basic

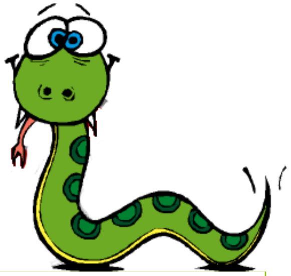
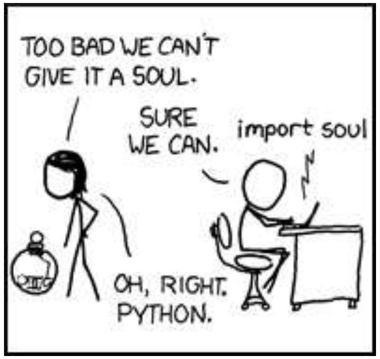


Image Processing with SciPy and NumPy

Introduction to Python

Foreword

- In the end, you should know how to…
  - … read text and MATLAB files
  - … load images in different formats (jpg, png, …)
  - … see images with Matplotlib
  - … apply filters via convolution



Loading data

- From a text file

```
#Col1    Col2     Col3    Col4
11.81  21.55  59.32  62.88
69.93  67.5   80.45  95.22
              …
41.77  0.66   90.78  67.29
```

```python
import numpy as np

data = np.loadtxt('datafile.txt')
```

Loading data

- From a MATLAB file

```
import scipy.io as io
matdata = io.loadmat('filename.mat')

print matdata['yourVariable']
```

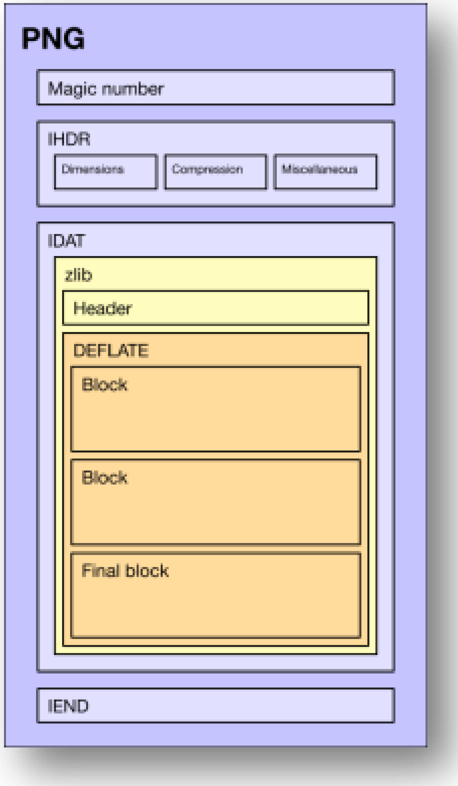
- From a binary file

```
import numpy as np

data = np.fromfile('filename.bin', dtype = int)
```

Images

- Are represented as **W×H×3** (RGB) or **W×H** (B&W) arrays
- Can we load it as a binary file?
  - **No!** Different formats exist: e.g., PNG, JPG, GIF, TIFF, …
- Python Imaging Library (PIL)
  - understands common formats
  - is used by SciPy



Loading images

- Using SciPy + Python Imaging Library:

```python
# image functions from SciPy
import scipy.ndimage as ndi

img = ndi.imread('python.png')
```

- To load it as grayscale,

```python
img = ndi.imread('python.png ', flatten = True)
```

Showing images

- Using Matplotlib:

```
import pylab as pl
import scipy.ndimage as ndi

img = ndi.imread('python.png')

pl.imshow( img )
pl.show()
```

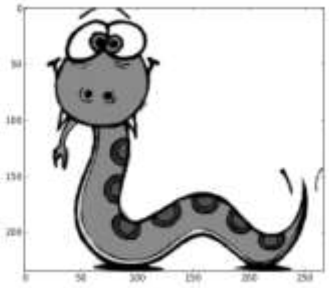
If W×H×3, `imshow` assumes image has colors

- If grayscale,

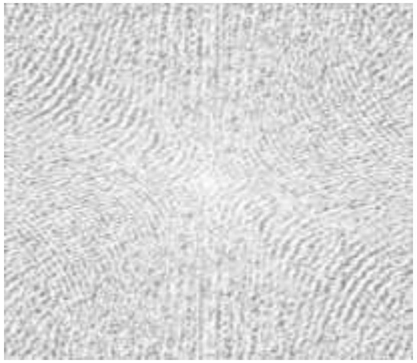
```
                …
pl.imshow( img, cmap = pl.cm.gray )
pl.show()
```

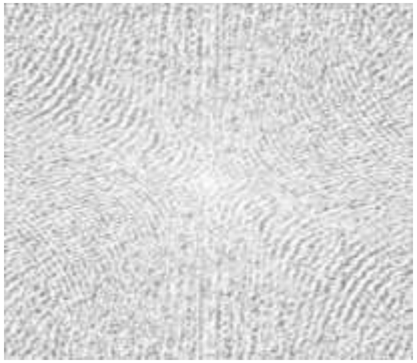
2D Fast Fourier Transform (FFT 2D)

- Transforms from space to frequency domain



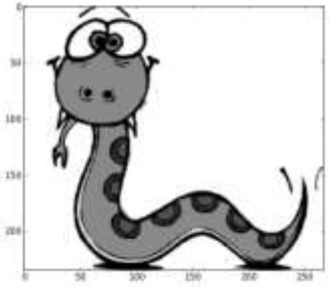
FFT 2D





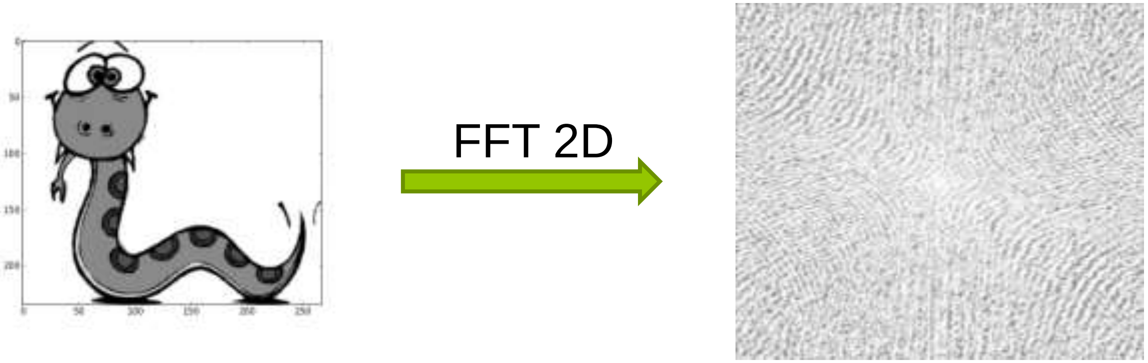
IFFT 2D

(inverse)



2D Fast Fourier Transform (FFT 2D)

- In Python

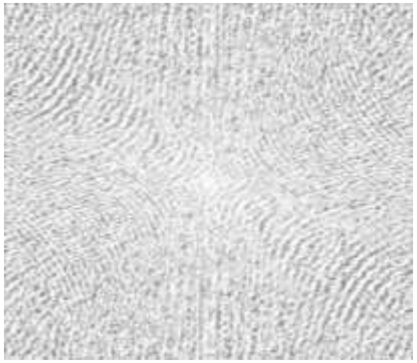


```
#...
from numpy.fft import fft2, fftshift
from numpy import real, log
#...
imgFFT = fftshift( fft2( img ) )
```

2D Fast Fourier Transform (FFT 2D)

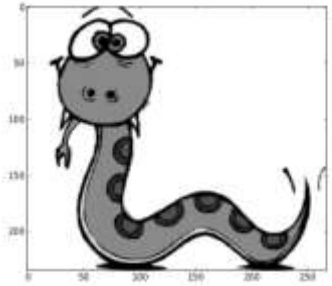
- In Python

```
#...
from numpy.fft import ifft2, ifftshift
from numpy import real, log
#...
img = ifft2( ifftshift( imgFFT ) )
```



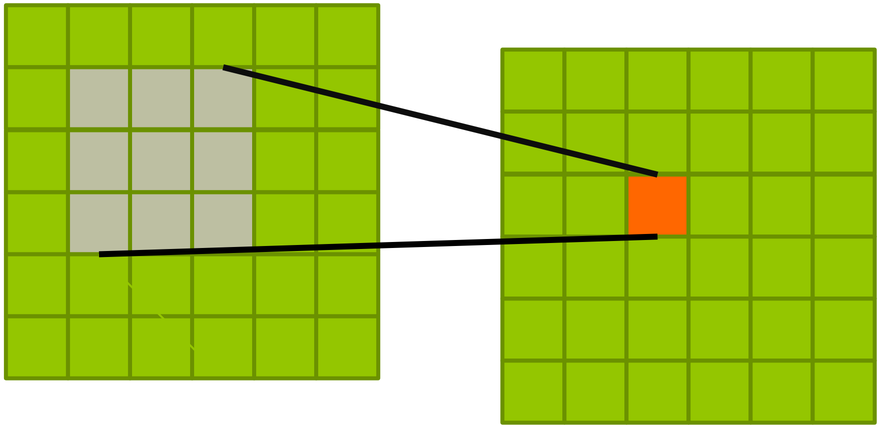
IFFT 2D

(inverse)



Filtering

- A low-pass filter (average over neighbors):

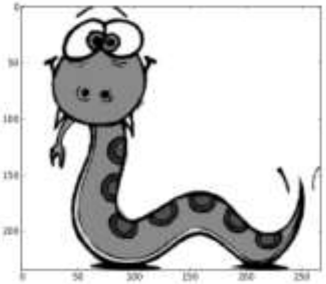


$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

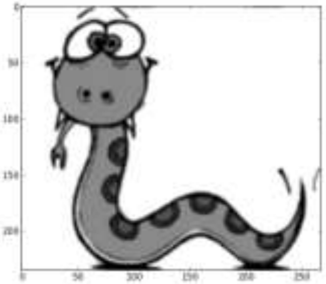
matrix
representation

Filtering (convolution)

- A low-pass filter (average over neighbors):

IFFT [FFT (

) × FFT ($\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$)]

= 

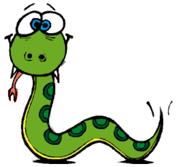
Filtering (convolution)

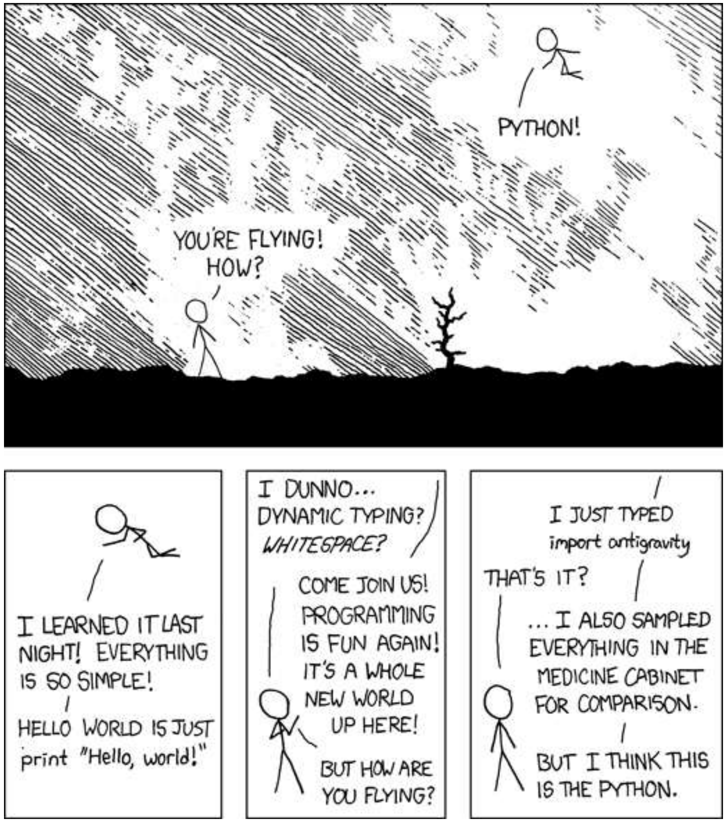
- In Python

```
#...
from numpy.fft import fft2, ifft2
from numpy import real, multiply, ones
#...
kern = 1./9. * ones([3,3])

finalImageFFT = multiply( fft2(img), fft2(kern, img.shape) )
finalImage = real( ifft2( finalImageFFT ) )
```

And now to the exercises…!




PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Also useful:

scikits-image
image processing in python

http://scikits-image.org/