Introduction to Python and Python/EMAN

Steve Ludtke
sludtke@bcm.tmc.edu

Python ?

PYTHON OOL- developed by Guido van Rossum, and named after Monty Python.(No one Expects the Inquisition) a simple high-level interpreted language. Combines ideas from ABC, C, Modula-3, and ICON. It bridges the gap between C and shell programming, making it suitable for rapid prototyping or as an extension of C. Rossum wanted to correct some of the ABC problems and keep the best features. At the time, he was working on the AMOEBA distributed OS group, and was looking for a scripting language with a syntax like ABC but with the access to the AMOEBA system calls, so he decided to create a language that was extensible; it is OO and supports packages, modules, classes, user-defined exceptions, a good C interface, dynamic loading of C modules and has no arbritrary restrictions.

www.python.org

So, why not X ?

- C/C++ : Fast code, good reusability, but: risky, pointers, memory management, hard for beginners
- Fortran 77/90/95 - yeeech
- Java : Strongly structured, enforces good programming, but: a pain to use, many portability/version problems, language gets in the way of getting work done
- Perl - Great for text processing, concise syntax, but impossible to learn, remember syntax, or write clean code
- Tcl - Used as embedded control language, and provided Tk, but very limited language features, somewhat difficult syntax

Python Features

- Simple, easy to learn
- Gigantic set of built-in libraries
- Portable (virtually all computers/OS supported)
- Interpreted and byte-compiled (like Java)
- Object oriented
- Very popular for application scripting (especially in scientific apps)
- Python for high-level work, with compute-intensive work in C++/Fortran libraries

Hello World

- Python

```
print “Hello, world!”
```

- Perl

```
print "Hello, world!\n";
```

- C:

```
#include <stdio.h>

main() {
    printf(“Hello, world!\n”);
    exit(0);
}
```

2 Col File Into Array With y^2

- Python

```
import sys
lines=sys.stdin.readlines()

result=[]
for line in lines:
        x=line.split()
        result.append([x[0],float(x[1])**2])
```

2 Col File Into Array With y^2

- Python

```
import sys
lines=sys.stdin.readlines()

result=[]
for line in lines:
        x=line.split()
        result.append([x[0],float(x[1])**2])
```

- Perl

```
@lines=readline <STDIN>;

@resultx=();
@resulty=();
for ($i=0; $i<@lines; $i++) {
      $line=$lines[$i];
      @x=split /\s/,$line;
      push @resultx,($x[0]);
      push @resulty,($x[1]**2);
}
```

2 Col File Into Array With y^2

- Python

```
import sys
lines=sys.stdin.readlines()

result=[]
for line in lines:
        x=line.split()
        result.append([x[0],float(x[1])**2])
```

- Perl

```
@lines=readline <STDIN>;

@resultx=();
@resulty=();
for ($i=0; $i<@lines; $i++) {
      $line=$lines[$i];
      @x=split /\s/,$line;
      push @resultx,($x[0]);
      push @resulty,($x[1]**2);
}
```

- C

```
#include <stdio.h>

main() {
char line[80];
float *resultx=(float *)malloc(8*sizeof(float));
float *resulty=(float *)malloc(8*sizeof(float));
int nalloc=8;
int n=0;

while (fgets(line,79,stdin)) {
   sscanf(line,"%f %f",resultx+n,resulty+n);
   resulty[n]*=resulty[n];
   n++;
   if (n==nalloc) {
      nalloc*=2;
      resultx=(float*)realloc(resultx,nalloc*sizeof(float));
      resulty=(float*)realloc(resulty,nalloc*sizeof(float));
   }
}
```

Interactive Python

- Python as a calculator
- strings, tuples, lists, dictionaries
- import (os,math,sys,string,random,etc.)
- help()
- if, while, for
- def
- files, readline, readlines

Basic Syntax Reference

- Indent
- Numbers:

```
0 1.5 1.2e43+2j
```

- Strings:

```
“test string” 'this too'
“””multiple line
string”””
```

- Lists:

```
lst=[1,2,'abc',1+3j]
lst[0]
```

- Dictionaries:

```
dict={'key1':'value1','key2':'value2',3:'value3'}
dict['key2']
dict[3]
```

- import:

```
import os
from math import *
```

- print:

```
print 'x=',x,'   y=',y
print 'x=%f y=%f'%(x,y)
```

- if,else:

```
if x>5:  print “x>5”
elif x<0:  print “x<0”
else:  print “0<x<5”
```

- for loops:

```
for i in list: print i
```

- while loops:

```
while x<10:
   print x
   x*=1.5
```

Simple Math Example

```
p=[]

for i in range(2,1000):
    for j in p:
        if (i%j==0): break
    else: p.append(i)

print p
```

Python Scripting for EMAN

- EMAN vs EMAN2

Python Scripting for EMAN

- *from EMAN import **
- *x=EMData()*

- or -

- *import EMAN*
- *x=EMAN.EMData()*

- *help('libpyEM')*

Display 'bad' class averages

```
cd ~/demo/groel/stage5
python
from EMAN import *
lst=readImages('classes.6.hed',-1,-1)
badlst=[]
for i in range(0,len(lst),2):
    if (lst[i].lcmp(lst[i+1],1,0)>.9) : badlst.extend([lst[i],lst
      [i+1]])
display(badlst)
```

Sort images by # particles

```
cd ~/demo/groel/stage5
python
from EMAN import *
lst=readImages('classes.6.hed',-1,-1)
lst.sort(lambda x,y: cmp(y.NImg(),x.NImg()))
lst=filter(lst,lambda x: (x.NImg()>0))
display(lst)
```

'Homework'

1) Write a program to factorize any number.

2) Write a program to find all numbers <1024 that are multiples of only 2,3,5 and 7 and are divisible by 4.

3)Write a program that reads the Euler angles from the header of images in a stack file and prints them out in the SPIDER convention

4)Write a program that generates 50 randomly oriented projections of a 3D model and writes them to a stack file

5)Write a program that reads a reference and a stack of images, rotationally aligns each image in the stack to the reference, then writes the average of the aligned stack to an output file

Problem #1

```
def factorize(x):
  ret=[]
  while x>1:
    for i in range(2,x/2):
      if (x%i==0):
        x/=i
        ret.append(i)
        break
    else:
      ret.append(x)
      break
  return ret
```

Problem #2

```
r=[]
for i in range(2,10):
    v=2**i
    for j in range (7):
        v1=v*3**j
        if (v1>1024) : continue
        for k in range (5):
            v2=v1*5**k
            if (v2>1024) : continue
              for l in range (4):
                v3=v2*7**l
                if (v3>1024) : continue
                r.append(v3)

r.sort()
print r
```

Problem #3

```
from EMAN import *

lst=readImages('input.hed',-1,-1)
for img in lst:
    print x.getEuler().getByType(Euler.SPIDER)
```

Problem #4

```
from EMAN import *
from random import random

model=EMData()
model.readImage('volume.mrc',-1)
for i in range(50):
     proj=model.project3d(random()*pi/2,
          random()*pi*2,random()*pi*2,-1)
     proj.writeImage('prj.img',-1)
```

Problem #5

```
from EMAN import *

ref=EMData()
ref.readImage('ref.hed',0)
toali=EMData()
sum=ref.copyHead()
sum.zero()
n=0
while (toali.readImage('input.hed',n)==0) :
        toali.rotAlign(ref)
        toali.rotateAndTranslate()
        sum=sum+toali

sum/=float(n)
sum.writeImage('out.mrc')
```

Python / EMAN

The following slides define the Python interface for EMAN1.6, the interface for EMAN2 (under development) has been substantially simplified and improved. We suggest that new development take place using the EMAN2 infrastructure rather than EMAN1. If you develop in EMAN2 in its current pre-alpha state, you may wish to contact sludtke@bcm.tmc.edu so you will be aware of the current state of development.

Library Methods (Objects)

- EMData - Main image class that encapsulates 1D, 2D or 3D image data.
- Euler - Representation of orientations, with functions to change between representations.
- XYData - For representing arbitrary 2-column numerical data as used in an x/y plot.
- VolFrag - for reduced memory representations of large images.

Library Methods (File I/O)

- int fileCount(string path)
- readImage(string path,int n,bool nodata)
- List readImages(string path)
- writeImage(string path,int n,ImageType type)

Library Methods (Info/Statistics)

- int xSize() (y/z)
- setSize(int x,y,z)
- bool isComplex()
- Euler getEuler()
- float alt(),az(),phi()
- setRAlign(float alt,az,phi)
- setRAlign(Euler ort)
- float Dx(),Dy(),Dz()
- setTAlign(dx,dy,dz)
- float Min(),Max()
- (x,y,z) minLoc(),maxLoc()
- float Mean(),Sigma()
- float Mean2(),Sigma2()
- float edgeMean()
- float cirMean()
- float Skewness()
- float Kurtosis()
- float dCenter()

Library Methods (Simple Image Processing)

- EMData +,-,*,/
- EMData copy()
- EMData copyHead()
- applyMask(int r,type)
  - 0: step cutoff to the mean value
  - 1: fills with flatband random noise
  - 2: is a tanh falloff to 0
  - 3: Gaussian falloff, r is the 1/e width
  - 4: step cutoff to 0
  - 5: INNER step cutoff to 0
  - 7 generates a 2 pixel ring of '1', replacing the current contents

- EMData clip(int x0,y0,w,h)
- EMData clip(int x0,y0,z0,w,h,d)
- zero(),one()
- hFlip(),vFlip()
- normalize()
- edgeNormalize(bool cir)
- normalizeMax()
- normalizeTo(EMData noisy,bool keepzero,invert)
- radialAverage(), radialSubtract()
- maskNormalize(EMData mask,bool sigmatoo)

Library Methods (Fourier)

- EMData doFFT()
- EMData doIFT()
- ri2ap(),ap2ri()
- toCorner()
- EMData convolute(EMData with)
- EMData calcCCF(EMData with, filter=None)
- EMData calcMCF(EMData with, filter=None)
- EMData littleBigDot(EMData with)

- filter(float hp,lp,int type)
  - 0 -> sharp cutoff
  - 1 -> gaussian lowpass (tanh hp)
  - 2 -> hyperbolic tangent
  - 3 -> butterworth highpass
  - 4 -> undo tanh highpass
  - 5 -> gaussian tanh mix
  - 6 -> linear ramp
  - 7 -> inverse gaussian highpass
  - 8 -> Gaussian highpass & lowpass

Library Methods (alignment)

- transAlign(EMData with,bool parent,bool int,int maxshift)
- rotAlign(EMData with)
- rotAlignPrecen(EMData with)
- transAlign3d(EMData with,bool parent, bool intonly,int maxshift)
- EMData RFAlign(EMData with,flip=None);
- EMData RTAlign(EMData with,int usedot=0,int maxshift=-1)
- EMData RTAlignRadon(EMData with,int maxshift=-1,EMData radonwith=None,radonthis=None)
- EMData RTFAlign(EMData with,flip=None,int usedot=1,int maxshift=-1);
- EMData RTFAlignSlow(EMData with,flip=None,int maxshift=-1)
- EMData RTFAlignSlowest(EMData with,flip=NULL,int maxshift=-1)
- EMData RTFAlignRadon(EMData with,int maxshift=-1,EMData thisf=None,radonwith=None, radonthis=None, radonthisf=None)
- refineAlign(EMData with,int mode)
- cmCenter(bool intonly)

Library Methods (realFilter)

- realFilter(int type, float v1,v2,v3)
  - 0: threshold filter, x<v1 -> 0
  - 1: x<v1 -> 0, then normalize
  - 2: x<v1 -> 0 and x>=v1 -> 1
  - 3: x>v1 -> x, 0<x<v1->0,   x<0 -> v2, then normalize
  - 4: x -> fabs(x)
  - 5: |x|-=v1, x->0 if |x|<0
  - 6: x!=0 x->1
  - 7: x<v1*maxval -> 0, normalize
  - 8: x>v1 ->0
  - 9: x = (x+v1)*v2
  - 10: x = sqrt(x) or -sqrt(-x)
  - 11: x -> exp(x/v1-v2), x/v1-v2>40 -> 40
  - 12: x -> 2*(v1-x)
  - 14: v1<=x<=v2 -> 1.0 else 0.0
  - 15: x -> log(x)
  - 16: x -> e^(x)
  - 17: x>mean+v1*sigma or x<mean-v2*sigma -> mean
  - 18: x -> x^2
  - 20: is a median filter with a +-val1 size, val1=1 -> 3x3 median
  - 21: like 20, but fills std dev in the box
  - 22: like 20, but max instead of median
  - 51: Vertical stripe filter, tries to remove vertical scanner CCD norm. artifacts

Library Methods (realFilter contd)

- 52: Block filter, v1 is dx/dy for calc an average, v2 is dx/dy for fill/step
- 53: Block filter, v1 is dx/dy, v2 is lp freq cutoff in pixels60: fill zeroes at edges with nearest horizontal/vertical value
- 61: fill constant areas with zero
- 62: eliminate beamstop in electron diffraction patterns, v1=sig multiplier, v2,3 are x,y of center, v1<0 -> radial subtract
- 63: fill zeroes at edges with nearest horizontal/vertical value damped towards Mean2() at edge
- 80: ~Laplacian filter, x -> d^2/dx^2+d^2/dy^2
- 90: peak filter -> pixel = pixel - max of values surrounding pixel
- 91: peak filter -> around x>x -> 0
- 100: x -> radius^2 (overwrites data)
- 101: x -> vert avg at x, this is ~ what is subtracted in 51
- 102: zero top and bottom edges, v1: nrows on top, v2: nrows on bottom
- 103: x -> radius (overwrites data)
- 104: transmittance to optical density conversion for images from Nikon coolscan 8000
- 105: zero edges of image, v1 is rows on top and bottom, v2 is rows on left and right

Library Methods (Data Access)

- getData() (C++ only)
- getDataRO() (C++ only)
- doneData() (C++ only)
- float valueAt(int x,y)
- float valueAt(int x,y,z)
- float valueAtSafe(int x,y,z)
- float valueAtSafe(int x,y)
- float setValueAt(int x,y,z, float newval)
- update() (must be called after data is modified with setValueAt)

Library Methods (2D<->3D)

- EMData project3d(float alt,az,phi, int mode, float thr)
- cutSlice(EMData map,float z,Euler ort)
- uncutSlice(EMData map,float z,Euler ort)
- insertSlice(EMData slice, float alt,az,phi,int mode,float weight)
- insertSlice3DReal(EMData slice,float weight)
- insertSliceWF(EMData slice, float alt,az,phi,int mode,list SNR,float padratio)
- setup4IS(int size)
- setup4Slice(bool redo)
- normSlice(EMData slice,float alt,az,phi)
- doneSlice(),doneSliceWF()

Library Methods (Euler)

- Euler Euler(float alt,az,phi)
- float alt(),az(),phi()
- tuple getByType(type t)
  - ie: Euler.MRC
- setByType(tuple v,type t)
- setSym(String symmetry)
- Euler NextSym()
- Euler SymN()
- int valid()
- Euler EulerMult(Euler)
- Euler FromLeft(Euler)
- int getMaxSymEl()
- Euler inverse()
- float diff(Euler)