image_informatics_lab_1

November 21, 2022

1 Image Informatics - Lab 1

1.1 numpy

1.1.1 The Basics

[1]:
```
import numpy as np

a = np.array([2,3,4])
print(a)
print(a.dtype)
print(a.shape)
print(a.size)
```

```
[2 3 4]
int64
(3,)
3
```

1.1.2 Array Creation

[2]:
```
a = np.zeros((3, 4))
print(a)
print(a.shape)

a = np.ones((3, 4))
print(a)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
(3, 4)
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
```

[3]:
```
a = np.ones((2,3,4), dtype=np.int16)
print(a)
print(a.dtype)
```

```
[[[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]

 [[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]]
int16
```

1.1.3 Basic Operations

[4]:
```
a = np.array([20,30,40,50])
b = np.arange(4)
print(a)
print(b)

c = a-b
print(c)

d = a<35
print(d)
```

```
[20 30 40 50]
[0 1 2 3]
[20 29 38 47]
[ True  True False False]
```

[5]:
```
a = np.array([[1,1], [0,1]])
b = np.array([[2,0], [3,4]])
print(a)
print(b)

c = a*b # elementwise product
print(c)
```

```
[[1 1]
 [0 1]]
[[2 0]
 [3 4]]
[[2 0]
 [0 4]]
```

1.1.4 Universal Functions

[6]:
```
a = np.arange(3)
print(a)

b = np.exp(a)
```

```
print(b)

c = np.sqrt(a)
print(c)
```

```
[0 1 2]
[1.         2.71828183 7.3890561 ]
[0.         1.         1.41421356]
```

1.1.5 Indexing, Slicing and Iterating

[7]:
```
a = np.arange(10)**3
print(a)
print(a[2])
print(a[2:5])

a[:6:2] = 1000
print(a)

print(a[ : :-1])

for i in a:
  print(i**(1/3.))
```

```
[  0   1   8  27  64 125 216 343 512 729]
8
[ 8 27 64]
[1000    1 1000   27 1000  125  216  343  512  729]
[ 729  512  343  216  125 1000   27 1000    1 1000]
9.999999999999998
1.0
9.999999999999998
3.0
9.999999999999998
4.999999999999999
5.999999999999999
6.999999999999999
7.999999999999999
8.999999999999998
```

[8]:
```
a = np.random.rand(5,4)
print(a)
print(a[2,2])
print(a[0:2,2])
print(a[:,2])
print(a[1:3,:])
print(a[-1])
```

```
[[0.93202252 0.32357935 0.92762528 0.14093997]
 [0.23737903 0.37213502 0.4359163  0.44682718]
 [0.27402657 0.33459239 0.16418221 0.94561049]
 [0.56125738 0.17764204 0.04799005 0.52448039]
 [0.63636354 0.93643268 0.81274158 0.56923676]]
0.1641822063584656
[0.92762528 0.4359163 ]
[0.92762528 0.4359163  0.16418221 0.04799005 0.81274158]
[[0.23737903 0.37213502 0.4359163  0.44682718]
 [0.27402657 0.33459239 0.16418221 0.94561049]]
[0.63636354 0.93643268 0.81274158 0.56923676]
```

1.1.6 Indexing with Boolean Arrays

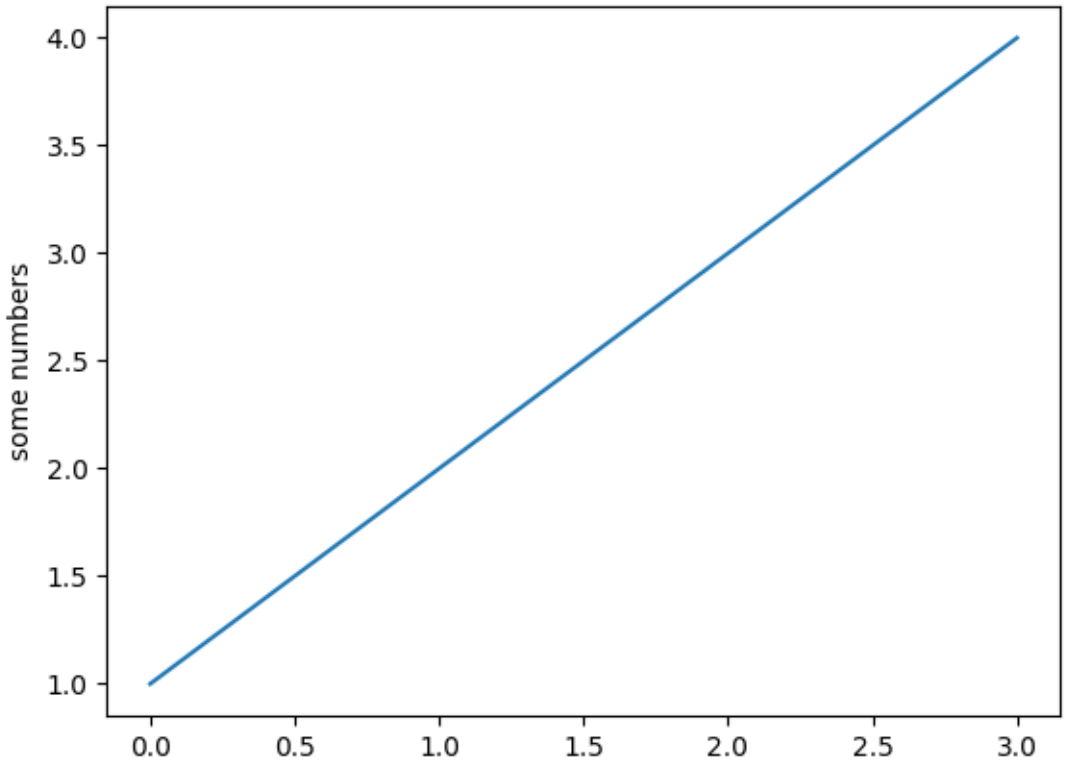
[9]:
```
a = np.arange(12).reshape(3,4)
b = a > 4
print(b)

c = a[b]
print(c)
```

```
[[False False False False]
 [False  True  True  True]
 [ True  True  True  True]]
[ 5  6  7  8  9 10 11]
```

1.2 pyplot

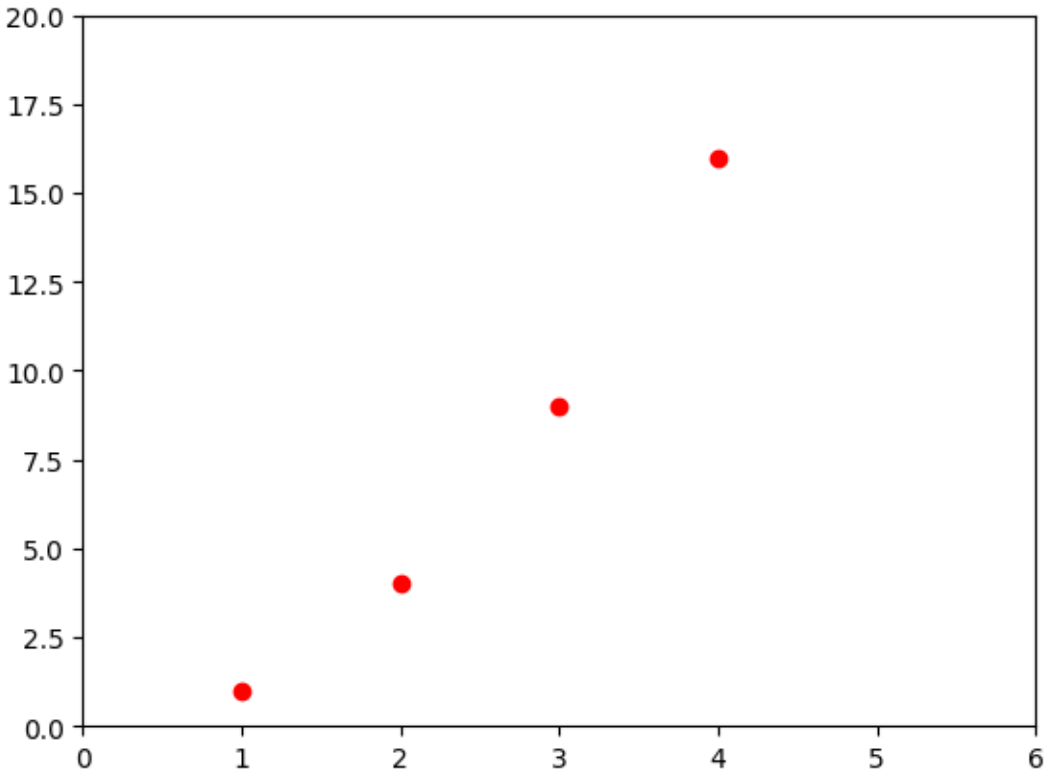
1.2.1 Basics

[10]:
```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()
```



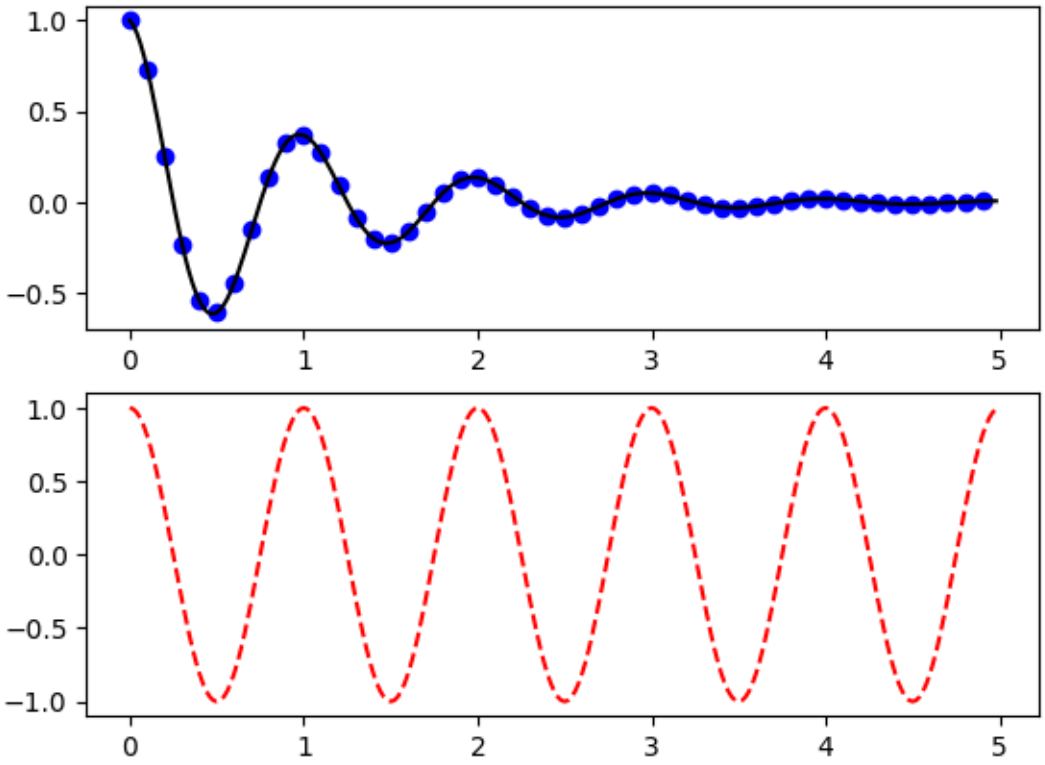
1.2.2 Formatting

[11]:
```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```



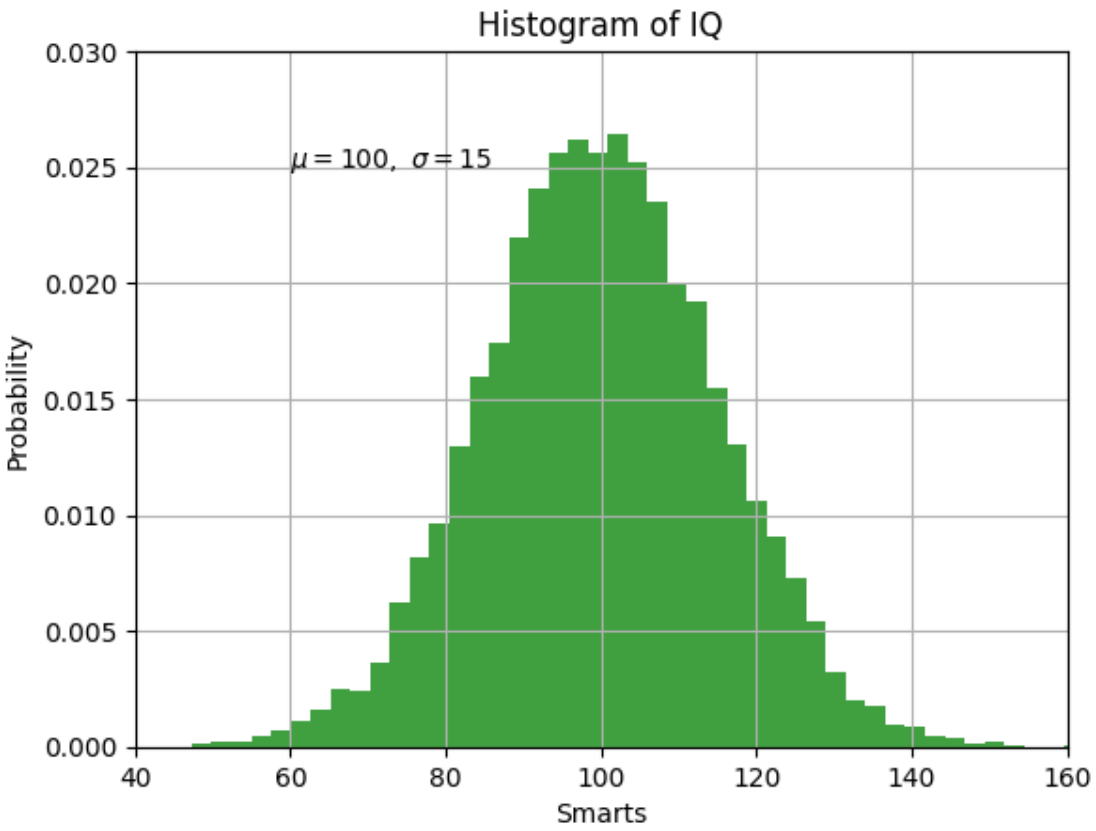
1.2.3 Multiple figures and axes

[12]:
```
def f(t):
  return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure()
plt.subplot(211)
plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), 'r--')
plt.show()
```



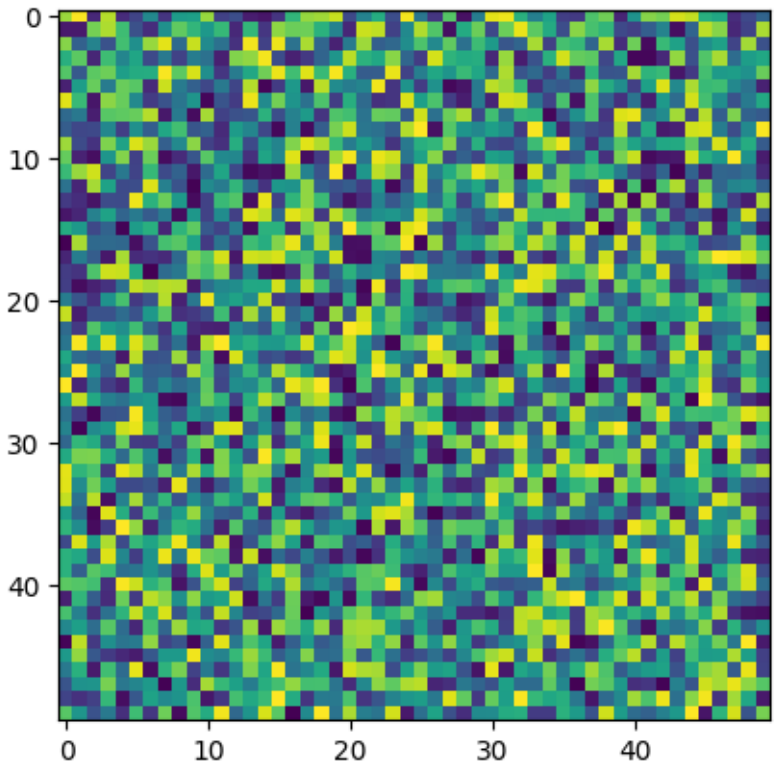
1.2.4 Working with text

[13]:
```
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

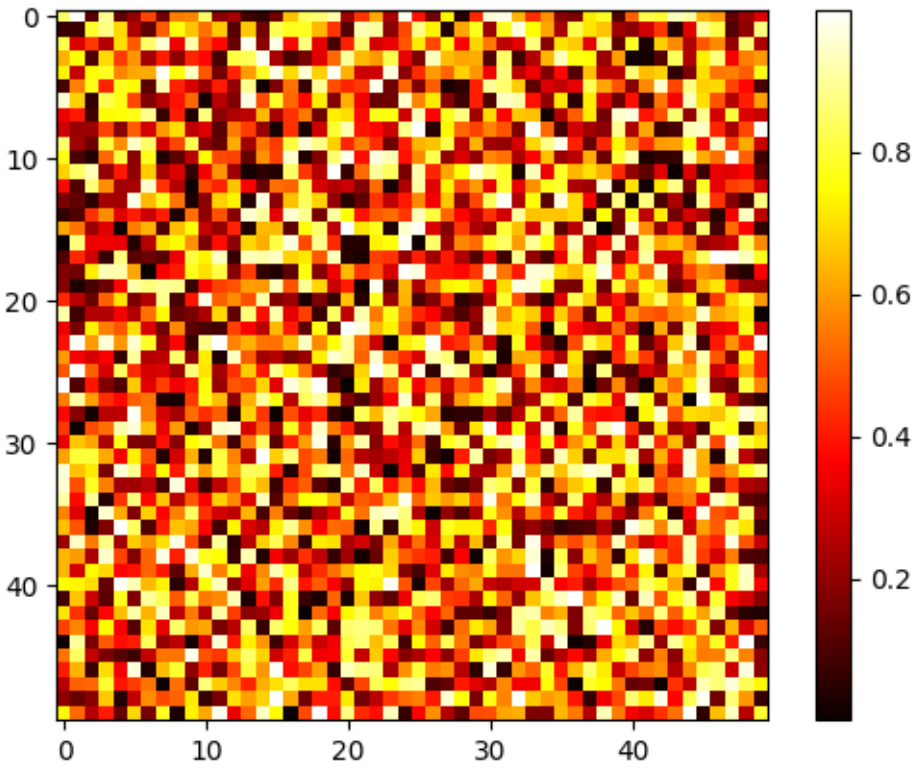


1.2.5 Working with image

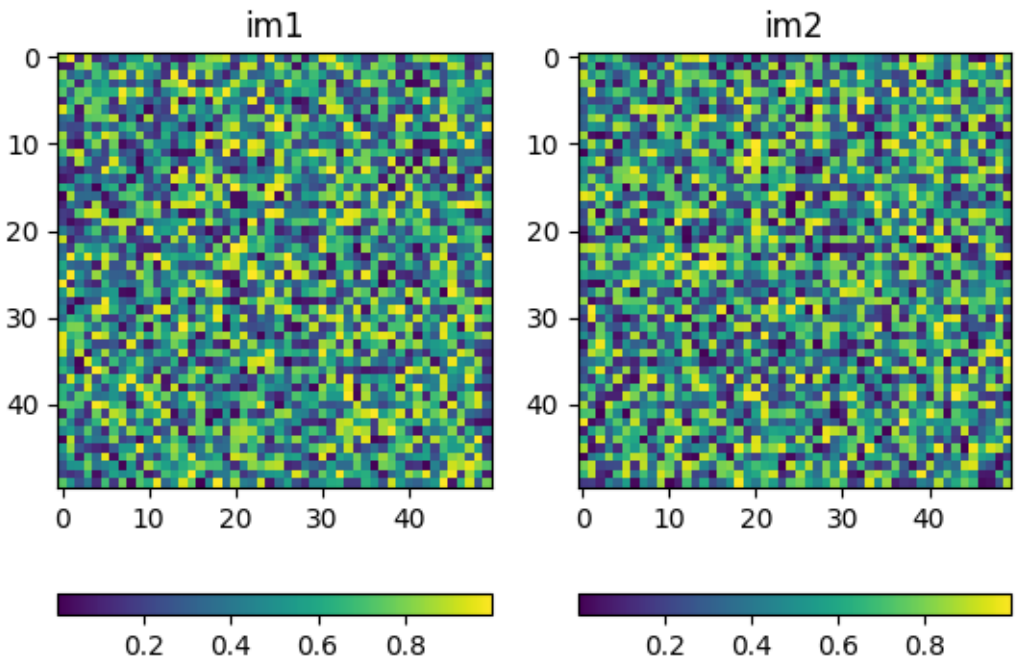
[14]:
```
import numpy as np

im = np.random.random((50,50))
p = plt.imshow(im)
```



[15]:
```
p = plt.imshow(im, cmap="hot")
c = plt.colorbar()
```



[16]:
```
im1 = im.copy()
im2 = np.random.random((50,50))

fig = plt.figure()
ax = fig.add_subplot(1, 2, 1)
p = plt.imshow(im1)
ax.set_title('im1')
c = plt.colorbar(orientation='horizontal')

ax = fig.add_subplot(1, 2, 2)
p = plt.imshow(im2)
ax.set_title('im2')
c = plt.colorbar(orientation='horizontal')
```

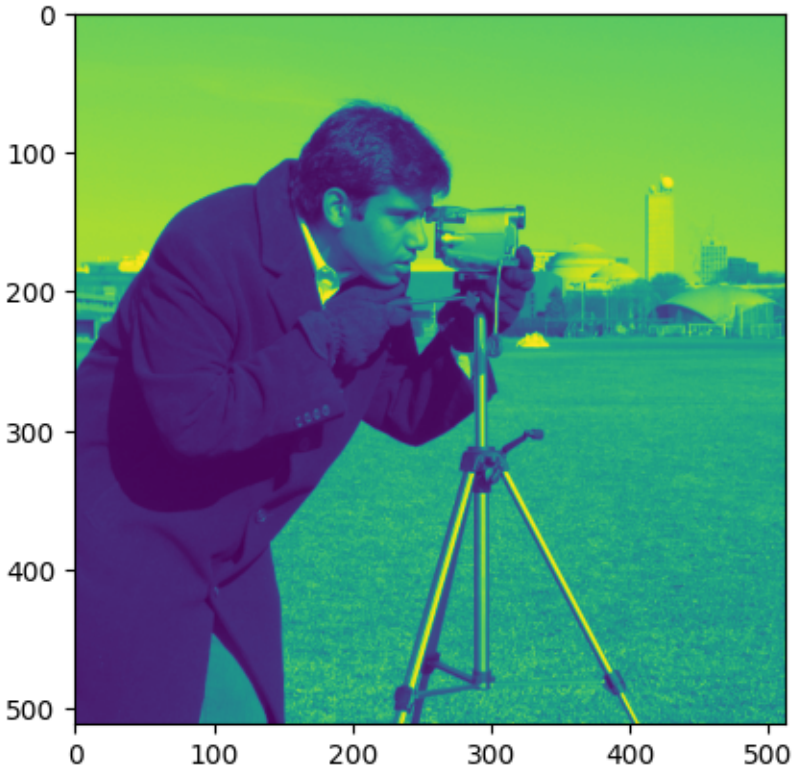


1.3 scikit-image

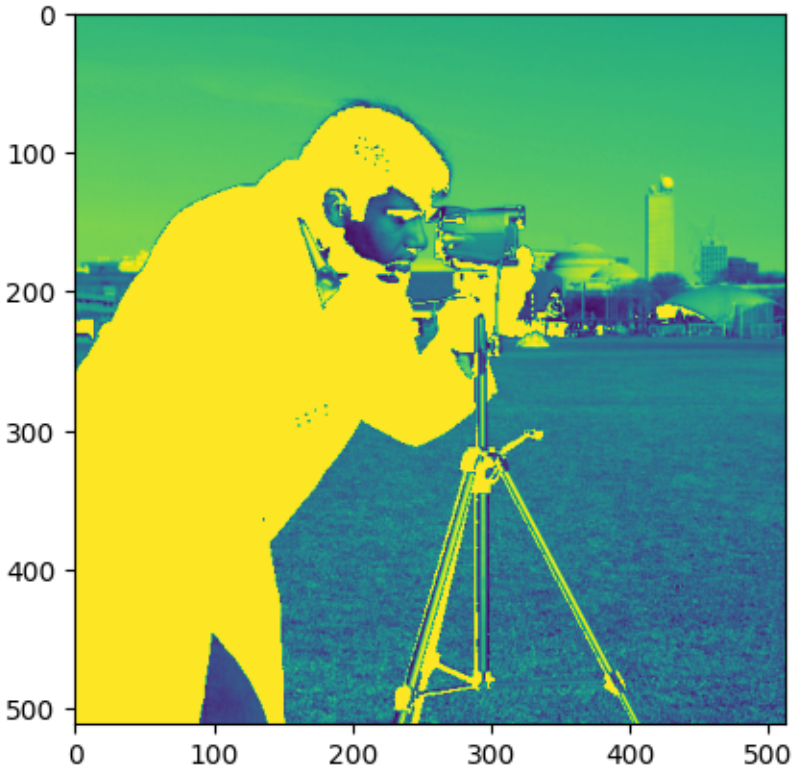
1.3.1 Basics

[17]:
```
from skimage import data

camera = data.camera()
p = plt.imshow(camera)
print(camera.shape)
print(camera.min())
print(camera.max())
print(camera.mean())
print(camera[10, 20])
```

```
(512, 512)
0
255
129.06072616577148
200
```



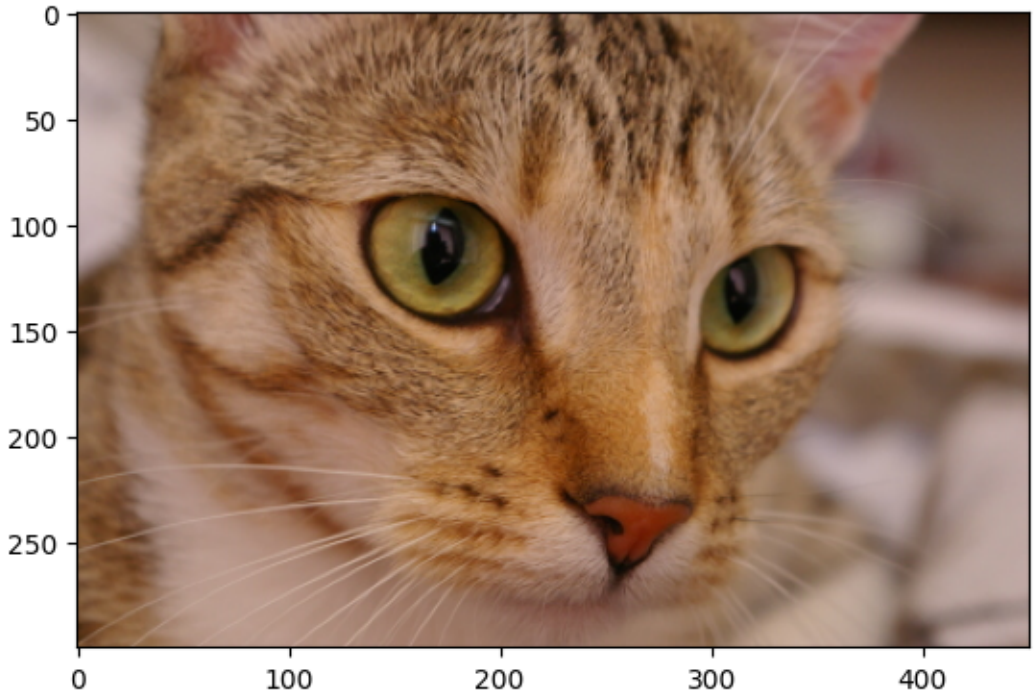
[18]:
```
mask = camera < 87
camera[mask] = 255
p = plt.imshow(camera)
```



1.3.2 Color images

[19]:
```
cat = data.chelsea()
p = plt.imshow(cat)
print(cat.shape)
```

```
(300, 451, 3)
```



1.4 Google Colab

1.4.1 Loading images

[ ]:
```
from google.colab import files

files = files.upload()
names = list(files.keys())
print(names)
```

[ ]:
```
import matplotlib.pyplot as plt
from skimage import io

im = io.imread(names[0])
p = plt.imshow(im)
```

2 Task: Read and display image

Use the function cv2.imread() to read an image. The image should be in the working directory, or a full path of image should be given.

Second argument is a flag which specifies the way image should be read.

cv2.IMREAD_COLOR: Loads a color image. Any transparency of image will be neglected. It is the default flag.

cv2.IMREAD_GRAYSCALE: Loads image in grayscale mode

cv2.IMREAD_UNCHANGED: Loads image as such including alpha channel

[9]:
```
import numpy as np
import cv2
from matplotlib import pyplot as plt
# Load an color image
img_color = cv2.imread('P1-images/NU.jpg', cv2.IMREAD_COLOR)
img_color = cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB)

plt.imshow(img_color)
plt.xticks([]), plt.yticks([]) # to hide tick values on X and Y axis
plt.show()
```

[11]:
```
# Load an color image in grayscale
img_gray = cv2.imread('P1-images/NU.jpg', cv2.IMREAD_GRAYSCALE)
plt.imshow(img_gray, cmap='gray')
plt.xticks([]), plt.yticks([]) # to hide tick values on X and Y axis
plt.show()
```


THE
WHISKY
SHOP
Warren
James