CLASS PROJECTS

Class projects must be submitted to me by email following the exact instructions on the class website, by 11:59pm on Saturday, Feb 22. Please read the instructions this week !

Class projects will be presented on
Monday, Feb 24 from **8am** - 10:30.

We will do all of the presentations in this one extended session. Much like a national meeting, you will have 1 minute to get your laptop working with the projector (I suggest testing it ahead of time), and 5 minutes to present. We may have time for a question or two.

No more homework
(perhaps another in-class lab)

Lecture 9

Complex Data Storage
PIL (Images)

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

How to Store Complex Data ?

- Students
  - Name, address, ...
- Classes
  - Description, Instructor, when offered, ...
- Class Year
  - Which class, year offered, students
- Grades
  - Class, student, grade

XML Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CATALOG>
   <PLANT>
      <COMMON>Bloodroot</COMMON>
      <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
      <ZONE>4</ZONE>
      <LIGHT>Mostly Shady</LIGHT>
      <PRICE CURRENCY=”dollar”>2.44</PRICE>
      <AVAILABILITY>031599</AVAILABILITY>
   </PLANT>
   <PLANT>
      <COMMON>Columbine</COMMON>
      <BOTANICAL>Aquilegia canadensis</BOTANICAL>
      <ZONE>3</ZONE>
      <LIGHT>Mostly Shady</LIGHT>
      <PRICE CURRENCY=”dollar” >9.37</PRICE>
      <AVAILABILITY>030699</AVAILABILITY>
   </PLANT>
</CATALOG>
```

Formatted Files

Genbank

```
  1 gatcctccat atacaacggt atctccacct caggtttaga tctcaacaac ggaaccattg
 61 ccgacatgag acagttaggt atcgtcgaga gttacaagct aaaacgagca gtagtcagct
121 ctgcatctga agccgctgaa gttctactaa gggtggataa catcatccgt gcaagaccaa
181 gaaccgccaa tagacaacat atgtaacata tttaggatat acctcgaaaa taataaaccg
241 ccacactgtc attattataa ttagaaacag aacgcaaaaa ttatccacta tataattcaa
301 agacgcgaaa aaaaaagaac aacgcgtcat agaacttttg gcaattcgcg tcacaaataa
361 attttggcaa cttatgtttc ctcttcgagc agtactcgag ccctgtctca agaatgtaat
421 aatacccatc gtaggtatgg ttaaagatag catctccaca acctcaaagc tccttgccga
481 gagtcgccct cctttgtcga gtaattttca cttttcatat gagaacttat tttcttattc
541 tttactctca catcctgtag tgattgacac tgcaacagcc accatcacta gaagaacaga
601 acaattactt aatagaaaaa ttatatcttc ctcgaaacga tttcctgctt ccaacatcta
661 cgtatatcaa gaagcattca cttaccatga cacagcttca gatttcatta ttgctgacag
```

Formatted Files

PDB

```
HETATM    1  C    FOR A   1A      -3.690  -1.575  -2.801  1.00  0.00           C
HETATM    2  O    FOR A   1A      -3.774  -1.363  -1.586  1.00  0.00           O
HETATM    3  H    FOR A   1A      -4.305  -1.047  -3.545  1.00  0.00           H
ATOM      4  N    VAL A   1       -3.043  -2.601  -3.416  1.00  0.00           N
ATOM      5  CA   VAL A   1       -2.415  -3.640  -2.601  1.00  0.00           C
ATOM      6  C    VAL A   1       -0.920  -3.596  -2.797  1.00  0.00           C
ATOM      7  O    VAL A   1       -0.385  -3.961  -3.851  1.00  0.00           O
ATOM      8  CB   VAL A   1       -2.919  -5.045  -2.884  1.00  0.00           C
ATOM      9  CG1  VAL A   1       -2.163  -6.141  -2.056  1.00  0.00           C
ATOM     10  CG2  VAL A   1       -4.427  -5.132  -2.644  1.00  0.00           C
ATOM     11  H    VAL A   1       -2.760  -2.649  -4.381  1.00  0.00           H
ATOM     12  HA   VAL A   1       -2.651  -3.401  -1.570  1.00  0.00           H
ATOM     13  HB   VAL A   1       -2.542  -5.494  -3.836  1.00  0.00           H
ATOM     14 1HG1  VAL A   1       -2.280  -5.946  -0.974  1.00  0.00           H
ATOM     15 2HG1  VAL A   1       -2.578  -7.146  -2.278  1.00  0.00           H
ATOM     16 3HG1  VAL A   1       -1.080  -6.174  -2.292  1.00  0.00           H
                                      ...
```

pickle

- 'Serialization' - converting a complex object to a stream of data

```
from cPickle import dump,load,dumps,loads

dump(obj,file[,protocol])   # stores ‘obj’ in file

obj=load(file)              # restores ‘obj’ from file

str=dumps(obj[,protocol])   # pickled representation of obj

obj=loads(str)              # restore representation of obj
```

shelve

```
import shelve      # dictionary-like object on disk
dct=shelve.open(filename[,protocol])
dct=shelve.open(filename,writeback=True)
dct.close()
```

General Image Processing

- PIL/PILLOW (today)
- SciPy 'ndimage' module
  - Apply NumPy capabilities to image processing
- OpenCV
  - Computer vision library with Python bindings
- 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

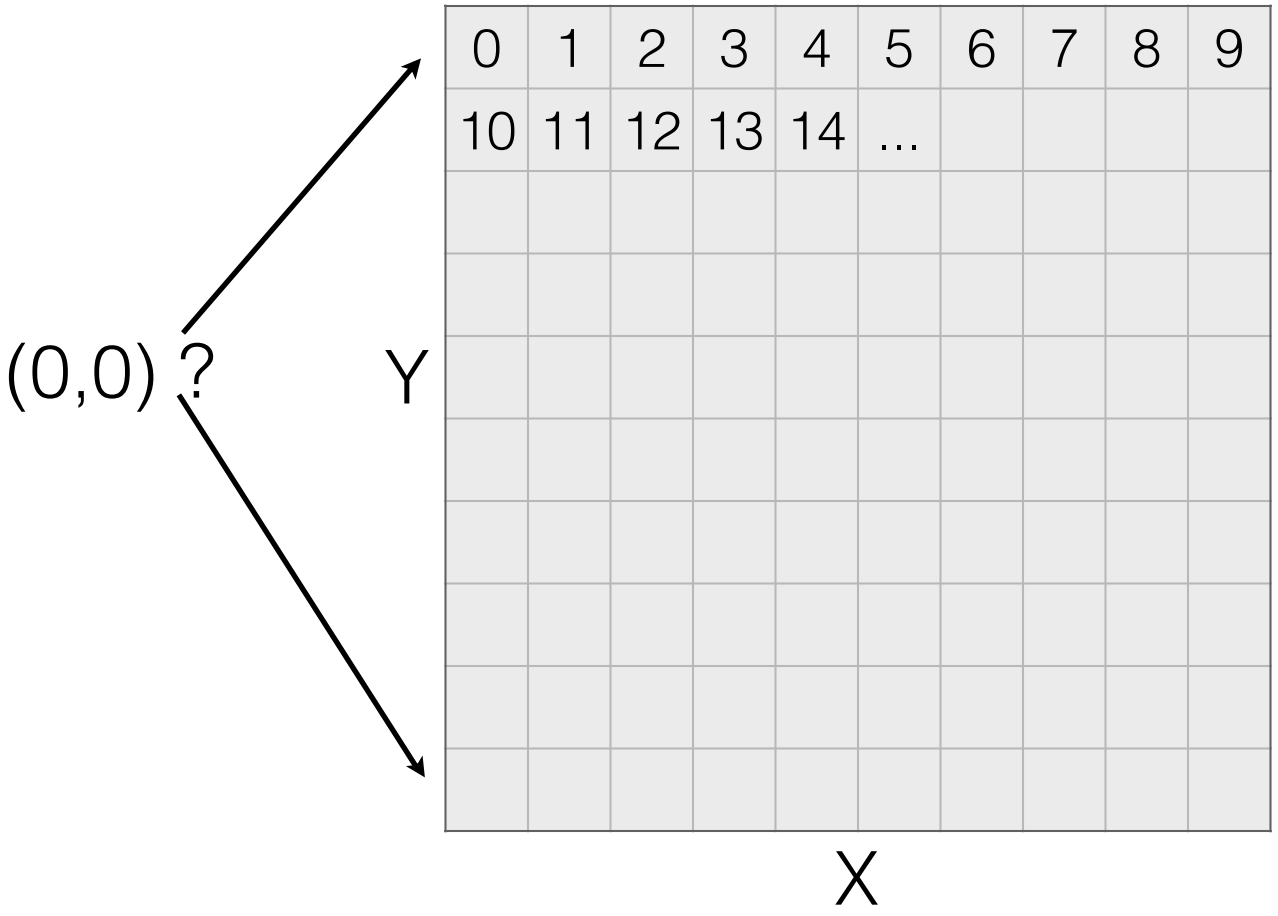
Installing PIL

- https://pypi.python.org/pypi/Pillow/2.3.0
- 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

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

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

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

* - 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="\0"*(128*128*4)        # string of zero pixels
from array import array
data=array("c",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("hh.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
  - 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")

Scientific Image Processing ?

- Quantitative
- 1-D, 2-D, 3-D, n-D
- Stacks of images

Tools

- ImageJ (<1997)
  - 8, 16, 32 bit images & stacks, Color
  - Java-based, Python support via Fiji
- ITK (~1999)
  - 2D & 3D, wide range of formats
  - Supports Visible Human project
  - Very flexible, but difficult to use, marginal Python
- EMAN2 (~1998, ~2005)
  - 1-D, 2-D, 3-D, images & stacks. Floating point greyscale
  - Designed for Cryo-EM, has been used with other microscopies, tissue expression data, FMRI, ...
  - C++ library with Python bindings, all programs in Python
  - Qt & OpenGL GUI
  - ~1500 Python functions/methods
  - ~200,000 lines of Python code and ~200,000 lines of C++ code

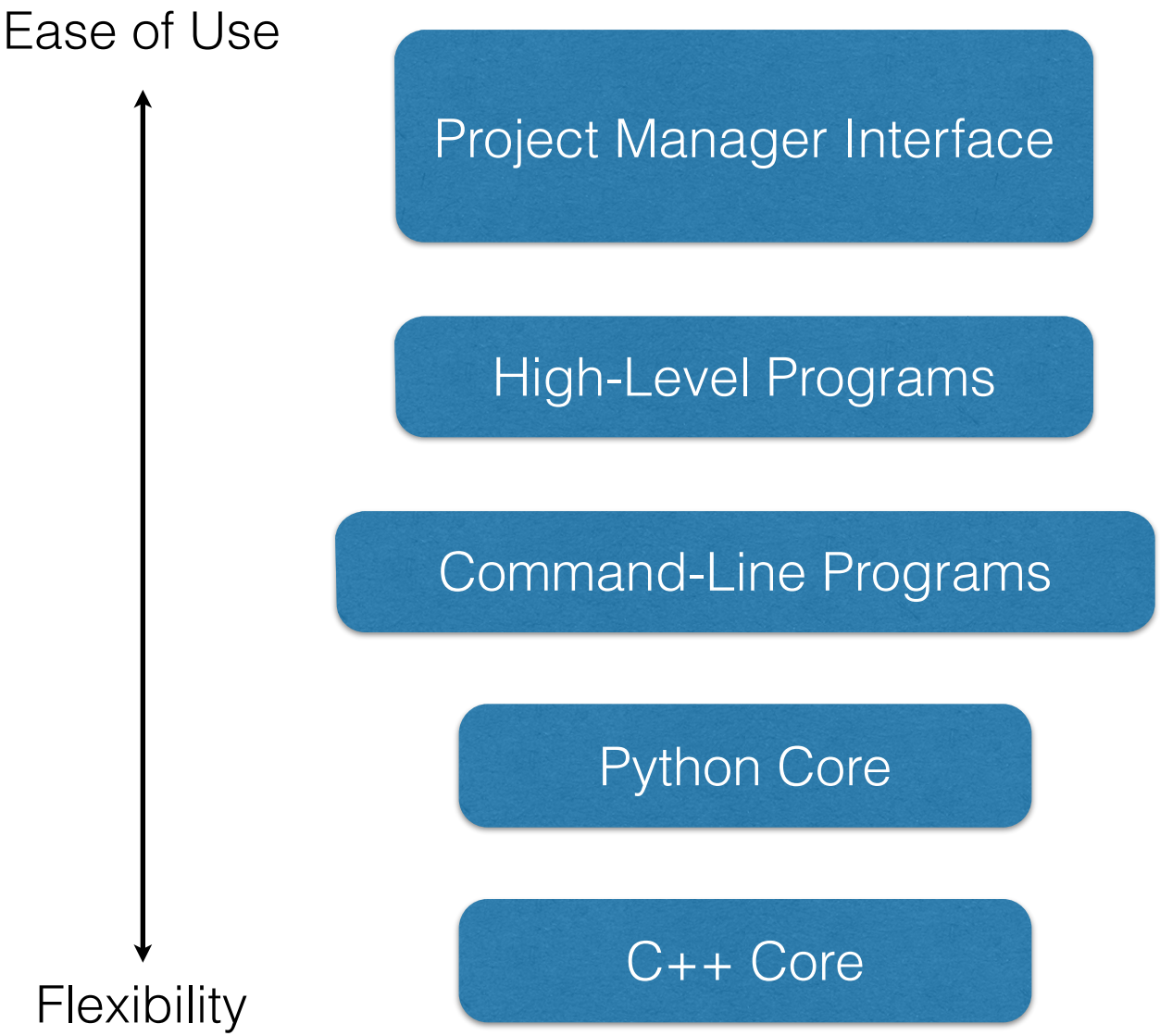
EMAN2

- EMAN2 Wiki (incl. download links):
  - http://blake.bcm.edu/emanwiki/EMAN2
- Discussion Mailing List/Google Group:
  - https://groups.google.com/forum/?fromgroups#!forum/eman2

EMAN2 Binaries Include

- Python 2.7 (except Mac)
- ipython
- PyQt4
- PyOpenGL
- matplotlib (&numpy)
- numerous C dependencies

EMAN2 Architecture



File Formats

MRC	R/W	IMAGIC	R/W
SPIDER	R/W	HDF5	R/W
PIF	R/W	ICOS	R/W
VTK	R/W	PGM	R/W
Amira	R/W	Xplor	W
Gatan DM2	R	Gatan DM3	R
TIFF	R/W	Scans-a-lot	R
LST	R/W	PNG	R/W
Video-4-Linux	R	JPEG	W

GUI

- e2display.py - General image/volume display
- e2filtertool.py - Build chains of image processing operations
- e2boxer.py - Interactive particle picker
- e2helixboxer.py - Filament picker
- e2tomoboxer.py - Interactive tomogram picker

Extensible Core

Type	Description	#
Processor	Generic image processing algorithms, filters, masks,	175
Aligner	Algorithms used to align 2 images or volumes to each other	22
Projector	Routines to generate 2-D projections of 3-D objects	6
Reconstructor	Routines to reconstruct 3-D objects from 2-D projections	11
Cmp	Similarity metrics used to compare two images or volumes	10
Averager	Average together stacks of images in various ways	7
Analyzer	Perform various operations on sets of images, such as classification or	6
Orientgen	Routines describing how projections cover the asymmetric	6

EMAN2 Intro

```
e2.py
Welcome to EMAN2
Prompt provided by IPython
Enter '?' for ipython help
In [3]: img=test_image()
In [4]: display(img)
-- make sure you can see the image
In [5]: img.mult(-1)
In [6]: b=[1,4,9,16,25,36,49,64]
In [7]: display(b)
In [8]: b[4]=4
-- right click on the plot window
In [9]: c=test_image_3d()
In [10]: display(c)
```

Create Images

- img= EMData() # empty image, default size
- img= EMData(nx,ny,nz) # specify size
- img= EMData(filename,...) # see next page
- img.to_zero() # all pixels = 0
- img.to_one() # all pixels = 1.0

Reading Images

- img= EMData()
- img.read_image(filename,img #,[header],[region],[3d])
- img=EMData(filename,img #,[header],[region],[3d])
- imgs=EMData.read_images(filename,[# list],[header])
- db=db_open_dict(“bdb:mydata”)
- img=db[#]

Writing Images

- img.write_image(filename,img #,[type],[header],[region],[mode])

Image Pixels

- img[0,0] # pixel value in lower left corner of image
- img[0,0]=22.5 # change the value at 0,0
- imgf=img.do_fft()
- imgf[10,10] # complex value at 10,10

Image Attributes

- img.get_attr(“mean”)
- img[“mean”]
- img.get_attr_dict()
- img[“myattr”]=“a string”