Lecture 8

Scope
Image Processing

Prof. Steven Ludtke
N410.07, sludtke@bcm.edu

Scope

What will this produce ?

```
def f(x):
    y=x*10
    return y

x=5
y=6

print y
print f(x)
print y
```

Scope

How about this ?

```
def f():
    y=x*10
    return y

x=5
y=6

print y
print f()
print y
```

Scope

- Local scope
  - Variables defined within a function, exist only within the function
  - Variables defined in a class definition become class variables
- Global scope
  - Variables defined at the “top level” in the program or module.
  - Variables declared using the “global” keyword
- Built-in names
  - Built in functions and variable names

Scope

What will this produce ?

```
def f(x):
    global y
    y=x*10
    return y

x=5
y=6

print y
print f(x)
print y
```

Scope

What will this produce ?

```
class abc:
  xyz=10
  def __init__(self) : self.qqq=5

  def f(self):
    # Which of the next 3 lines is correct?
    print(self.xyz)
    print(abc.xyz)
    print(xyz)

x=abc()
y=abc()
x.f()
x.xyz=15
print(x.xyz,y.xyz,abc.xyz)   # What will this give?
```

General Image Processing

- PIL/PILLOW (today)
- SciPy 'ndimage' module
  - Apply NumPy capabilities to image processing
- OpenCV
  - Computer vision library with Python bindings
- Mahotas
  - Another computer vision library
- EMAN2
  - Greyscale quantitative image processing
  - Links to structural biology (CyroEM)
- GIMP
  - Free Photoshop-like, cross-platform
  - Python based 'modules'

PIL/PILLOW

- Python Imaging Library
- Open-source/commercial
- PIL unofficially dead, PILLOW is the replacement
- Reads-writes many standard formats
- Generic image processing

Installing PIL

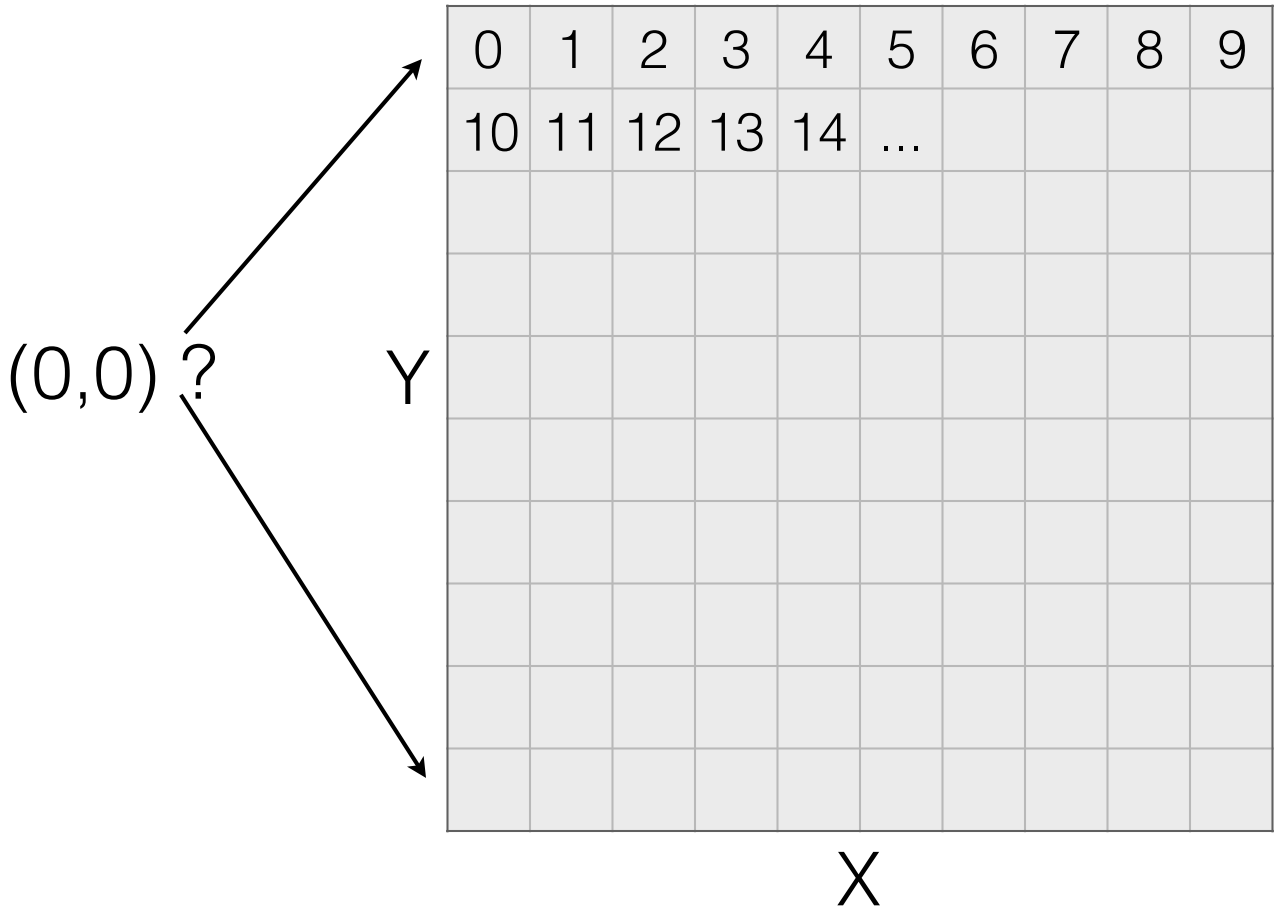
- (SHOULD ALREADY BE INSTALLED WITH ANACONDA)

- https://pypi.python.org/pypi/Pillow
- Windows:
  - Standard installer packages should work
- Linux
  - Should be in package installer (may need to look for python imaging)
- Mac
  - Make sure you have Xcode & command-line tools installed
  - Install libjpeg (http://www.ijg.org/)
    - http://snippets.dzone.com/posts/show/38
  - sudo easy_install pillow
  - If this doesn't work, you may want to try the instructions on either the pillow or the pil website.

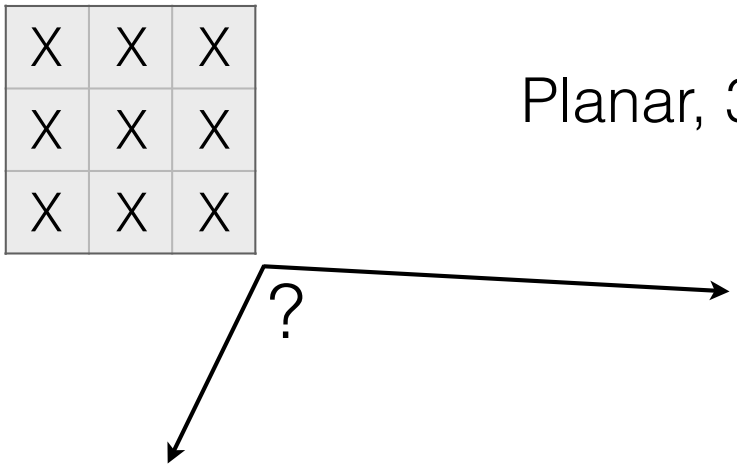
- to test: 'import PIL'

Images

Pixel stored at location x+nx*y (row major)
or y+ny*x (column major, less common)



Color Images



Planar, 3x3 image, row-major

R	R	R
R	R	R
R	R	R
G	G	G
G	G	G
G	G	G
B	B	B
B	B	B
B	B	B

Interleaved, 3x3 image, row-major

R	G	B	R	G	B	R	G	B
R	G	B	R	G	B	R	G	B
R	G	B	R	G	B	R	G	B

PIL - File Formats*

Fmt	Bits	Loss	Cmpr	Notes
BMP	8 or less			
GIF	8 total, cmap	X	X	
IM	all modes !			LabEye & IFUNC
JPEG	8	X	X	
PCX	8 or less			
PNG	8		X	
PPM	8			PBM,PGM,PPM
TIFF	8	R	R	
EPS				Needs GS to read most
PDF				Write only

* - Only the most common ones

Image Modes

- 1 (1-bit pixels, black and white, stored with one pixel per byte)
- L (8-bit pixels, black and white)
- P (8-bit pixels, mapped to any other mode using a colour palette)
- RGB (3x8-bit pixels, true colour)
- RGBA (4x8-bit pixels, true colour with transparency mask)
- RGBX (3x8-bit pixels, true colour with padding byte)
- CMYK (4x8-bit pixels, colour separation)
- YCbCr (3x8-bit pixels, colour video format)
- I (32-bit signed integer pixels
- F (32-bit floating point pixels)

PIL

```
from PIL import Image
im=Image.open("file.jpg")
data=b"\0"*(128*128*4)        # 'bytes' of zero pixels
from array import array       # note this is NOT numpy
data=array("b",data)          # convert to an array object
im=Image.frombuffer("RGBX",(128,128),data,"raw","RGBX",0,1)
im.show()                     # machine specific display
pix=im.load()                 # for pixel access
pix[x,y]                      # access pixel at x,y
im.save(filename,[format],[options])
```

Using Numpy

```
from numpy import *

from PIL import Image

a=fromfunction(lambda x,y:sin(x/10.0)*cos(y/10.0),(128,128))

im=Image.fromarray(a)

im.show()                # Image is black !?!?

a+=1

a*=127

im2=Image.fromarray(a)

im2.show()
```

PIL

```
im=Image.open("test.jpg")

a=array(im)

a[0,0]

im2=Image.fromarray(a)
```

PIL

```
a=fromfunction(lambda x,y:(127+sin(x/100.)*127., 127+cos(y/
100.)*127.,127+sin(x/500.)*127.),(256,256))

b=dstack(a)

c=b.astype(uint8)

im=Image.fromarray(c)

im.show()
```

PIL

- ImageChops - invert(a), lighter(a,b), darker(a,b), add(a,b), subtract(a,b), difference(a,b), screen(a,b)

- ImageDraw

```
from PIL import Image,ImageDraw

a=Image.new("RGBA",(128,128))

draw=ImageDraw.Draw(a)

draw.line((x0,y0,x1,y1),fill="red") # point, rectangle,
arc, chord, ellipse, text
```

- ImageFilter - BLUR, CONTOUR, DETAIL, EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS, FIND_EDGES, SMOOTH, SMOOTH_MORE, SHARPEN

- etc.

PIL Attributes

- im.format
- im.mode
- im.size
- im.info

Images in MatPlotLib

```
ipython -pylab

im=fromfunction(lambda x,y:sin(x/10.0)*cos(y/10.0),(64,64))

imshow(im)

imshow(im,cmap=cm.gray)

savefig("a.png")
```

mahotas

```
import numpy as np
import mahotas
import pylab

img = mahotas.imread('test.jpeg')
T_otsu = mahotas.thresholding.otsu(img)
seeds,_ = mahotas.label(img > T_otsu)
labeled = mahotas.cwatershed(img.max() - img, seeds)

pylab.imshow(labeled)
```

Homework 8

- Write a program that finds all of the .jpg files in the current directory, reads each one, performs some sort of image processing with PIL on each, and writes each back to disk with "_proc" in its name, ie - image.jpg -> image_proc.jpg
- tip - you may find the os library (lecture 4) useful