What Powers Instagram? Part II

Announcement

- Instructors new office hour: Tuesday Thursday half an hour after class or by appointment
- If you can not make it to TA office hours, please email the instructor or the TAs to make appointment to meet outside the TA office hour window

Current TA office hours (subject to change):

- Monday 11:30 am – 2:30 pm William (2 hours), CADE Lab5, WEB L210
- Tuesday 3:30 pm – 5:30 p.m Ross (2 hours), CADE Lab5, WEB L210
- Wednesday 1:00 p.m – 3:00 p.m Vikram (2 hours), MEB 3115
- Thursday 3:30 p.m – 5:30 pm Ross (2 hours), CADE Lab5, WEB L210
- Friday 1:00pm – 3:00 p.m. William (2 hours), CADE Lab5, WEB L210

What Powers Instagram?

Credit:
http://instagram-engineering.tumblr.com/post/13649370142/what-powers-instagram-hundreds-of-instances
Summary:
http://highscalability.com/blog/2012/4/9/the-instagram-architecture-facebook-bought-for-a-cool-billio.html

How did Instagram Succeed from both Business and Tech Perspectives?

http://engineering.instagram.com/

Core principles in Choosing a System

- Keep it very simple
- Don’t reinvent the wheel
- Go with proven and solid technologies when you can

Main idea: getting your architecture written with high-scalability!

Some Key Aspects in Engineering

- Operating Systems / Hosting
- Load balancing
- Application servers to handle request
- Data Storage
- Task Queues and Push notification
- Monitoring

Other Notions for Success (up for debate)

- History: from Burbn (check-in to location) to Instagram (post photos)
- Turning user habit into cash
- Public (profile) by default
- Speed
- Cross network posting
- Asymmetric follow model (not the same as facebook)
- The Hook Model: online game, social media, emails

http://www.nirandfar.com/2012/02/habits-are-new-viral-why-startups-must.html

http://www.adambreckler.com/4-ways-instagram-hacked-early-growth

http://www.nirandfar.com/2012/03/want-to-hook-your-users-drive-them-crazy.html

Image Processing with Python

Lecture modeled after:
http://www.scipy-lectures.org/advanced/image_processing/
Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Image Manipulation and Processing
Many options

- Using primarily NumPy and SciPy, with some matplotlib (today)
- NumPy, SciPy and matplotlib are all installed in the CADE labs!
- PIL and Pillow
- scikit-image
- OpenCV
- SimpleCV
- mahotas

SciPy

The SciPy library is one of the core packages that make up the SciPy stack.

It provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization.

http://www.scipy.org/

A **module** is a **Python** object with arbitrarily named attributes that you can bind and reference. Simply, a **module** is a file consisting of **Python** code. A **module** can define functions, classes and variables.

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

http://www.numpy.org/

Matplotlib

matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, scatterplots, etc, with just a few lines of code.

http://matplotlib.org/

Tools Needed

scipy.ndimage

- provides functions operating on n-dimensional NumPy arrays
- Image = 2D numerical array = NumPy array

numpy: basic array manipulation

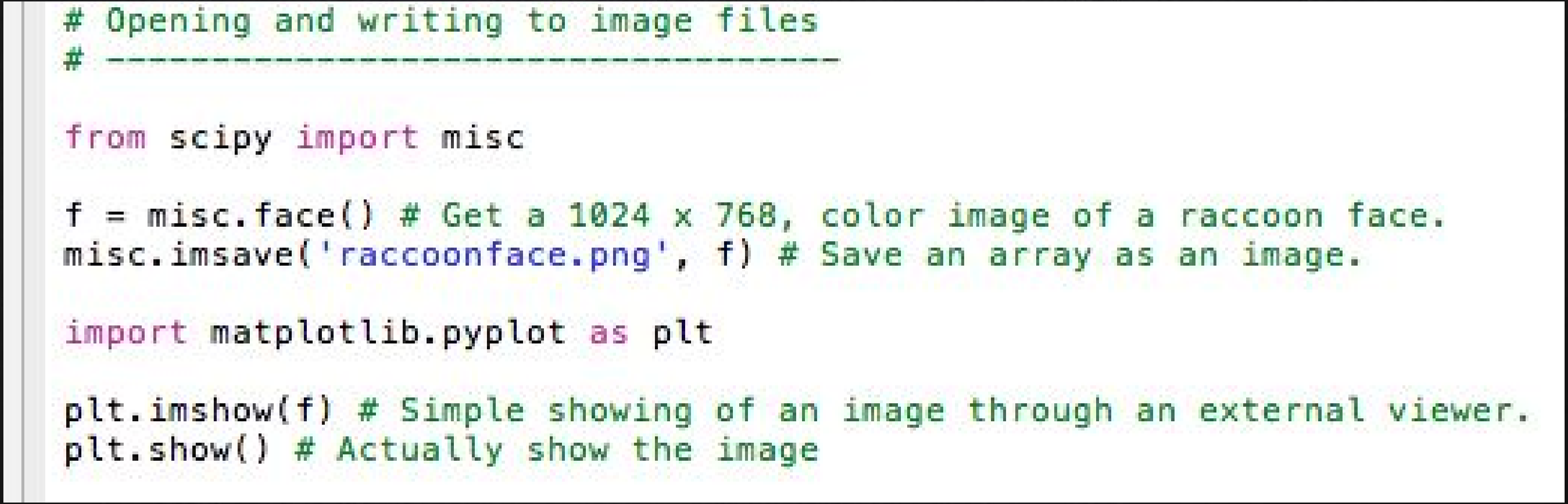
scipy: scipy.ndimage submodule, dedicated to image processing (n-dim images)

Opening and writing to image files

Display Images

```
$ python plot_open_raccoon.py
```

```
# Opening and writing to image files
from scipy import misc

f = misc.face() # Get a 1024 x 768, color image of a raccoon face.
misc.imsave('raccoonface.png', f) # Save an array as an image.

import matplotlib.pyplot as plt

plt.imshow(f) # Simple showing of an image through an external viewer.
plt.show() # Actually show the image
```

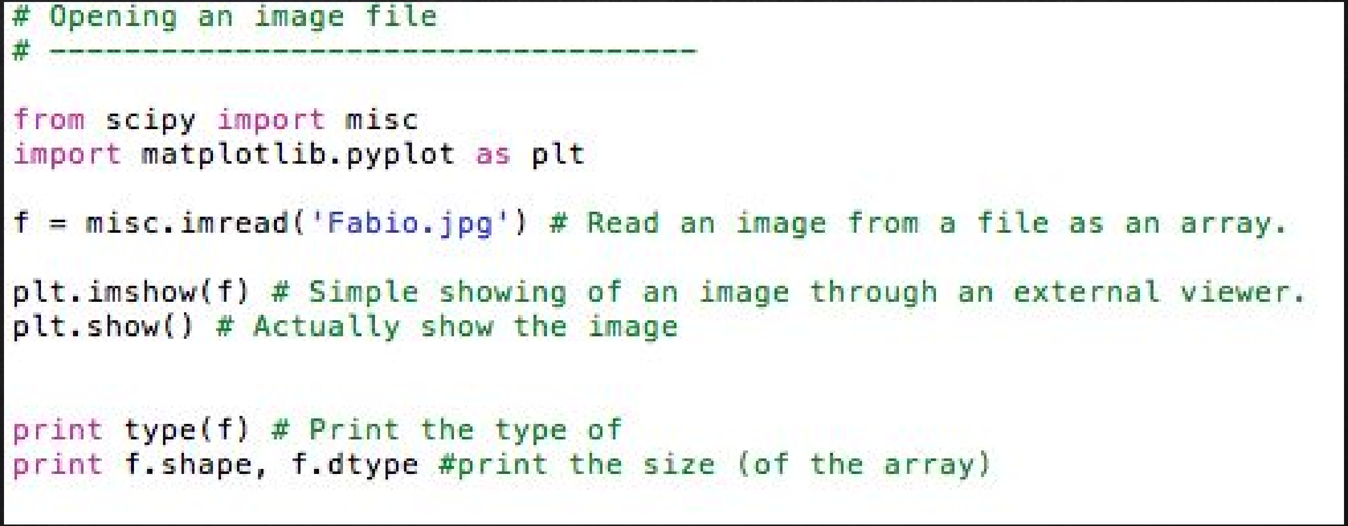


```
# Opening and writing to image files
# -----------------------------------

from scipy import misc

f = misc.face() # Get a 1024 x 768, color image of a raccoon face.
misc.imsave('raccoonface.png', f) # Save an array as an image.

import matplotlib.pyplot as plt

plt.imshow(f) # Simple showing of an image through an external viewer.
plt.show() # Actually show the image
```

Lecture modeled after:
http://www.scipy-lectures.org/advanced/image_processing/
Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

```
$ python plot_open_Fabio.py
```

```
# Opening an image file
from scipy import misc
import matplotlib.pyplot as plt

f = misc.imread('Fabio.jpg') # Read an image from a file as an array.

plt.imshow(f) # Simple showing of an image through an external viewer.
plt.show() # Actually show the image

print type(f) # Print the type of
print f.shape, f.dtype #print the size (of the array)
```



```
# Opening an image file
# -----------------------------------

from scipy import misc
import matplotlib.pyplot as plt

f = misc.imread('Fabio.jpg') # Read an image from a file as an array.

plt.imshow(f) # Simple showing of an image through an external viewer.
plt.show() # Actually show the image


print type(f) # Print the type of
print f.shape, f.dtype #print the size (of the array)
```

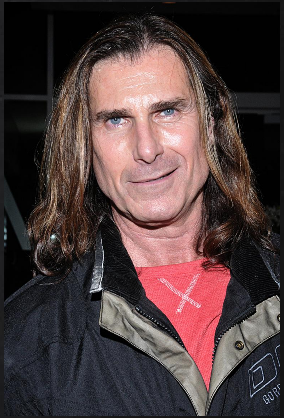
Lecture modeled after:
http://www.scipy-lectures.org/advanced/image_processing/
Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Fabio and Lena

Lena: a standard test image widely used in the field of image processing since 1973.

Lena Söderberg: by photographer Dwight Hooker, cropped from the centerfold of the November 1972 issue of *Playboy* magazine.

https://en.wikipedia.org/wiki/Lenna



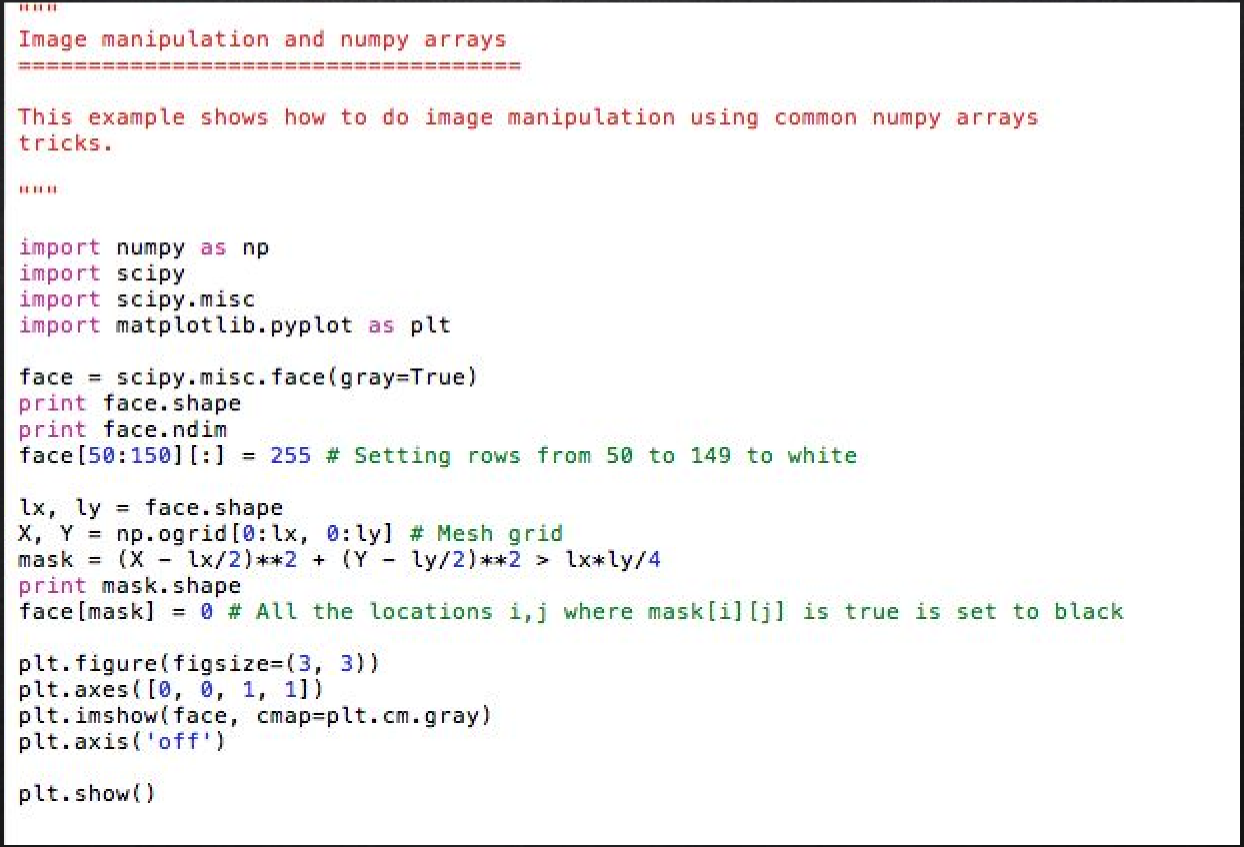
"Suggestive pictures used in lectures on image processing ... convey the message that the lecturer caters to the males only. For example, it is amazing that the "Lena" pin-up image is still used as an example in courses and published as a test image in journals today..." -- Dianne O'Leary

So you are going to see a lot of **Fabio Lanzoni** today…

Basic Manipulations

$ python plot_numpy_array.py

```
import numpy as np
import scipy
import scipy.misc
import matplotlib.pyplot as plt

face = scipy.misc.face(gray=True)
print face.shape
print face.ndim
face[50:150][:] = 255 # Setting rows from 50 to 149 to white

lx, ly = face.shape
X, Y = np.ogrid[0:lx, 0:ly] # Mesh grid
mask = (X – lx/2)**2 + (Y – ly/2)**2 > lx*ly/4
print mask.shape
face[mask] = 0 # All the locations i,j where mask[i][j] is true is set to black

plt.figure(figsize=(3, 3))
plt.axes([0, 0, 1, 1])
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')

plt.show()
```



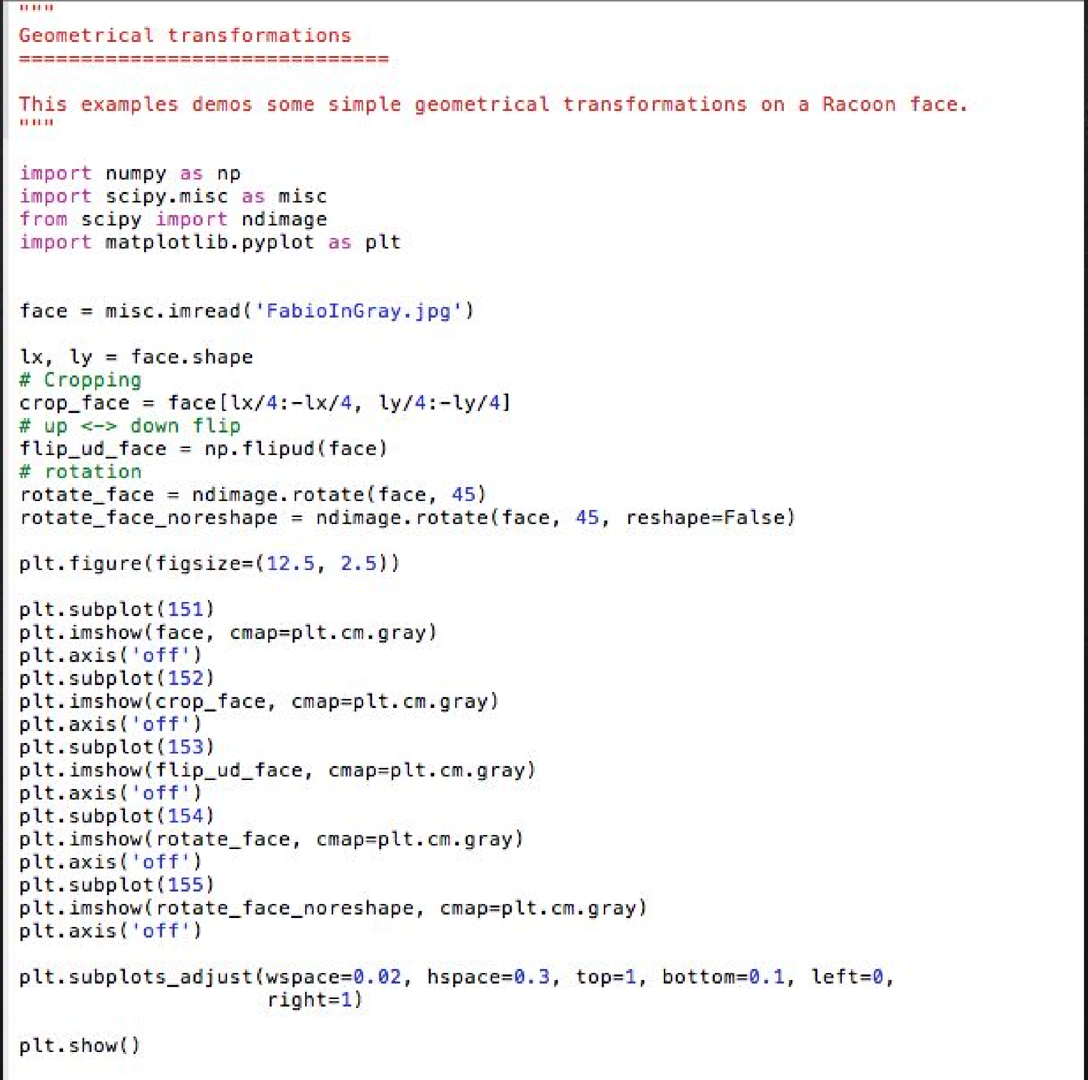
```
"""
Image manipulation and numpy arrays
====================================

This example shows how to do image manipulation using common numpy arrays
tricks.

"""

import numpy as np
import scipy
import scipy.misc
import matplotlib.pyplot as plt

face = scipy.misc.face(gray=True)
print face.shape
print face.ndim
face[50:150][:] = 255 # Setting rows from 50 to 149 to white

lx, ly = face.shape
X, Y = np.ogrid[0:lx, 0:ly] # Mesh grid
mask = (X - lx/2)**2 + (Y - ly/2)**2 > lx*ly/4
print mask.shape
face[mask] = 0 # All the locations i,j where mask[i][j] is true is set to black

plt.figure(figsize=(3, 3))
plt.axes([0, 0, 1, 1])
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')

plt.show()
```

Lecture modeled after:
http://www.scipy-lectures.org/advanced/image_processing/
Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Geometric Manipulations

$ python plot_geom_face.py

```
import numpy as np
import scipy.misc as misc
from scipy import ndimage
import matplotlib.pyplot as plt
face = misc.imread('FabioInGray.jpg')
lx, ly = face.shape
# Cropping
crop_face = face[lx/4:-lx/4, ly/4:-ly/4]
# up <-> down flip
flip_ud_face = np.flipud(face)
# rotation
rotate_face = ndimage.rotate(face, 45)
rotate_face_noreshape = ndimage.rotate(face, 45, reshape=False)
plt.figure(figsize=(12.5, 2.5))
plt.subplot(151)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(152)
plt.imshow(crop_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(153)
plt.imshow(flip_ud_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(154)
plt.imshow(rotate_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(155)
plt.imshow(rotate_face_noreshape, cmap=plt.cm.gray)
plt.axis('off')
plt.subplots_adjust(wspace=0.02, hspace=0.3, top=1, bottom=0.1, left=0,
            right=1)
plt.show()
```



```
"""
Geometrical transformations
==============================

This examples demos some simple geometrical transformations on a Racoon face.
"""

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


face = misc.imread('FabioInGray.jpg')

lx, ly = face.shape
# Cropping
crop_face = face[lx/4:-lx/4, ly/4:-ly/4]
# up <-> down flip
flip_ud_face = np.flipud(face)
# rotation
rotate_face = ndimage.rotate(face, 45)
rotate_face_noreshape = ndimage.rotate(face, 45, reshape=False)

plt.figure(figsize=(12.5, 2.5))

plt.subplot(151)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(152)
plt.imshow(crop_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(153)
plt.imshow(flip_ud_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(154)
plt.imshow(rotate_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(155)
plt.imshow(rotate_face_noreshape, cmap=plt.cm.gray)
plt.axis('off')

plt.subplots_adjust(wspace=0.02, hspace=0.3, top=1, bottom=0.1, left=0,
                    right=1)

plt.show()
```

Lecture modeled after: http://www.scipy-lectures.org/advanced/image_processing/

Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Image Filtering

Blurring using Gaussian Filter

Gaussian Filter

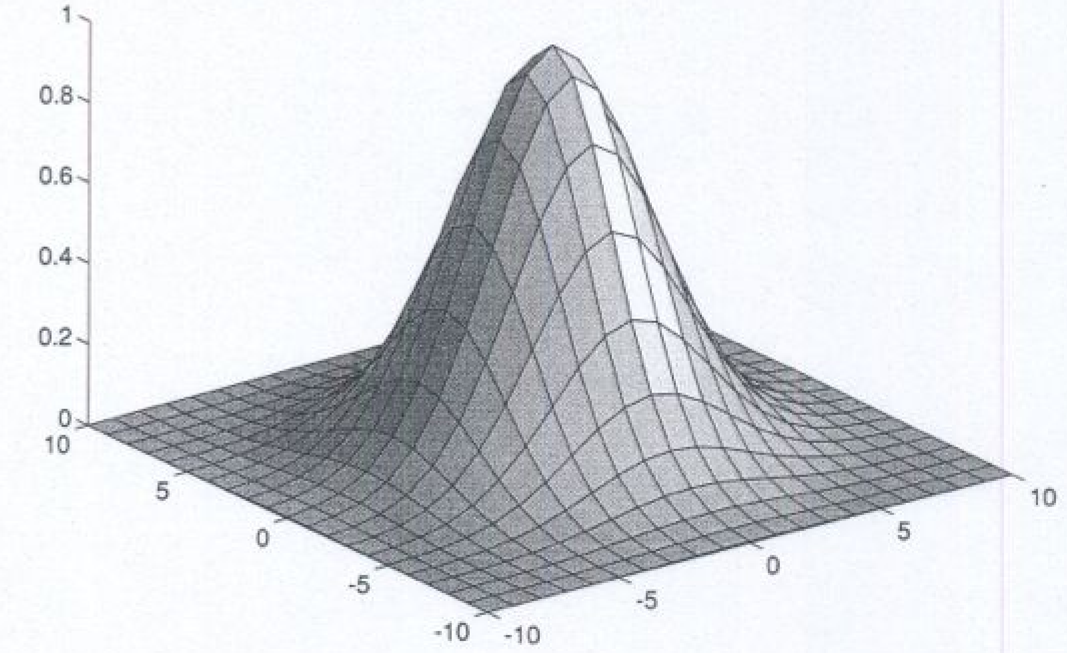


Figure 4.11: The two-dimensional Gaussian function with zero mean.

http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter4.pdf

Discrete Gaussian Filter

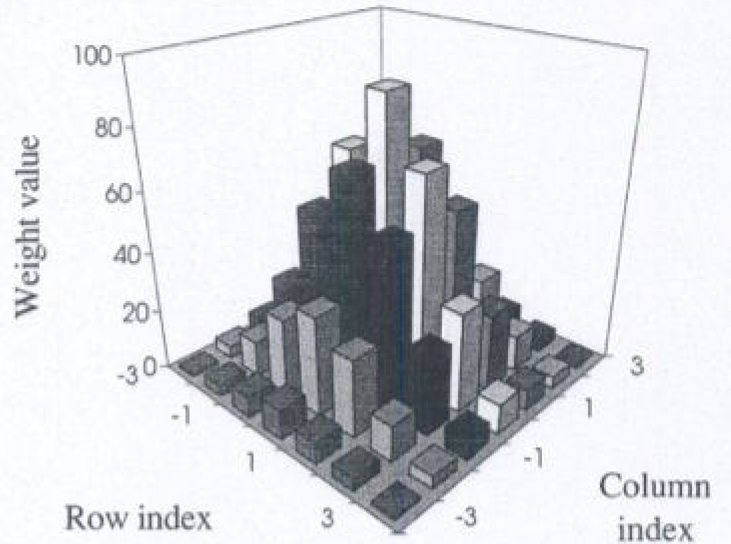
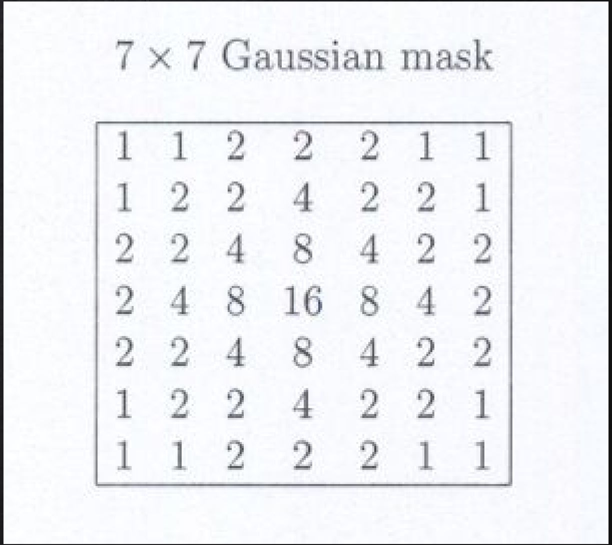


Figure 4.15: A 3-D plot of the 7 × 7 Gaussian mask.



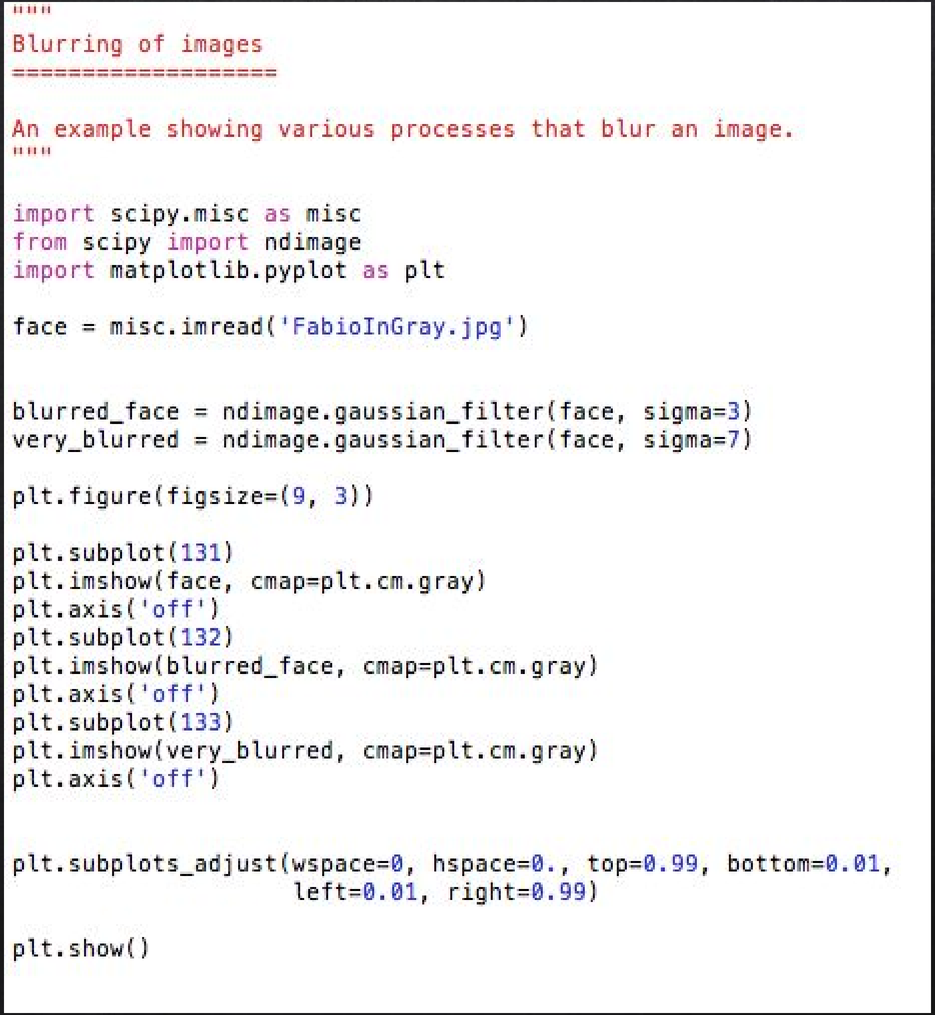
7 × 7 Gaussian mask

1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter4.pdf

$ python plot_blur.py

```
import scipy.misc as misc
from scipy import ndimage
import matplotlib.pyplot as plt

face = misc.imread('FabioInGray.jpg')


blurred_face = ndimage.gaussian_filter(face, sigma=3)
very_blurred = ndimage.gaussian_filter(face, sigma=7)

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(132)
plt.imshow(blurred_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(133)
plt.imshow(very_blurred, cmap=plt.cm.gray)
plt.axis('off')


plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                    left=0.01, right=0.99)

plt.show()
```



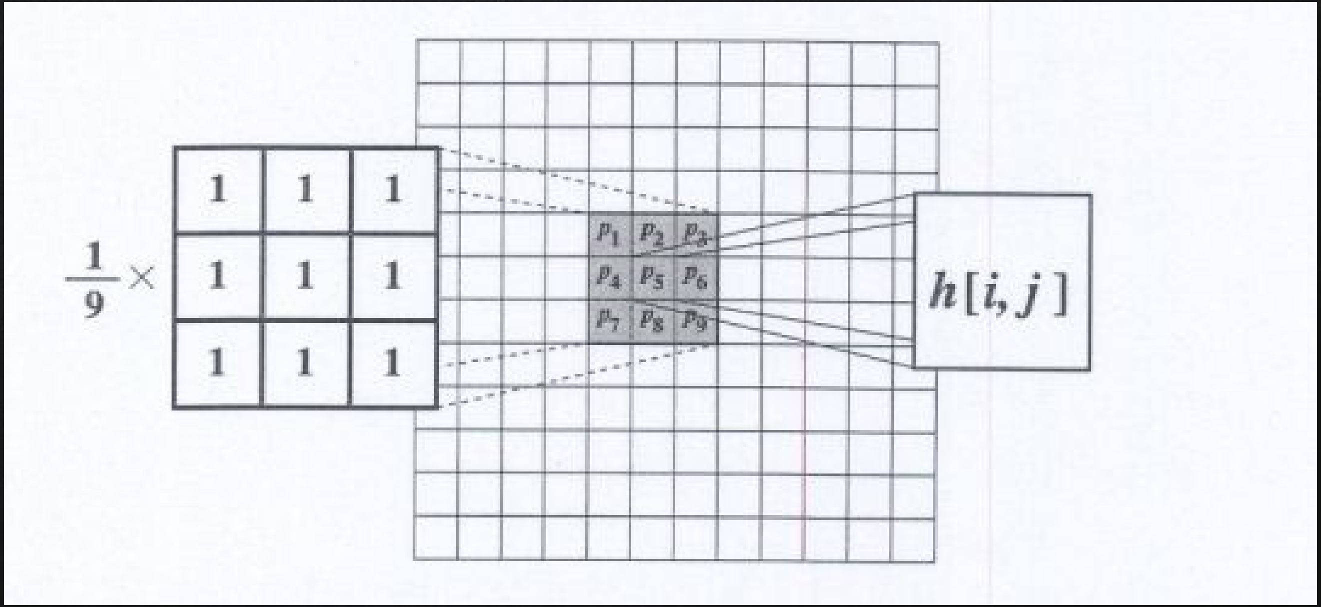
```
"""
Blurring of images
===================

An example showing various processes that blur an image.
"""

import scipy.misc as misc
from scipy import ndimage
import matplotlib.pyplot as plt

face = misc.imread('FabioInGray.jpg')


blurred_face = ndimage.gaussian_filter(face, sigma=3)
very_blurred = ndimage.gaussian_filter(face, sigma=7)

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(132)
plt.imshow(blurred_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(133)
plt.imshow(very_blurred, cmap=plt.cm.gray)
plt.axis('off')


plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                    left=0.01, right=0.99)

plt.show()
```

Lecture modeled after: http://www.scipy-lectures.org/advanced/image_processing/

Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Blurring using Uniform/Mean Filter

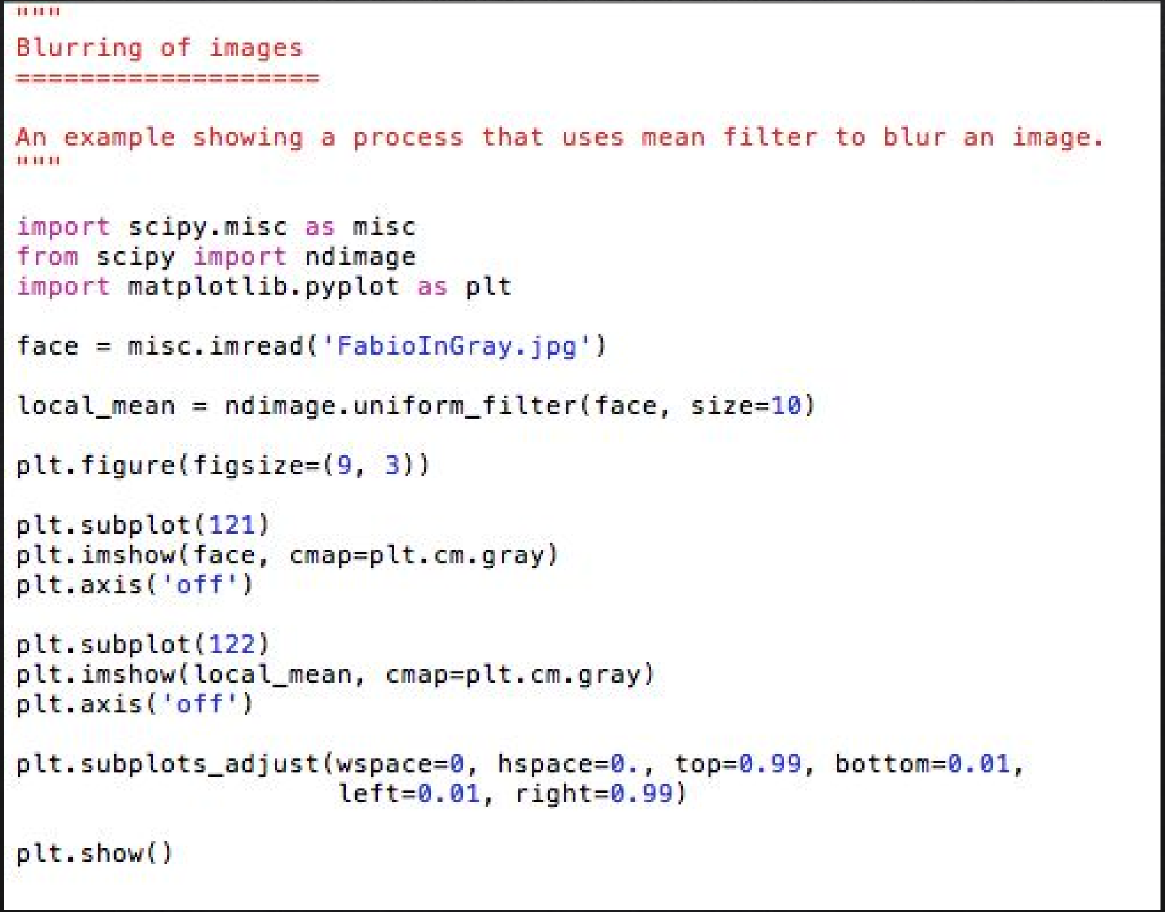
Mean Filter



http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter4.pdf

$ python plot_meanfilter.py

```
import scipy.misc as misc
from scipy import ndimage
import matplotlib.pyplot as plt

face = misc.imread('FabioInGray.jpg')

local_mean = ndimage.uniform_filter(face, size=10)

plt.figure(figsize=(9, 3))

plt.subplot(121)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(122)
plt.imshow(local_mean, cmap=plt.cm.gray)
plt.axis('off')

plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                left=0.01, right=0.99)

plt.show()
```



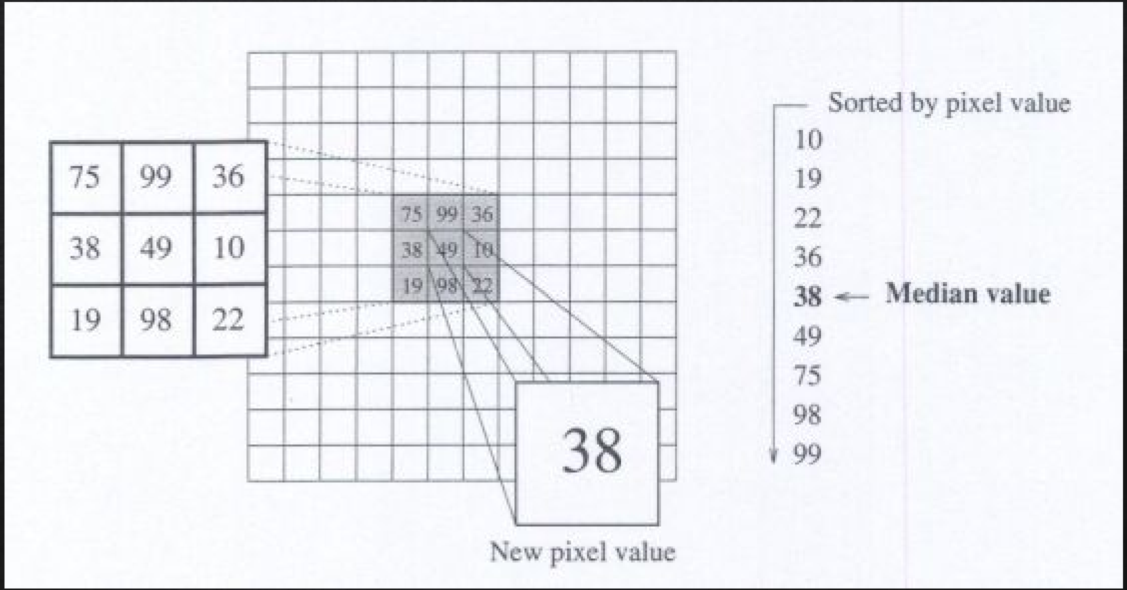
```
"""
Blurring of images
===================

An example showing a process that uses mean filter to blur an image.
"""

import scipy.misc as misc
from scipy import ndimage
import matplotlib.pyplot as plt

face = misc.imread('FabioInGray.jpg')

local_mean = ndimage.uniform_filter(face, size=10)

plt.figure(figsize=(9, 3))

plt.subplot(121)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(122)
plt.imshow(local_mean, cmap=plt.cm.gray)
plt.axis('off')

plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                    left=0.01, right=0.99)

plt.show()
```

Lecture modeled after: http://www.scipy-lectures.org/advanced/image_processing/

Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Denoising using Median Filter

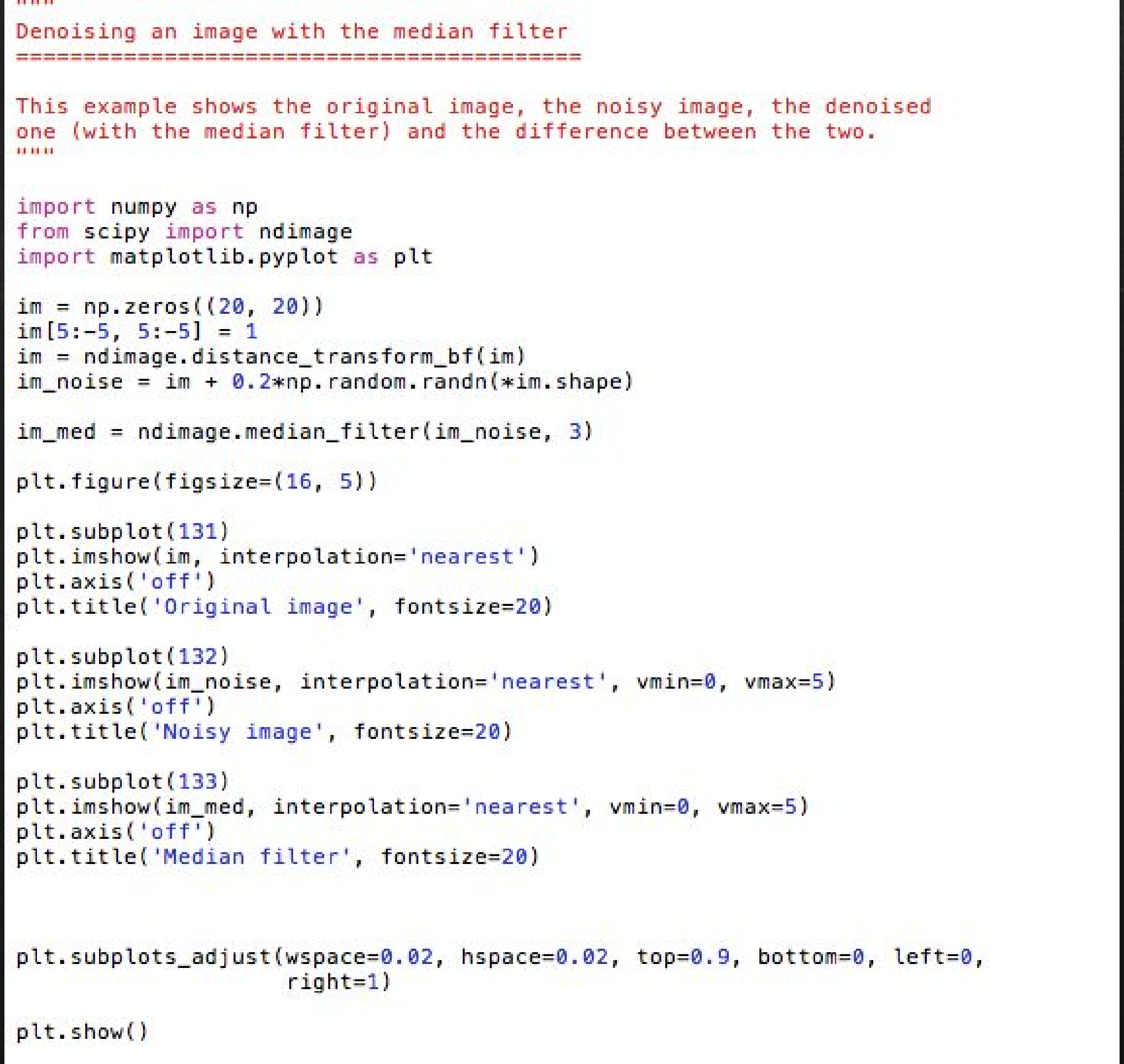
Median Filter



http://www.cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter4.pdf

$ python plot_denoising.py

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

im = np.zeros((20, 20))
im[5:-5, 5:-5] = 1
im = ndimage.distance_transform_bf(im)
im_noise = im + 0.2*np.random.randn(*im.shape)

im_med = ndimage.median_filter(im_noise, 3)

plt.figure(figsize=(16, 5))
plt.subplot(131)
plt.imshow(im, interpolation='nearest')
plt.axis('off')
plt.title('Original image', fontsize=20)

plt.subplot(132)
plt.imshow(im_noise, interpolation='nearest', vmin=0, vmax=5)
plt.axis('off')
plt.title('Noisy image', fontsize=20)

plt.subplot(133)
plt.imshow(im_med, interpolation='nearest', vmin=0, vmax=5)
plt.axis('off')
plt.title('Median filter', fontsize=20)
plt.subplots_adjust(wspace=0.02, hspace=0.02, top=0.9, bottom=0, left=0,
            right=1)

plt.show()
```



```
"""
Denoising an image with the median filter
=========================================

This example shows the original image, the noisy image, the denoised
one (with the median filter) and the difference between the two.
"""

import numpy as np
from scipy import ndimage
import matplotlib.pyplot as plt

im = np.zeros((20, 20))
im[5:-5, 5:-5] = 1
im = ndimage.distance_transform_bf(im)
im_noise = im + 0.2*np.random.randn(*im.shape)

im_med = ndimage.median_filter(im_noise, 3)

plt.figure(figsize=(16, 5))

plt.subplot(131)
plt.imshow(im, interpolation='nearest')
plt.axis('off')
plt.title('Original image', fontsize=20)

plt.subplot(132)
plt.imshow(im_noise, interpolation='nearest', vmin=0, vmax=5)
plt.axis('off')
plt.title('Noisy image', fontsize=20)

plt.subplot(133)
plt.imshow(im_med, interpolation='nearest', vmin=0, vmax=5)
plt.axis('off')
plt.title('Median filter', fontsize=20)



plt.subplots_adjust(wspace=0.02, hspace=0.02, top=0.9, bottom=0, left=0,
                    right=1)

plt.show()
```

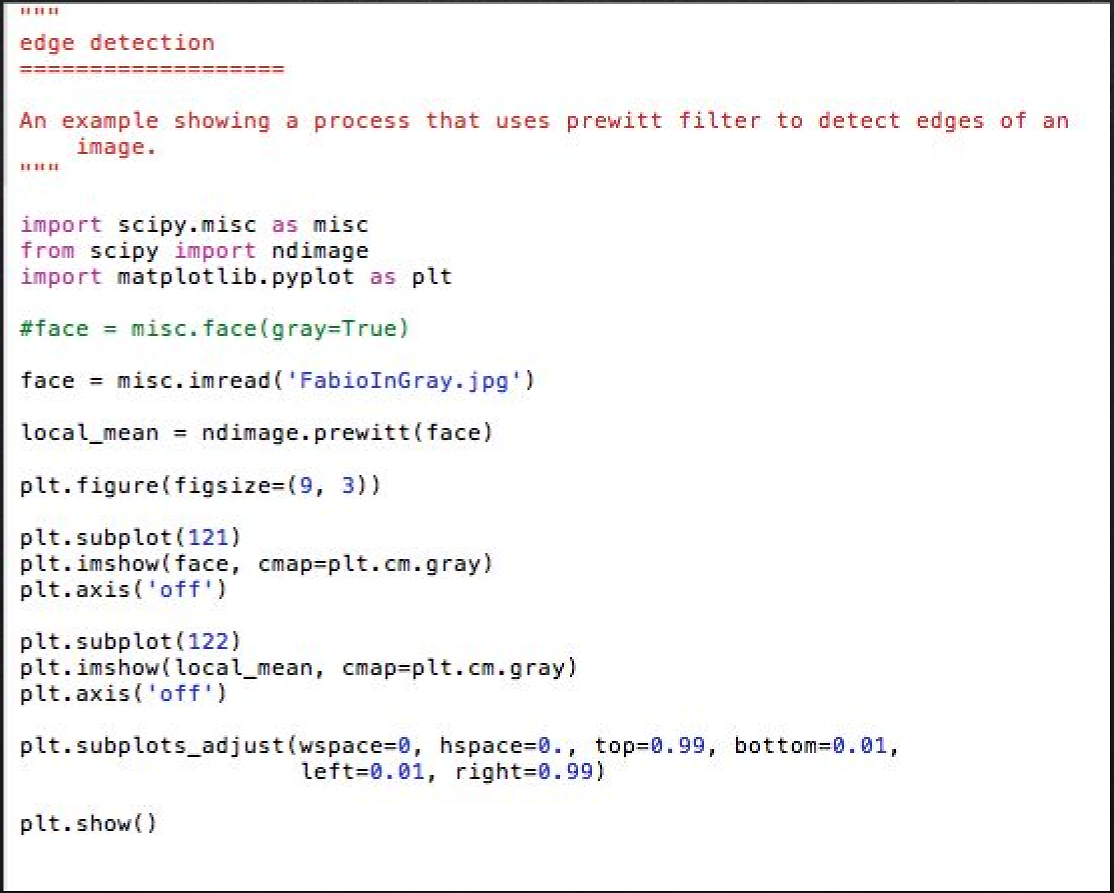
Lecture modeled after: http://www.scipy-lectures.org/advanced/image_processing/

Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Edge Detection Using Prewitt Filter

$ python plot_prewittfilter.py

```
import scipy.misc as misc
from scipy import ndimage
import matplotlib.pyplot as plt

#face = misc.face(gray=True)

face = misc.imread('FabioInGray.jpg')

local_mean = ndimage.prewitt(face)

plt.figure(figsize=(9, 3))

plt.subplot(121)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(122)
plt.imshow(local_mean, cmap=plt.cm.gray)
plt.axis('off')

plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
           left=0.01, right=0.99)

plt.show()
```

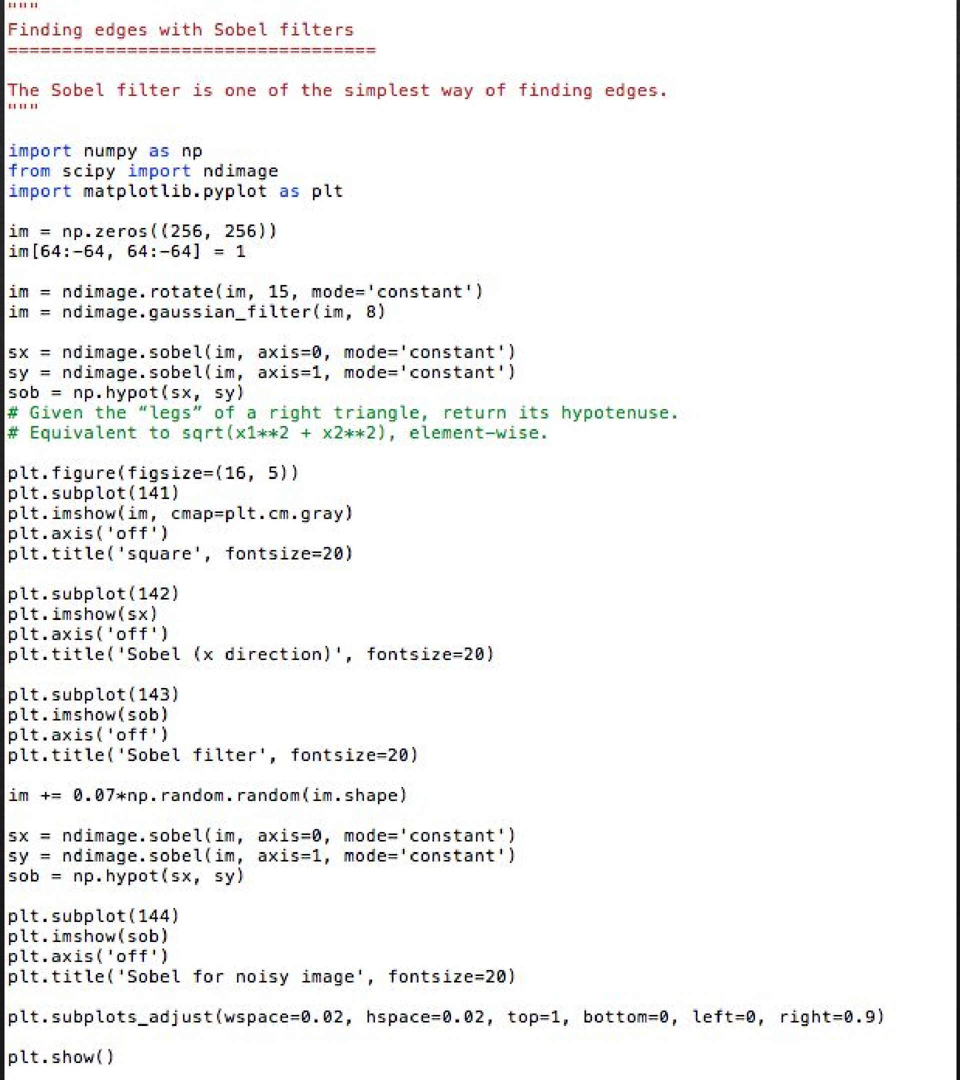


```
"""
edge detection
====================

An example showing a process that uses prewitt filter to detect edges of an
    image.
"""

import scipy.misc as misc
from scipy import ndimage
import matplotlib.pyplot as plt

#face = misc.face(gray=True)

face = misc.imread('FabioInGray.jpg')

local_mean = ndimage.prewitt(face)

plt.figure(figsize=(9, 3))

plt.subplot(121)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(122)
plt.imshow(local_mean, cmap=plt.cm.gray)
plt.axis('off')

plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                    left=0.01, right=0.99)

plt.show()
```

Lecture modeled after: http://www.scipy-lectures.org/advanced/image_processing/
Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Edge Detection Using Gradient operator (Sobel)

```
$ python plot_find_edges.py

# See Demo Screen
```



```
"""
Finding edges with Sobel filters
==================================

The Sobel filter is one of the simplest way of finding edges.
"""

import numpy as np
from scipy import ndimage
import matplotlib.pyplot as plt

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

im = ndimage.rotate(im, 15, mode='constant')
im = ndimage.gaussian_filter(im, 8)

sx = ndimage.sobel(im, axis=0, mode='constant')
sy = ndimage.sobel(im, axis=1, mode='constant')
sob = np.hypot(sx, sy)
# Given the “legs” of a right triangle, return its hypotenuse.
# Equivalent to sqrt(x1**2 + x2**2), element-wise.

plt.figure(figsize=(16, 5))
plt.subplot(141)
plt.imshow(im, cmap=plt.cm.gray)
plt.axis('off')
plt.title('square', fontsize=20)

plt.subplot(142)
plt.imshow(sx)
plt.axis('off')
plt.title('Sobel (x direction)', fontsize=20)

plt.subplot(143)
plt.imshow(sob)
plt.axis('off')
plt.title('Sobel filter', fontsize=20)

im += 0.07*np.random.random(im.shape)

sx = ndimage.sobel(im, axis=0, mode='constant')
sy = ndimage.sobel(im, axis=1, mode='constant')
sob = np.hypot(sx, sy)

plt.subplot(144)
plt.imshow(sob)
plt.axis('off')
plt.title('Sobel for noisy image', fontsize=20)

plt.subplots_adjust(wspace=0.02, hspace=0.02, top=1, bottom=0, left=0, right=0.9)

plt.show()
```

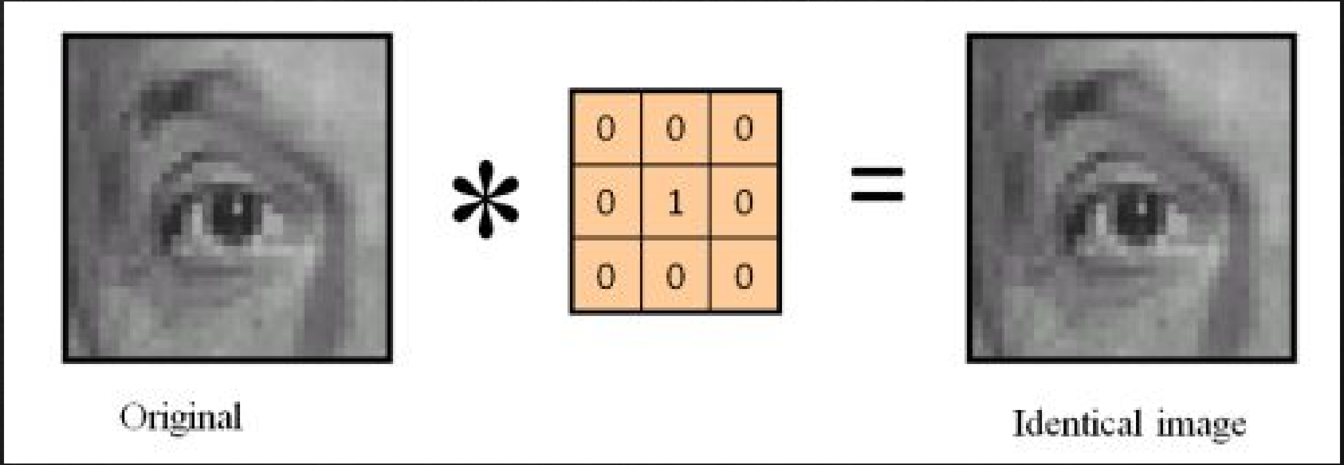
Lecture modeled after: http://www.scipy-lectures.org/advanced/image_processing/

Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

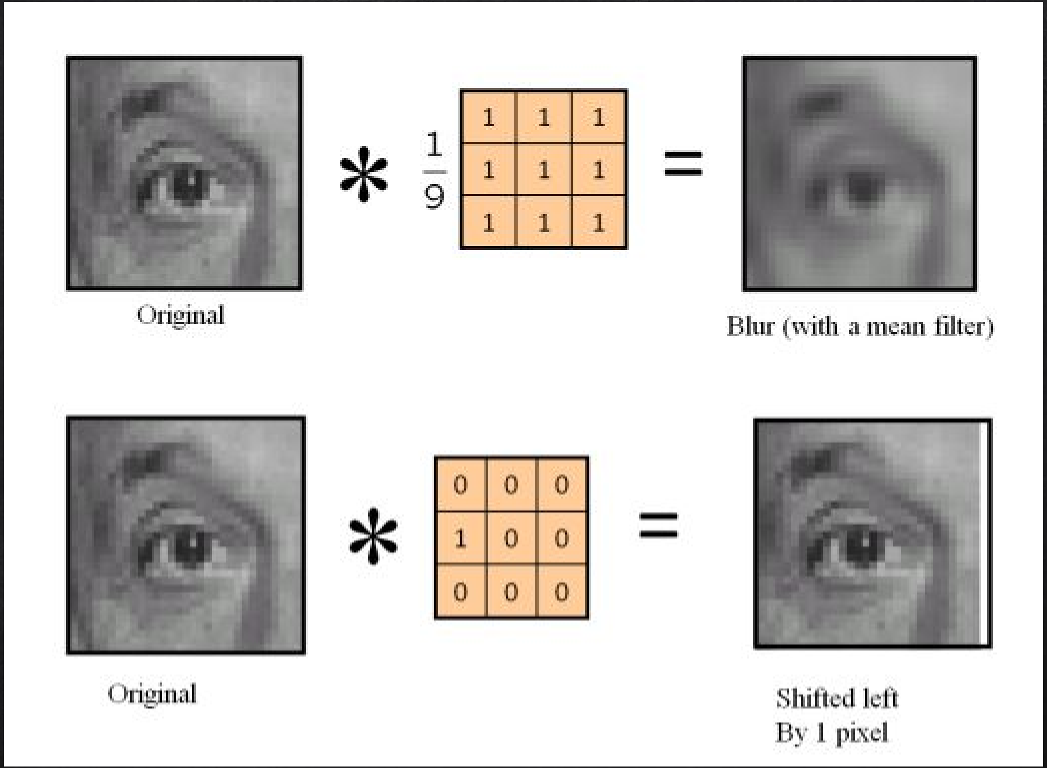
Convolution with a kernel

$$\begin{bmatrix} 10 & 5 & 3 \\ 4 & 5 & 1 \\ 1 & 1 & 7 \end{bmatrix} * \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 1.0 & 0.5 \end{bmatrix}}_{\text{kernel}} = \begin{bmatrix} & & \\ & 7 & \\ & & \end{bmatrix}$$

Credit: http://www.cs.utexas.edu/~grauman/courses/378/slides/lecture4.pdf

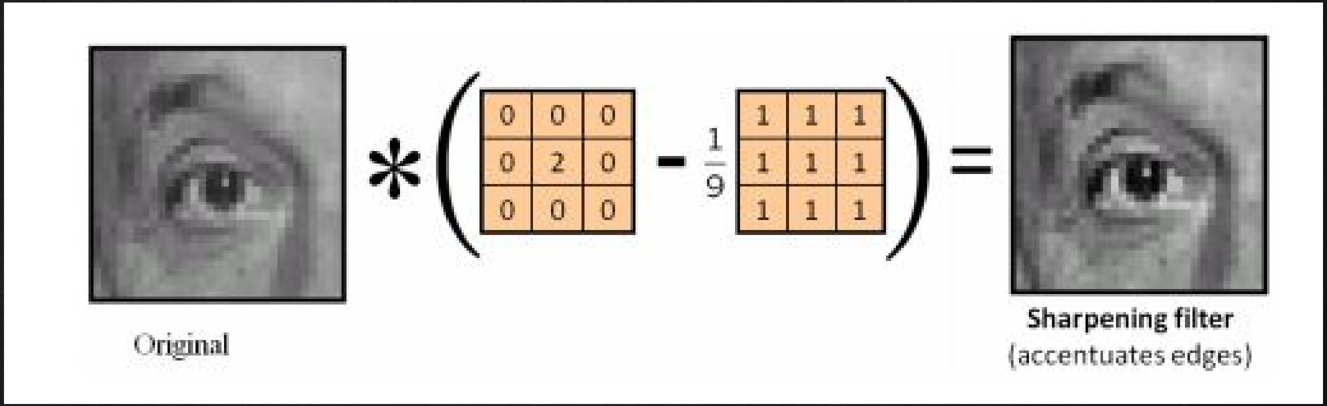


Credit:
http://www.cs.cornell.edu/courses/cs1114/2013sp/sections/s06_convolution.pdf



Original
$*\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} =$
Blur (with a mean filter)
Original
$*\ \begin{bmatrix}0&0&0\\1&0&0\\0&0&0\end{bmatrix} =$
Shifted left
By 1 pixel

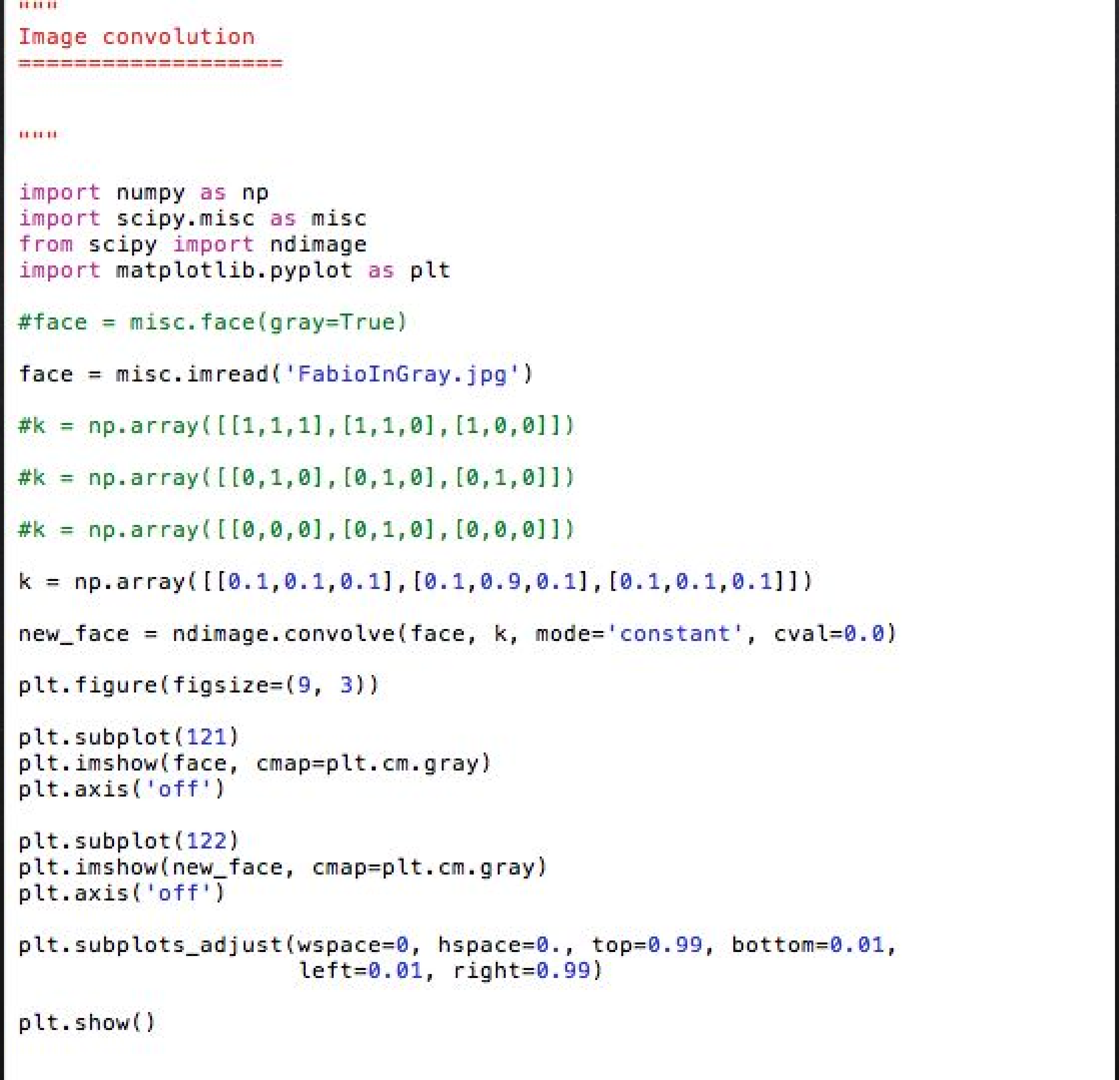
Credit:
http://www.cs.cornell.edu/courses/cs1114/2013sp/sections/s06_convolution.pdf



Credit:
http://www.cs.cornell.edu/courses/cs1114/2013sp/sections/s06_convolution.pdf

$ python plot_convolve.py

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

#face = misc.face(gray=True)

face = misc.imread('FabioInGray.jpg')

#k = np.array([[1,1,1],[1,1,0],[1,0,0]])

#k = np.array([[0,1,0],[0,1,0],[0,1,0]])

#k = np.array([[0,0,0],[0,1,0],[0,0,0]])

k = np.array([[0.1,0.1,0.1],[0.1,0.9,0.1],[0.1,0.1,0.1]])

new_face = ndimage.convolve(face, k, mode='constant', cval=0.0)

plt.figure(figsize=(9, 3))

plt.subplot(121)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(122)
plt.imshow(new_face, cmap=plt.cm.gray)
plt.axis('off')
plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                left=0.01, right=0.99)

plt.show()
```



```
"""
Image convolution
=====================


"""

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

#face = misc.face(gray=True)

face = misc.imread('FabioInGray.jpg')

#k = np.array([[1,1,1],[1,1,0],[1,0,0]])

#k = np.array([[0,1,0],[0,1,0],[0,1,0]])

#k = np.array([[0,0,0],[0,1,0],[0,0,0]])

k = np.array([[0.1,0.1,0.1],[0.1,0.9,0.1],[0.1,0.1,0.1]])

new_face = ndimage.convolve(face, k, mode='constant', cval=0.0)

plt.figure(figsize=(9, 3))

plt.subplot(121)
plt.imshow(face, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(122)
plt.imshow(new_face, cmap=plt.cm.gray)
plt.axis('off')

plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                    left=0.01, right=0.99)

plt.show()
```

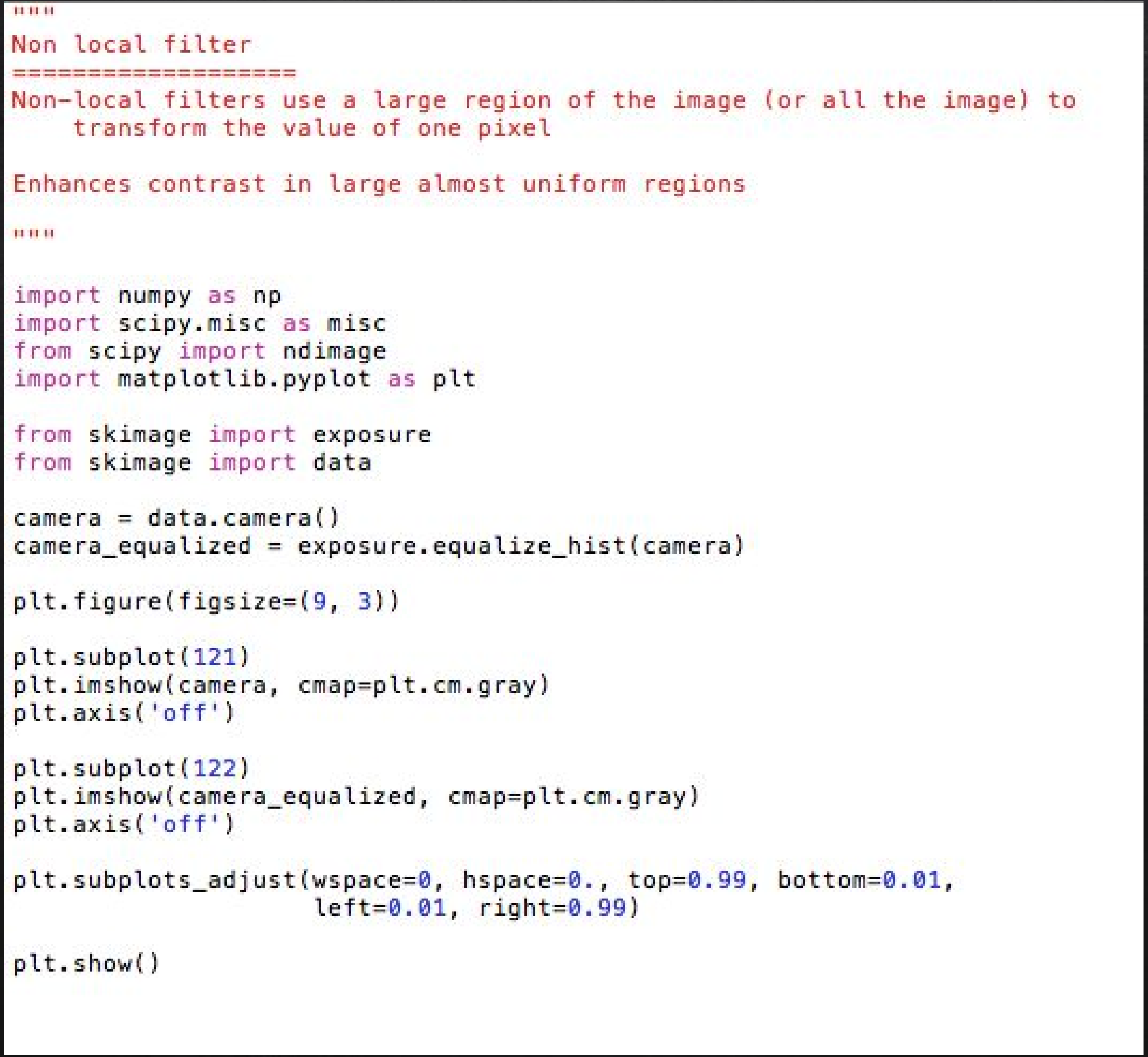
Lecture modeled after: http://www.scipy-lectures.org/advanced/image_processing/
Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Advanced functions with Scikit-Image

Credit: http://www.scipy-lectures.org/packages/scikit-image/index.html

$ python plot_camera.py

```python
"""
Non local filter
===================
Non-local filters use a large region of the image (or all the image) to transform the value of one pixel

Enhances contrast in large almost uniform regions
"""

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

from skimage import exposure
from skimage import data

camera = data.camera()
camera_equalized = exposure.equalize_hist(camera)

plt.figure(figsize=(9, 3))

plt.subplot(121)
plt.imshow(camera, cmap=plt.cm.gray)
plt.axis('off')
plt.subplot(122)
plt.imshow(camera_equalized, cmap=plt.cm.gray)
plt.axis('off')
plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                    left=0.01, right=0.99)
plt.show()
```



```
"""
Non local filter
====================
Non-local filters use a large region of the image (or all the image) to
    transform the value of one pixel

Enhances contrast in large almost uniform regions

"""

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

from skimage import exposure
from skimage import data

camera = data.camera()
camera_equalized = exposure.equalize_hist(camera)

plt.figure(figsize=(9, 3))

plt.subplot(121)
plt.imshow(camera, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(122)
plt.imshow(camera_equalized, cmap=plt.cm.gray)
plt.axis('off')

plt.subplots_adjust(wspace=0, hspace=0., top=0.99, bottom=0.01,
                    left=0.01, right=0.99)

plt.show()
```

Lecture modeled after: http://www.scipy-lectures.org/packages/scikit-image/index.html
Authors: *Emmanuelle Gouillart, Gaël Varoquaux*

Bonus Project

Write a 1 page essay (1-inch margin, 12 points font), detailing why you think Instagram has succeeded as a company, from both technology and business standpoint.

Please state at least 2 reasons each, for both tech and business perspectives, that Instagram is able to succeed.

Bonus points will constitute 3 points (of 100 points) in your final grade.

Points are awarded based on the quality of writing.

Do lots of online research, for example:

https://en.wikipedia.org/wiki/Instagram

http://www.inc.com/eric-markowitz/life-and-times-of-instagram-the-complete-original-story.html

THANKS!

Any questions?

You can find me at
beiwang@sci.utah.edu

http://www.sci.utah.edu/~beiwang/teaching/cs1060.html

Credits