00_fundamentals_skimage

September 11, 2019

To participate, you'll need to git clone (or download the .zip from GitHub):

https://github.com/imagexd/2019-tutorial-skimage

You can do that in git using:

If you have already cloned the material, please issue `git pull` now and reload the notebook to ensure that you have the latest updates.

[1]:
```
%matplotlib inline
```

- https://scikit-image.org
- Take a look at the gallery, API docs
- The underlying stack: NumPy, SciPy (especially `scipy.ndimage`), Cython

But, before we can start using `scikit-image`, we need to understand how images are represented. For this, we will use NumPy.

0.0.1 A quick 5-minute quiz, to refresh our NumPy skills!

Start with:

```
import numpy as np
```

Then answer:

1. What version of NumPy do you have installed? (Hint: `np.__version__`)
2. How do I construct a 300x500 array of zeros?
3. How do I generate an array with the numbers from 0 to 9?
4. How do I take the square of this array?
5. What is the data-type of this array? (Hint: `x.dtype`) What is the data-type of the array `[1, 2, 3]`?
6. What is the shape of the following array: `[[1, 2, 3], [4, 5, 6]]`
7. What, do you predict, is the result of `[[1, 2, 3], [4, 5, 6]] + [10, 11, 12]`? (Hint: broadcasting)
8. What, do you predict, is the result of `np.array([1,2,3], dtype=np.uint8) - 2`? (Hint: the 'u' stands for unsigned)
9. What, do you predict, is the result of `x = np.ones((5, 1)); x @ x.T` (Hint: @ is matrix multiplication)
10. What, do you predict, is the result of `x = np.array([1, 2, 3]); x[x < 3]` (Hint: what is `x > 3`?)

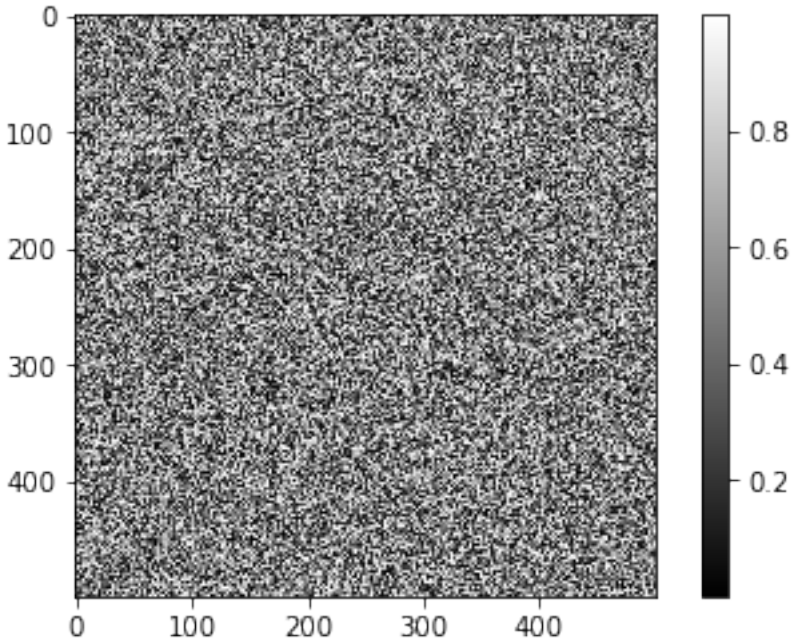
1 Images are numpy arrays

Images are represented in `scikit-image` using standard `numpy` arrays. This allows maximum inter-operability with other libraries in the scientific Python ecosystem, such as `matplotlib` and `scipy`.

Let's see how to build a grayscale image as a 2D array:

[2]:
```
import numpy as np
from matplotlib import pyplot as plt
```

[3]:
```
random_image = np.random.random([500, 500])

plt.imshow(random_image, cmap='gray')
plt.colorbar();
```

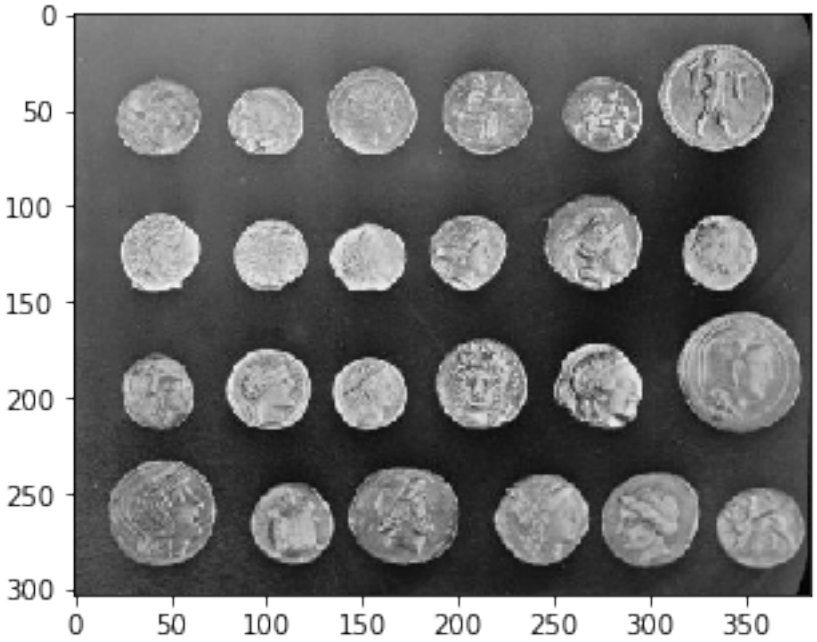


The same holds for "real-world" images:

[4]:
```
from skimage import data

coins = data.coins()

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

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

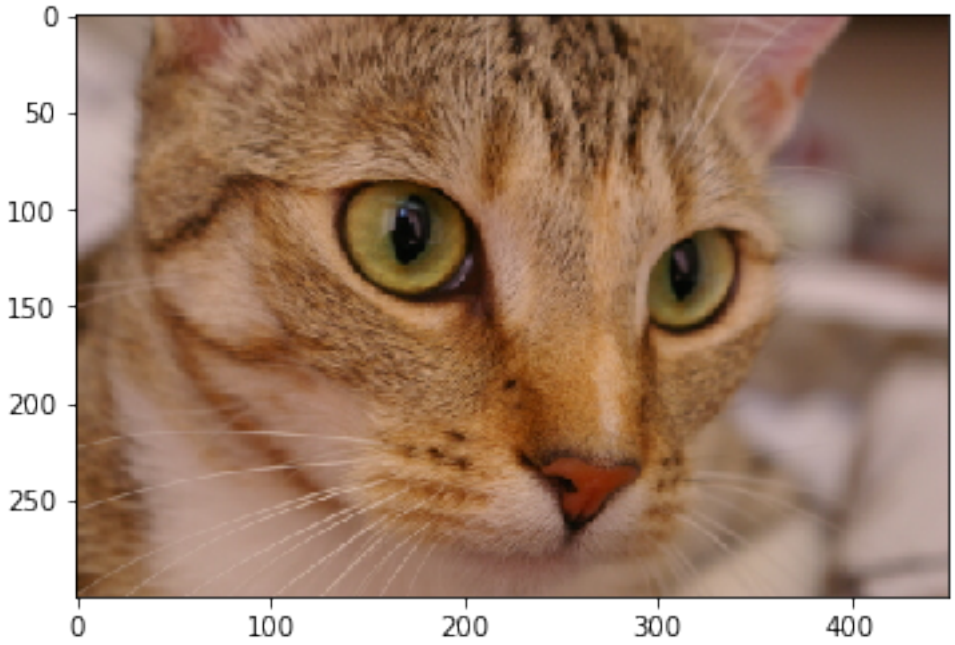
```
Type: <class 'numpy.ndarray'>
dtype: uint8
shape: (303, 384)
```



A color image is a 3D array, where the last dimension has size 3 and represents the red, green, and blue channels:

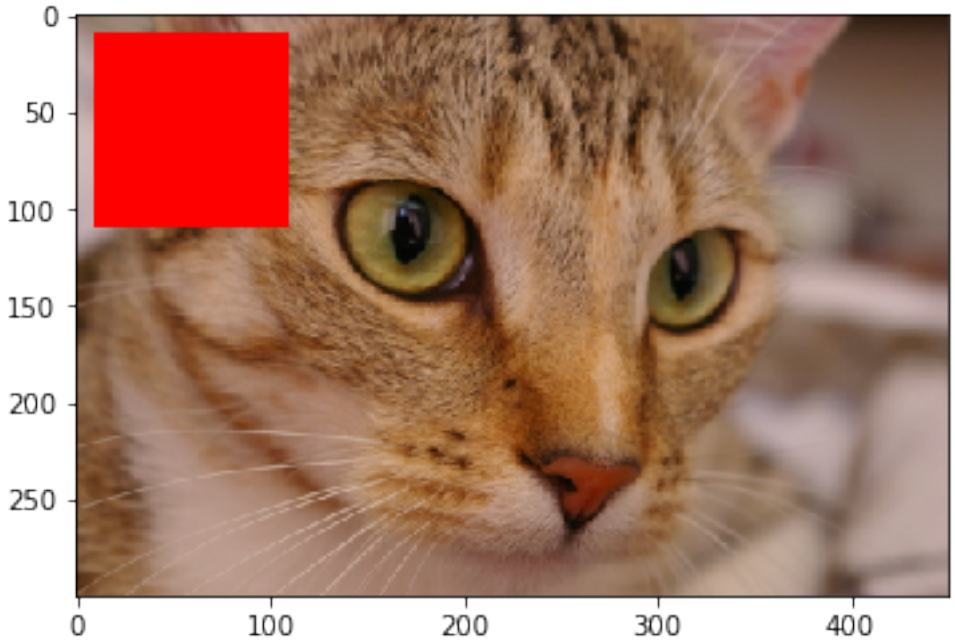
[5]:
```
cat = data.chelsea()

print("Shape:", cat.shape)
print("Values min/max:", cat.min(), cat.max())

plt.imshow(cat);
```

```
Shape: (300, 451, 3)
Values min/max: 0 231
```



These are *just NumPy arrays*. E.g., we can make a red square by using standard array slicing and manipulation:

[6]:
```
cat[10:110, 10:110, :] = [255, 0, 0]  # [red, green, blue]
plt.imshow(cat);
```



Images can also include transparent regions by adding a 4th dimension, called an *alpha layer*.

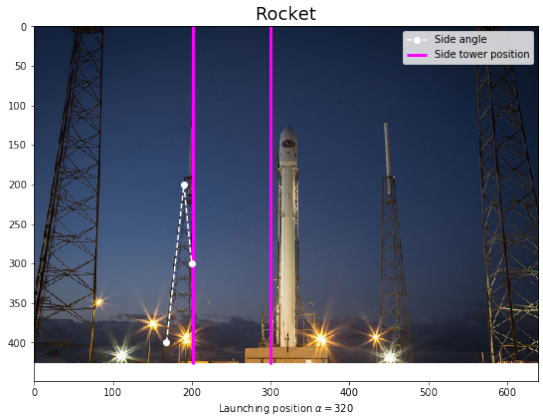
1.0.1 Other shapes, and their meanings

Image type	Coordinates
2D grayscale	(row, column)
2D multichannel	(row, column, channel)
3D grayscale (or volumetric)	(plane, row, column)
3D multichannel	(plane, row, column, channel)

1.1 Displaying images using matplotlib

[7]:
```
from skimage import data

img0 = data.chelsea()
img1 = data.rocket()
```

[8]:
```
import matplotlib.pyplot as plt

f, (ax0, ax1) = plt.subplots(1, 2, figsize=(20, 10))

ax0.imshow(img0)
ax0.set_title('Cat', fontsize=18)
ax0.scatter([10, 10, 10, 10], [10, 20, 30, 40], color='white')
ax0.axis('off')

ax1.imshow(img1)
ax1.set_title('Rocket', fontsize=18)
ax1.set_xlabel(r'Launching position $\alpha=320$')

ax1.vlines([202, 300], 0, img1.shape[0], colors='magenta',
           linewidth=3, label='Side tower position')
ax1.plot([168, 190, 200], [400, 200, 300], color='white',
         linestyle='--', marker='o', label='Side angle')

ax1.legend();
```



For more on plotting, see the Matplotlib documentation and pyplot API.

1.2 Data types and image values

In literature, one finds different conventions for representing image values:

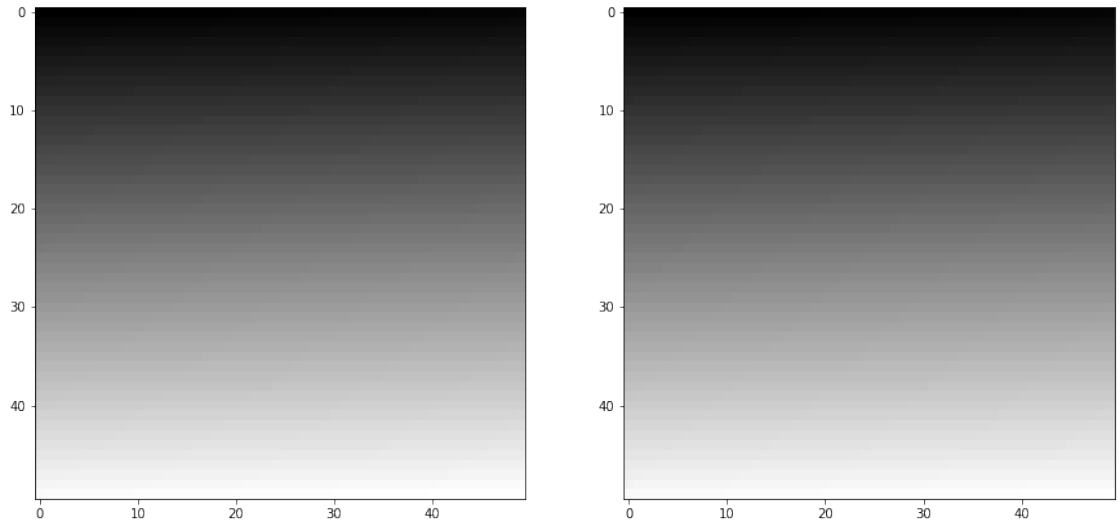
```
0 - 255   where  0 is black, 255 is white
0 - 1     where  0 is black, 1 is white
```

`scikit-image` supports both conventions–the choice is determined by the data-type of the array.

E.g., here, I generate two valid images:

[9]:
```
linear0 = np.linspace(0, 1, 2500).reshape((50, 50))
linear1 = np.linspace(0, 255, 2500).reshape((50, 50)).astype(np.uint8)

print("Linear0:", linear0.dtype, linear0.min(), linear0.max())
print("Linear1:", linear1.dtype, linear1.min(), linear1.max())

fig, (ax0, ax1) = plt.subplots(1, 2, figsize=(15, 15))
ax0.imshow(linear0, cmap='gray')
ax1.imshow(linear1, cmap='gray');
```

```
Linear0: float64 0.0 1.0
Linear1: uint8 0 255
```



The library is designed in such a way that any data-type is allowed as input, as long as the range is correct (0-1 for floating point images, 0-255 for unsigned bytes, 0-65535 for unsigned 16-bit integers).

You can convert images between different representations by using `img_as_float`, `img_as_ubyte`, etc.:

[10]:
```
from skimage import img_as_float, img_as_ubyte

image = data.chelsea()

image_ubyte = img_as_ubyte(image)
image_float = img_as_float(image)

print("type, min, max:", image_ubyte.dtype, image_ubyte.min(), image_ubyte.
 ↪max())
print("type, min, max:", image_float.dtype, image_float.min(), image_float.
 ↪max())
print()
print("231/255 =", 231/255.)
```

```
type, min, max: uint8 0 231
type, min, max: float64 0.0 0.9058823529411765

231/255 = 0.9058823529411765
```

Your code would then typically look like this:

```
def my_function(any_image):
    float_image = img_as_float(any_image)
    # Proceed, knowing image is in [0, 1]
```

We recommend using the floating point representation, given that `scikit-image` mostly uses that format internally.

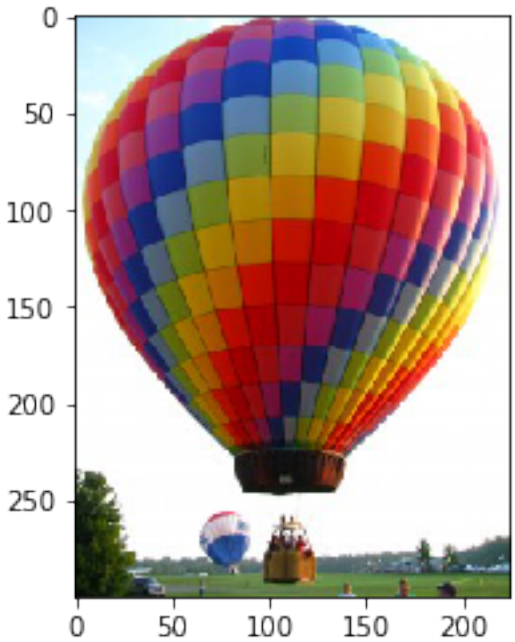
1.3 Image I/O

Mostly, we won't be using input images from the scikit-image example data sets. Those images are typically stored in JPEG or PNG format. Since scikit-image operates on NumPy arrays, *any* image reader library that provides arrays will do. Options include imageio, matplotlib, pillow, etc.

scikit-image conveniently wraps many of these in the `io` submodule, and will use whichever of the libraries mentioned above are installed:

[11]:
```
from skimage import io

image = io.imread('data/balloon.jpg')

print(type(image))
print(image.dtype)
print(image.shape)
print(image.min(), image.max())

plt.imshow(image);
```

```
<class 'numpy.ndarray'>
uint8
(300, 225, 3)
0 255
```



We also have the ability to load multiple images, or multi-layer TIFF images:

[12]:
```
from skimage import data_dir

!ls $data_dir
```

```
__init__.py                     grass.png
__pycache__                     gray_morph_output.npz
_binary_blobs.py                green_palette.png
_blobs_3d_fiji_skeleton.tif     horse.png
_detect.py                      hubble_deep_field.jpg
astronaut.png                   ihc.png
astronaut_GRAY_hog_L1.npy       lbpcascade_frontalface_opencv.xml
astronaut_GRAY_hog_L2-Hys.npy   lfw_subset.npy
block.png                       logo.png
brick.png                       microaneurysms.png
bw_text.png                     moon.png
bw_text_skeleton.npy            motorcycle_disp.npz
camera.png                      motorcycle_left.png
cells_qpi.npz                   motorcycle_right.png
cells_qpi_zipped.zip            mssim_matlab_output.npz
checker_bilevel.png             multi.fits
chelsea.png                     multipage.tif
chessboard_GRAY.png             multipage_rgb.tif
chessboard_GRAY_U16.tif         no_time_for_that_tiny.gif
chessboard_GRAY_U16B.tif        orb_descriptor_positions.txt
chessboard_GRAY_U8.npy          page.png
chessboard_GRAY_U8.npz          palette_color.png
chessboard_RGB.png              palette_gray.png
chessboard_RGB_U8.npy           phantom.png
chessboard_RGB_U8.npz           rank_filter_tests.npz
clock_motion.png                retina.jpg
coffee.png                      rocket.jpg
coins.png                       rough-wall.png
color.png                       simple.fits
diamond-matlab-output.npz       tests
disk-matlab-output.npz          text.png
foo3x5x4indexed.png             truncated.jpg
```

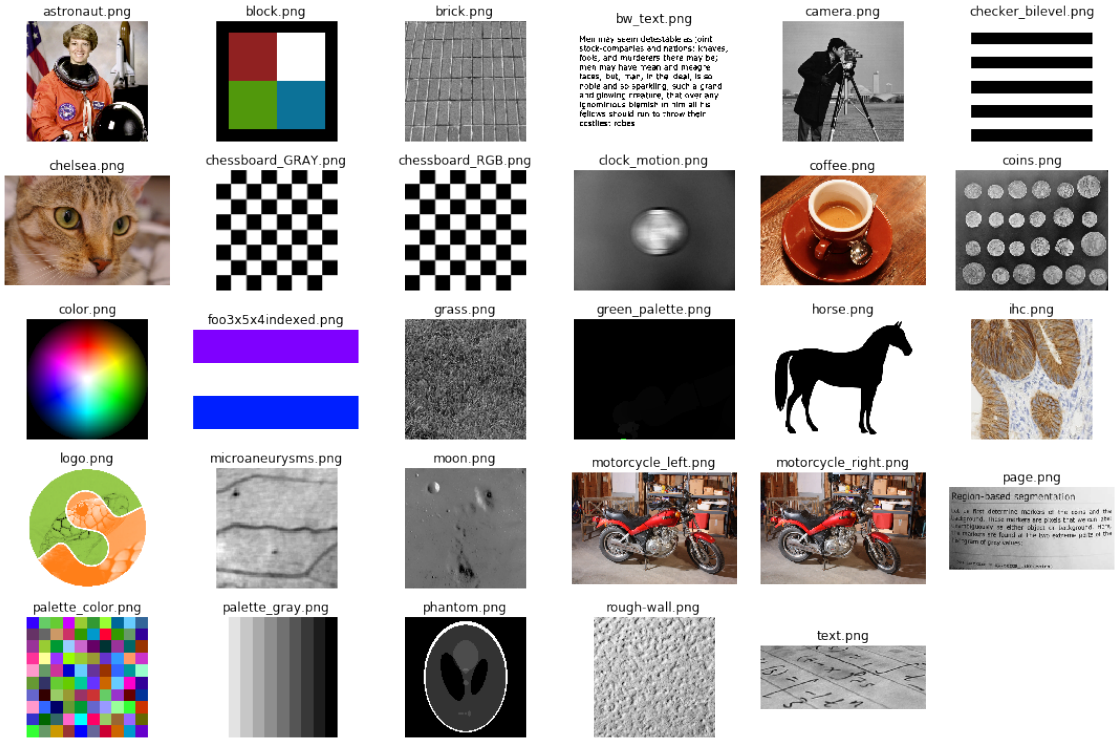
[13]:
```
ic = io.ImageCollection(data_dir + '/*.png')

print('Type:', type(ic))

ic.files
```

```
Type: <class 'skimage.io.collection.ImageCollection'>
```

```
[13]: ['/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/astronaut.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/block.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/brick.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/bw_text.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/camera.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/checker_bilevel.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/chelsea.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/chessboard_GRAY.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/chessboard_RGB.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/clock_motion.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/coffee.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/coins.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/color.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/foo3x5x4indexed.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/grass.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/green_palette.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/horse.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/ihc.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/logo.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/microaneurysms.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/moon.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/motorcycle_left.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
      packages/skimage/data/motorcycle_right.png',
       '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
```

```
packages/skimage/data/page.png',
 '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/skimage/data/palette_color.png',
 '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/skimage/data/palette_gray.png',
 '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/skimage/data/phantom.png',
 '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/skimage/data/rough-wall.png',
 '/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/skimage/data/text.png']
```

[14]:
```
import os

f, axes = plt.subplots(nrows=5, ncols=len(ic) // 5 + 1, figsize=(15, 10))

# subplots returns the figure and an array of axes
# we use `axes.ravel()` to turn these into a list
axes = axes.ravel()

for ax in axes:
    ax.axis('off')

for i, image in enumerate(ic):
    axes[i].imshow(image, cmap='gray')
    axes[i].set_title(os.path.basename(ic.files[i]))

plt.tight_layout()
```



1.4 Exercise: draw the letter H

Define a function that takes as input an RGB image and a pair of coordinates (row, column), and returns a copy with a green letter H overlaid at those coordinates. The coordinates point to the top-left corner of the H.

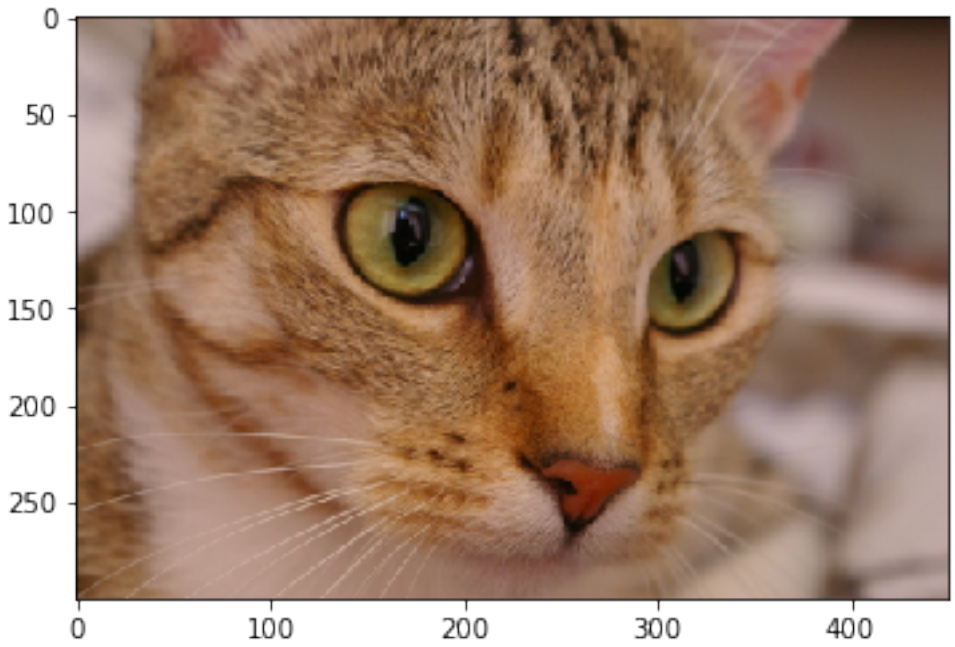
The arms and strut of the H should have a width of 3 pixels, and the H itself should have a height of 24 pixels and width of 20 pixels.

Start with the following template:

[15]:
```
def draw_H(image, coords, color=(0, 255, 0)):
    out = image.copy()

    ...

    return out
```

Test your function like so:

[16]:
```
cat = data.chelsea()
cat_H = draw_H(cat, (50, -50))
plt.imshow(cat_H);
```



1.5 Exercise: visualizing RGB channels

Display the different color channels of the image along (each as a gray-scale image). Start with the following template:

[17]:
```
# --- read in the image ---

image = plt.imread('data/Bells-Beach.jpg')

# --- assign each color channel to a different variable ---

r = ...
g = ...
b = ...

# --- display the image and r, g, b channels ---

f, axes = plt.subplots(1, 4, figsize=(16, 5))

for ax in axes:
    ax.axis('off')

(ax_r, ax_g, ax_b, ax_color) = axes

ax_r.imshow(r, cmap='gray')
```

```
ax_r.set_title('red channel')

ax_g.imshow(g, cmap='gray')
ax_g.set_title('green channel')

ax_b.imshow(b, cmap='gray')
ax_b.set_title('blue channel')

# --- Here, we stack the R, G, and B layers again
#     to form a color image ---
ax_color.imshow(np.stack([r, g, b], axis=2))
ax_color.set_title('all channels');
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call last)

       <ipython-input-17-e057ff15c311> in <module>
        18 (ax_r, ax_g, ax_b, ax_color) = axes
        19
   ---> 20 ax_r.imshow(r, cmap='gray')
        21 ax_r.set_title('red channel')
        22


       ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/
↪__init__.py in inner(ax, data, *args, **kwargs)
      1599     def inner(ax, *args, data=None, **kwargs):
      1600         if data is None:
   -> 1601             return func(ax, *map(sanitize_sequence, args), **kwargs)
      1602
      1603         bound = new_sig.bind(ax, *args, **kwargs)


       ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/cbook/
↪deprecation.py in wrapper(*args, **kwargs)
       367                 f"%(removal)s.  If any parameter follows {name!r}, they "
       368                    f"should be pass as keyword, not positionally.")
   --> 369         return func(*args, **kwargs)
       370
       371     return wrapper


       ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/cbook/
↪deprecation.py in wrapper(*args, **kwargs)
```

```
      367                f"%(removal)s.  If any parameter follows {name!r}, they "
      368                   f"should be pass as keyword, not positionally.")
  --> 369         return func(*args, **kwargs)
      370
      371     return wrapper


      ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/axes/
↪_axes.py in imshow(self, X, cmap, norm, aspect, interpolation, alpha, vmin,␣
↪vmax, origin, extent, shape, filternorm, filterrad, imlim, resample, url,␣
↪**kwargs)
     5669                                resample=resample, **kwargs)
     5670
  -> 5671         im.set_data(X)
     5672         im.set_alpha(alpha)
     5673         if im.get_clip_path() is None:


      ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/image.
↪py in set_data(self, A)
      683                 not np.can_cast(self._A.dtype, float, "same_kind")):
      684             raise TypeError("Image data of dtype {} cannot be␣
↪converted to "
  --> 685                             "float".format(self._A.dtype))
      686
      687         if not (self._A.ndim == 2


      TypeError: Image data of dtype object cannot be converted to float
```

Now, take a look at the following R, G, and B channels. How would their combination look? (Write some code to confirm your intuition.)

[18]:
```
from skimage import draw

red = np.zeros((300, 300))
green = np.zeros((300, 300))
blue = np.zeros((300, 300))

r, c = draw.circle(100, 100, 100)
red[r, c] = 1

r, c = draw.circle(100, 200, 100)
green[r, c] = 1

r, c = draw.circle(200, 150, 100)
blue[r, c] = 1

f, axes = plt.subplots(1, 3)
for (ax, channel) in zip(axes, [red, green, blue]):
    ax.imshow(channel, cmap='gray')
    ax.axis('off')
```

[19]:
```
# Hint: np.stack([...], axis=2)
```

1.6 Exercise: Convert to grayscale ("black and white")

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:

$$Y = 0.2126R + 0.7152G + 0.0722B$$

Use Python 3.5's matrix multiplication, `@`, to convert an RGB image to a grayscale luminance image according to the formula above.

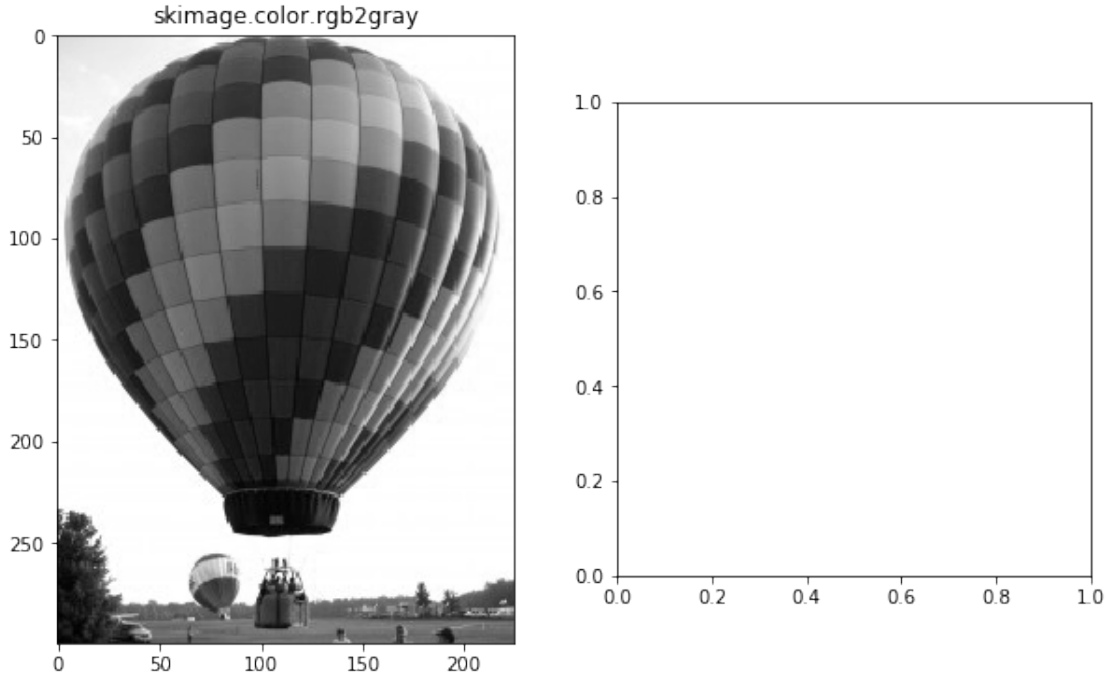
Compare your results to that obtained with `skimage.color.rgb2gray`.

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`).

[20]:
```
from skimage import color, img_as_float

image = img_as_float(io.imread('data/balloon.jpg'))

gray = color.rgb2gray(image)
my_gray = ...

# --- display the results ---

f, (ax0, ax1) = plt.subplots(1, 2, figsize=(10, 6))

ax0.imshow(gray, cmap='gray')
ax0.set_title('skimage.color.rgb2gray')

ax1.imshow(my_gray, cmap='gray')
ax1.set_title('my rgb2gray')
```

```
      ␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call last)

       <ipython-input-20-a5fdfebe8302> in <module>
        13 ax0.set_title('skimage.color.rgb2gray')
        14
   ---> 15 ax1.imshow(my_gray, cmap='gray')
        16 ax1.set_title('my rgb2gray')


       ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/
↪__init__.py in inner(ax, data, *args, **kwargs)
      1599     def inner(ax, *args, data=None, **kwargs):
      1600         if data is None:
   -> 1601             return func(ax, *map(sanitize_sequence, args), **kwargs)
      1602
      1603         bound = new_sig.bind(ax, *args, **kwargs)


       ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/cbook/
↪deprecation.py in wrapper(*args, **kwargs)
       367                 f"%(removal)s.  If any parameter follows {name!r}, they "
        368                    f"should be pass as keyword, not positionally.")
   --> 369         return func(*args, **kwargs)
        370
```

```
      371     return wrapper


      ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/cbook/
↪deprecation.py in wrapper(*args, **kwargs)
      367                 f"%(removal)s.  If any parameter follows {name!r}, they "
      368                    f"should be pass as keyword, not positionally.")
  --> 369         return func(*args, **kwargs)
      370
      371     return wrapper


      ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/axes/
↪_axes.py in imshow(self, X, cmap, norm, aspect, interpolation, alpha, vmin,␣
↪vmax, origin, extent, shape, filternorm, filterrad, imlim, resample, url,␣
↪**kwargs)
     5669                                 resample=resample, **kwargs)
     5670
  -> 5671         im.set_data(X)
     5672         im.set_alpha(alpha)
     5673         if im.get_clip_path() is None:


      ~/anaconda3/envs/imagexd19/lib/python3.7/site-packages/matplotlib/image.
↪py in set_data(self, A)
      683                 not np.can_cast(self._A.dtype, float, "same_kind")):
      684             raise TypeError("Image data of dtype {} cannot be␣
↪converted to "
  --> 685                             "float".format(self._A.dtype))
      686
      687         if not (self._A.ndim == 2


      TypeError: Image data of dtype object cannot be converted to float
```



1.7 Just for fun

1.7.1 Demo: skimage + keras

[25]:
```
from tensorflow.keras.applications.inception_v3 import InceptionV3,␣
 ↪preprocess_input, decode_predictions
net = InceptionV3()
```

```
/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:526: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:527: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:528: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
```

```
packages/tensorflow/python/framework/dtypes.py:529: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:530: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/tensorflow/python/framework/dtypes.py:535: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

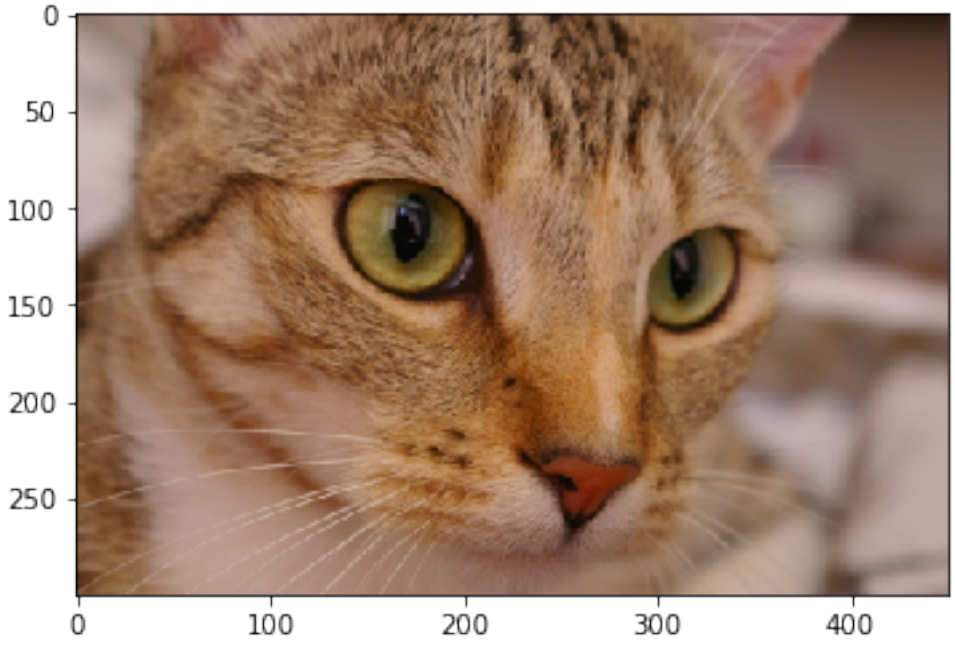
```
WARNING:tensorflow:From /Users/dani/anaconda3/envs/imagexd19/lib/python3.7/site-
packages/tensorflow/python/ops/resource_variable_ops.py:435: colocate_with (from
tensorflow.python.framework.ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Colocations handled automatically by placer.
Downloading data from https://github.com/fchollet/deep-learning-
models/releases/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels.h5
96116736/96112376 [==============================] - 12s 0us/step
```

[26]:
```
from skimage import transform

def inception_predict(image):
    # Rescale image to 299x299, as required by InceptionV3
    image_prep = transform.resize(image, (299, 299, 3), mode='reflect')

    # Scale image values to [-1, 1], as required by InceptionV3
    image_prep = (img_as_float(image_prep) - 0.5) * 2

    predictions = decode_predictions(
        net.predict(image_prep[None, ...])
    )

    plt.imshow(image, cmap='gray')

    for pred in predictions[0]:
        (n, klass, prob) = pred
        print(f'{klass:>15} ({prob:.3f})')
```

[27]:
```
from skimage import data, img_as_float
inception_predict(data.chelsea())
```

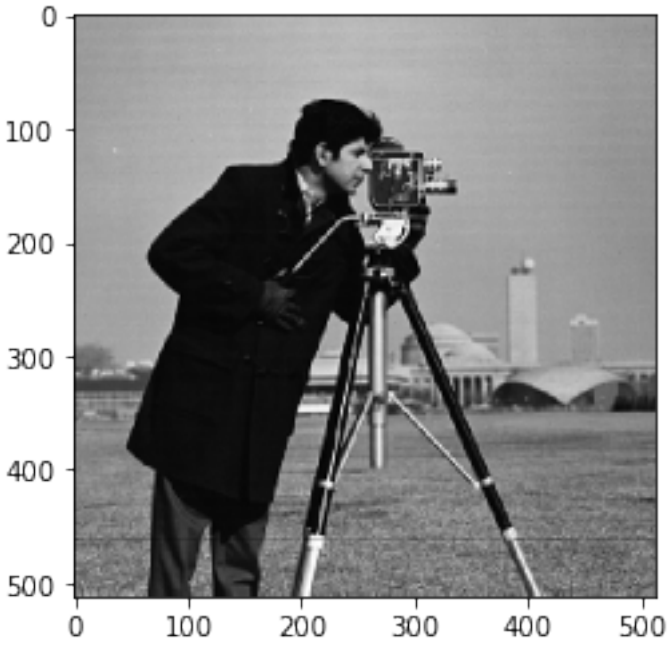
```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/dat
```

```
a/imagenet_class_index.json
40960/35363 [==================================] - 0s 0us/step
   Egyptian_cat (0.904)
          tabby (0.054)
      tiger_cat (0.035)
           lynx (0.000)
    plastic_bag (0.000)
```



[28]:
```
inception_predict(data.camera())
```

```
        tripod (0.937)
        crutch (0.002)
    binoculars (0.002)
  reflex_camera (0.001)
      backpack (0.000)
```



[29]:
```
inception_predict(data.coffee())
```

```
       espresso (0.982)
            cup (0.002)
     coffee_mug (0.001)
         eggnog (0.001)
espresso_maker (0.001)
```


0
50
100
150
200
250
300
350
0
100
200
300
400
500

[ ]: