

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 arbitrary 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 1.51.2e4 3+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

- *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)
```

Python Scripting for EMAN

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

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')
```