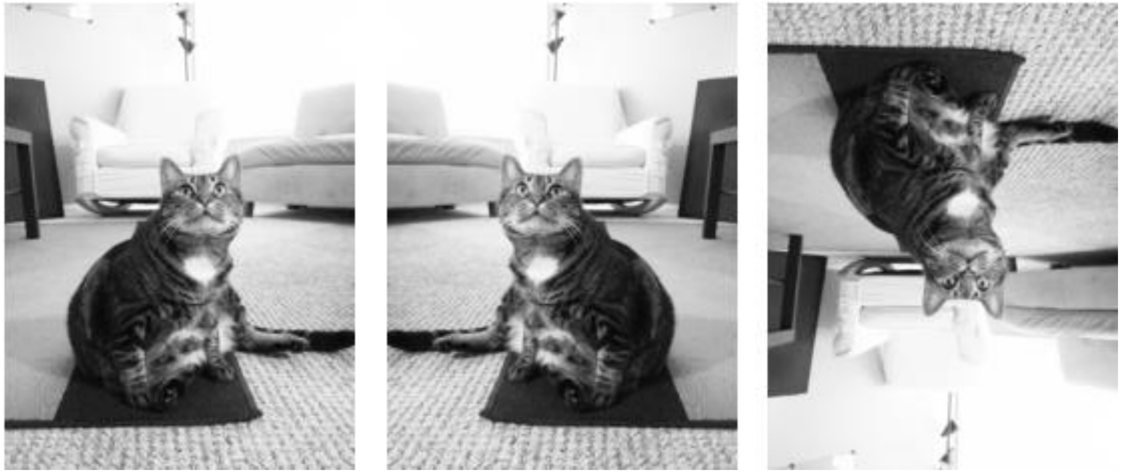


FLIPPING

- Flip the image with .transpose
  - Image.FLIP_LEFT_RIGHT ➔ Horizontal flip
  - Image.FLIP_TOP_BOTTOM ➔ Vertical Flip



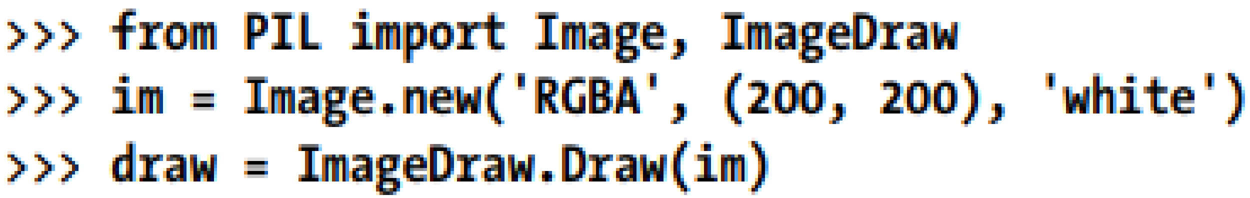
```
>>> catIm.transpose(Image.FLIP_LEFT_RIGHT).save('horizontal_flip.png')
>>> catIm.transpose(Image.FLIP_TOP_BOTTOM).save('vertical_flip.png')
```



DRAWING ON IMAGES

- Import Image and ImageDraw
- Create an new image
  - Ex. 200x200 and white background

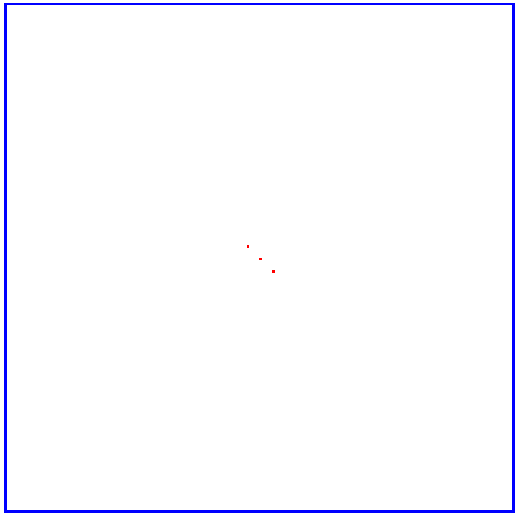
- Initialise the image to draw ➔ Creates a draw object
  - With .draw

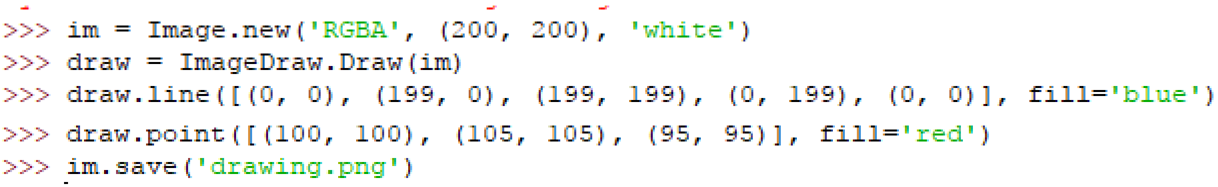


```
>>> from PIL import Image, ImageDraw
>>> im = Image.new('RGBA', (200, 200), 'white')
>>> draw = ImageDraw.Draw(im)
```

DRAWING ON IMAGES

- Drawing shapes
  - Points ➔ draw individual pixels
    - Draw.point(xy, fill)
  - Lines ➔ draw a line
    - Draw.line(xy, fill, width)

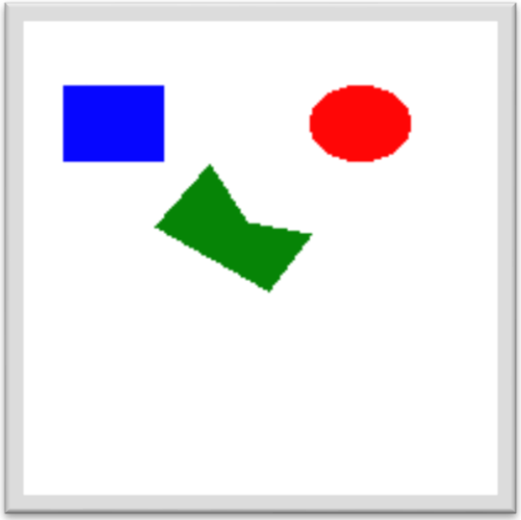


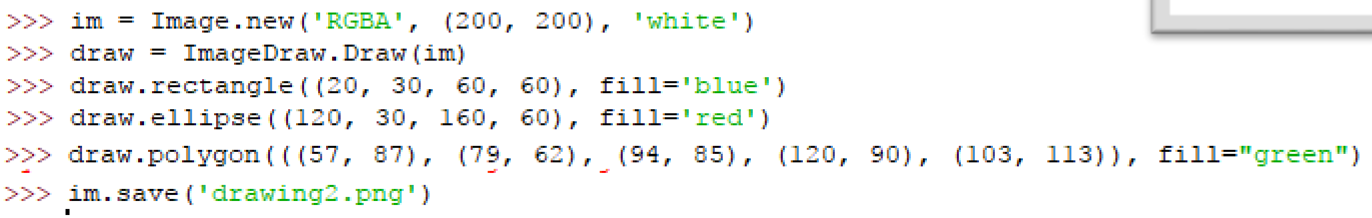


```
>>> im = Image.new('RGBA', (200, 200), 'white')
>>> draw = ImageDraw.Draw(im)
>>> draw.line([(0, 0), (199, 0), (199, 199), (0, 199), (0, 0)], fill='blue')
>>> draw.point([(100, 100), (105, 105), (95, 95)], fill='red')
>>> im.save('drawing.png')
```

DRAWING ON IMAGES

- Drawing Shapes
  - Rectangles ➔ draw a rectangle
    - Draw.rectangle(xy, fill, outline
  - Ellipses ➔ draw an ellipse or ‘circle’
    - Draw.ellipse(xy, fill, outline)
  - Polygons ➔ draw a polygon
    - Draw.polygon(xy, fill , outline)

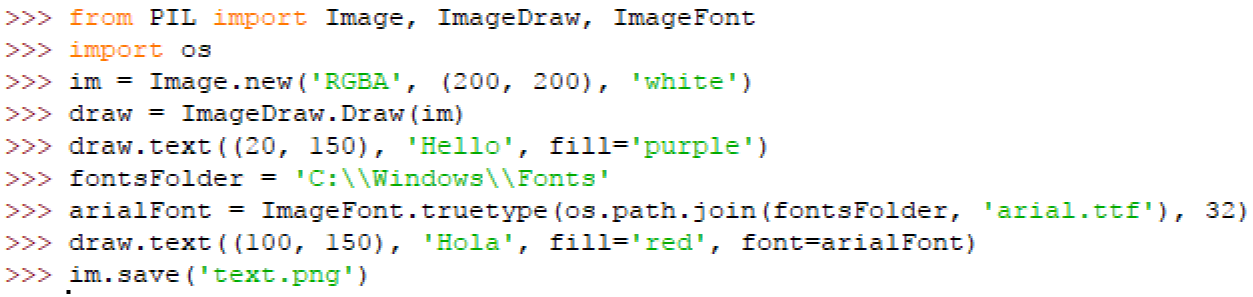




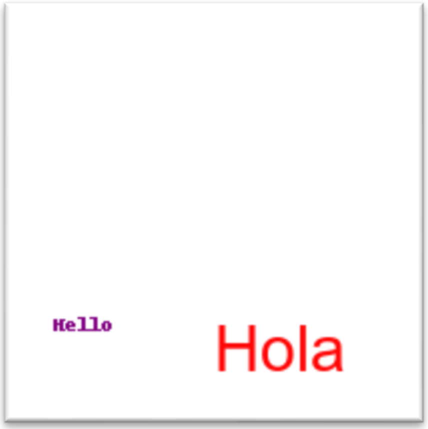
```
>>> im = Image.new('RGBA', (200, 200), 'white')
>>> draw = ImageDraw.Draw(im)
>>> draw.rectangle((20, 30, 60, 60), fill='blue')
>>> draw.ellipse((120, 30, 160, 60), fill='red')
>>> draw.polygon(((57, 87), (79, 62), (94, 85), (120, 90), (103, 113)), fill="green")
>>> im.save('drawing2.png')
```

DRAWING ON IMAGES

- Drawing text
  - draw.text()
    - xy, text, fill, font
  - from PIL import ImageFont
    - Truetype ➔ Path to fonts on your system.



```
>>> from PIL import Image, ImageDraw, ImageFont
>>> import os
>>> im = Image.new('RGBA', (200, 200), 'white')
>>> draw = ImageDraw.Draw(im)
>>> draw.text((20, 150), 'Hello', fill='purple')
>>> fontsFolder = 'C:\\Windows\\Fonts'
>>> arialFont = ImageFont.truetype(os.path.join(fontsFolder, 'arial.ttf'), 32)
>>> draw.text((100, 150), 'Hola', fill='red', font=arialFont)
>>> im.save('text.png')
```



Q&A

- What is an RGBA value?
- How can you get RGBA value of ‘CornflowerBlue’ from the Pillow module
- What is a box tuple?
- What function returns an Image object for, say, an image file named zophie.png?
- How can you find out the width and height of an Image object’s image?

Q&A

- What method would you call to get Image object for a 100x100 image, exluding the lower left quarter of it?
- After making changes to an Image object, how could you save it as an image file?
- What module contains Pillow’s shape drawing code?
- Image objects do not have drawing methods. What kind of object does? How do you get this kind of object?