# Lecture 5

Math and Plotting

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

# Despair

(don't)

1) "I'M NOT LEARNING ANYTHING, THE HOMEWORK IS TAKING FOREVER AND IS IMPOSSIBLE !
If you're spending time trying to figure out the homework, you're learning more than you think.

2) "I'M GOING TO FAIL !"
As long as you keep trying, and turn in your (even failed) attempts to figure it out, you will stay above the dreaded C.

## Programs you can Run

- Do NOT use a word-processor like Word, Pages, etc. Use a 'text editor'. The built-in editor ‘idle’ is a good choice for beginners.
- Just type ‘python program.py’

**or**

- use a ‘.py’ extension for your programs
- for unix/mac, put:
- #!/usr/bin/env python
- on the first line of the file, and type:
- chmod a+x file.py

# Homework Review

- Rewrite the sequence translation assignment from the last homework to use biopython to represent the sequences and perform the translation for you, instead of using your own dictionary.

- Write a program using biopython to perform a pubmed query to retrieve reference citations for all of the papers published by your mentor (or anyone else with at least 5 publications). Use the citation list to identify all of the coauthors he/she has published with, and count the number of joint articles with each coauthor. Turn in the program and the output of the program for the person you ran it on. You do not need to do error checking for things like the same coauthor with slight name variations.

- Rewrite the sequence translation assignment from the last homework to use biopython to represent the sequences and perform the translation for you, instead of using your own dictionary.

```
import sys

xlate={"ttt":"f" ... "ggg":"g"}

fsp=sys.argv[1]
dna=file(fsp,"r").read()
dna=dna.translate(None,"0123456789 \t\n\r").lower()
out=(fsp+”.prot","w")
dna=dna[dna.find(“atg”):]

for i in xrange(0,len(dna),3):
 triplet=dna[i:i+3]
 try: amino=xlate[triplet]
 except:
  print "Unknown triplet: ",triplet
  sys.exit(1)
 if amino==“0” : break
 out.write(amino)

out.write("\n")
```

```
import sys
from Bio.Seq import Seq
from Bio.Alphabet import IUPAC

fsp=sys.argv[1]
dna=file(fsp,"r").read()
dna=dna.translate(None,"0123456789 \t\n\r").lower()
out=(fsp+".prot","w")
dna=dna[dna.find("atg"):]

seq=Seq(dna,IUPAC.unambiguous_dna)
out.write(seq.translate())
```

File reading replacement ?
http://biopython.org/DIST/docs/tutorial/Tutorial.html#sec52

Write a program using biopython to perform a pubmed query to retrieve reference citations for all of the papers published by your mentor (or anyone else with at least 5 publications). Use the citation list to identify all of the coauthors he/she has published with, and count the number of joint articles with each coauthor. Turn in the program and the output of the program for the person you ran it on. You do not need to do error checking for things like the same coauthor with slight name variations.

- How do we represent the data ?
  - Most objects determined by BioPython
  - Use Collections.Counter to count
- Break the task into small pieces
- Code each of the pieces

# Break the Task into Small Pieces

- Initialize Entrez
- Query for list of pubmed ids for author
- Retrieve all PubMed entries, MedLine parsed
- Incorporate each author’s list into Counter object
- Print results

```
from Bio import Entrez
Entrez.email = "me@me.com"
handle = Entrez.esearch(db="pubmed", term="Ludtke SJ[Author]" ,retmax=500)
record = Entrez.read(handle)
handle.close()
ids=record[“IdList"]
print ids
```

```
['23211764', '22696402', '22405004', '22325770', '21827954', '21827945',
'21220123', '21121065', '21038867', '20935055', '20888963', '20885381',
'20835797', '20462491', '20194787', '20090755', '20080547', '22353480',
'19580754', '19446530', '19264960', '19191587', '18621707', '18536725',
'18334219', '18158904', '17327167', '17098196', '16931051', '16928740',
'16859925', '16491093', '16364911', '16271896', '16140524', '16084392',
'16075070', '15811380', '15581887', '15272307', '15242589', '15193640',
'15166242', '15065668', '14962379', '14750990', '14643210', '12714606',
'12218169', '12149473', '11756679', '11718559', '11356062', '11352589',
‘10600563’, '9199788',  '8913604',  '8901513',  '8744303',  ‘7495788',
'7647240',  '8110813',  '7510532',  '7679294']
```

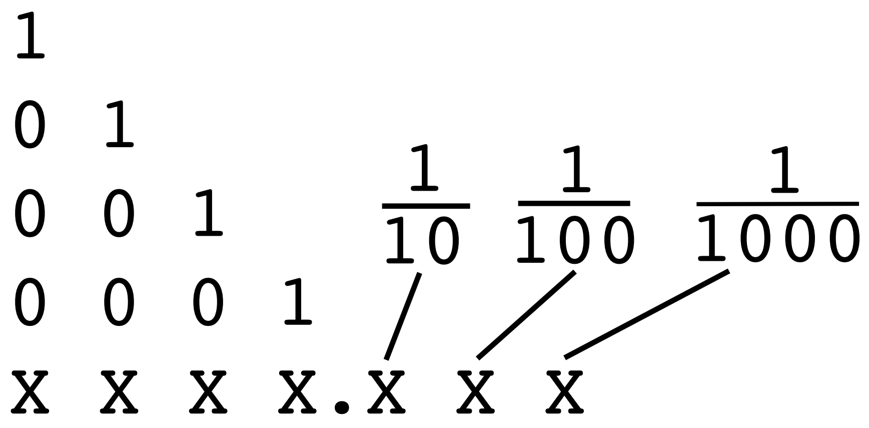
```
from Bio import Entrez
from Bio import Medline

Entrez.email = "sludtke@bcm.edu"
handle = Entrez.esearch(db="pubmed", term="Ludtke SJ[Author]", retmax=500)
record = Entrez.read(handle)
handle.close()
ids=record["IdList"]

handle=Entrez.efetch(db=“pubmed”,id=ids[0],rettype="medline")
records = list(Medline.parse(handle))
```

```
from Bio import Entrez
from Bio import Medline

Entrez.email = "sludtke@bcm.edu"

def pubsbyauthor(query):
handle = Entrez.esearch(db="pubmed", term="Ludtke SJ[Author]", retmax=500)
record = Entrez.read(handle)
handle.close()
ids=record["IdList"]

handle=Entrez.efetch(db="pubmed",id=tuple(ids),rettype="medline")
records = list(Medline.parse(handle))
handle.close()

from collections import Counter

ctr=Counter()

for r in records:
   ctr.update(r["AU"])

import pprint
print len(ctr)
pprint.pprint(ctr.most_common())
```

# Decimal Numbers

```
1
0 1
0 0 1
0 0 0 1
x x x x
```

# Decimal Numbers



# Binary Numbers

```
1
2 6 3 1
8 4 2 6  8 4 2 1
x x x x  x x x x
```

```
  1    0000 0001
  2    0000 0010
  4    0000 0100
  8    0000 1000
 15    0000 1111
212    1101 0100
```

# Binary Numbers

```
1                        1   1   1
2 6 3 1                  -   -   -
                         2   4   8
8 4 2 6  8 4 2 1        /   /   /
x x x x  x x x x.x x x
```

```
0.25     0000 0000.010
0.625    0000 0000.101
0.3      0000 0000.0100 1100 1100 ...
```

# IEEE Floating Point

- "Binary Scientific Notation"
- 127.25        1111111.01
- $1.2725\text{x}10^{2} \rightarrow 1.11111101\text{x}10^{110}$

- Single:
- SEEEEEEE EMMMMMMM MMMMMMMM MMMMMMMM
- S – Sign bit 0=+
- E – Exponent, bias 127
- M – Significand (Mantissa), implicit 1 when normalized

# Digital Representation of Numbers

normal Python

| | | |
|---|---|---|
| • Bit | 0-1 | |
| • Nibble | (4 bits) | 0-15 |
| • Byte (char) | (8 bits) | 0-255 |
| • Word (short) | (16 bits) | 0-65,535 |
| • Longword (long) | (32 bits) | 0-4,294,967,296 |
| • Long Longword | (64 bits) | 0-1.844x$10^{19}$ |
| • Float | (32 bits) | $10^{38}$ |
| • Double | (64 bits) | $10^{308}$ |

Python “long integers” are a special object, not a standard number.

# Numerical Processing

- NumPy - Fast arrays, linear algebra, etc.
- Matplotlib - Matlab-style plotting interface
- SciPy - Large library of numerical computing algorithms

# Installing SciPy,etc.

- NumPy and matplotlib will be sufficient for the homework, SciPy is also a useful tool
- www.scipy.org
- Can consider: http://matplotlib.sourceforge.net/users/installing.html
- Linux
    - Use your package installer, should be available
- Mac (system python includes numpy)
    - If you are using 10.9 (and you should be) or 10.7-10.8, you may try:
        - http://fonnesbeck.github.com/ScipySuperpack/
    - For Snow Leopard or as an alternative for 10.7-10.9:
        - You can get Matplotlib easily as a side effect of installing (which we will use in a later lecture): http://ncmi.bcm.edu/ncmi/software/counter_222/software_86
        - Otherwise, this tip may help: http://blake.bcm.edu/emanwiki/EMAN2/COMPILE_EMAN2_MAC_OS_X#matplotlib_.281.0.29
- Windows
    - 32 bit - http://sourceforge.net/projects/scipy/files/scipy/0.13.2
    - 64 bit - You can try: http://www.lfd.uci.edu/~gohlke/pythonlibs/

## What’s wrong with this ?

```
from math import *

x=[i/100000.0 for i in xrange(1000000)]
y=[sin(i) for i in x]
```

## What’s wrong with this ?

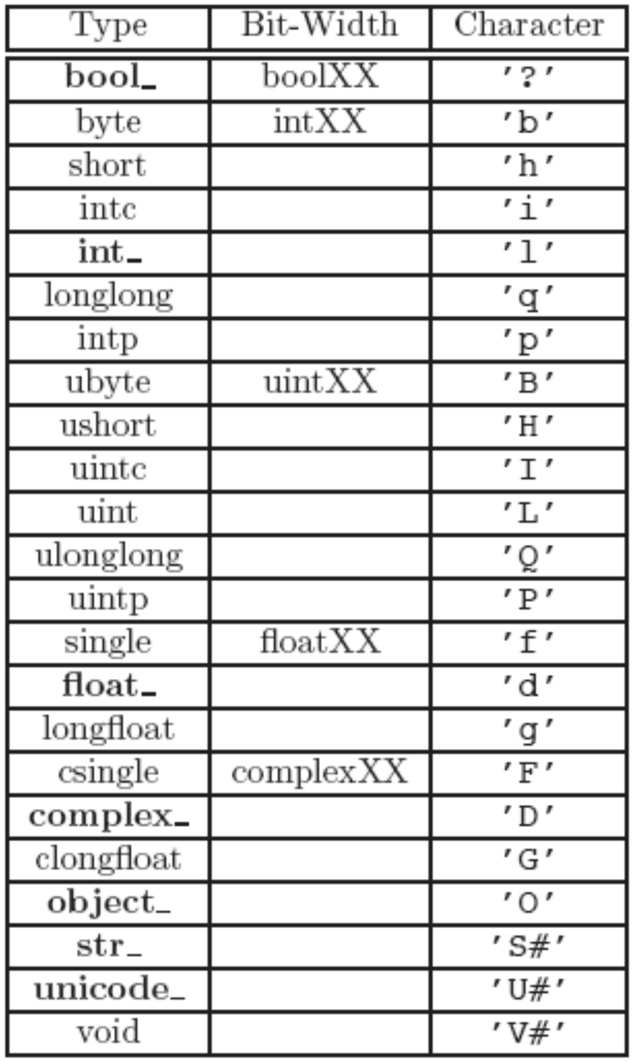
```
from math import *

x=[i/1000000.0 for i in xrange(10000000)]
y=[sin(i) for i in x]
yy=tuple(y)

import sys
print sys.getsizeof(y),sys.getsizeof(yy)
```

## How about this ?

```
from numpy import *

x=arange(0,10.0,0.000001)
y=sin(x)
```

# NumPy

- http://csc.ucdavis.edu/~chaos/courses/nlp/Software/NumPyBook.pdf # numpy book
- from numpy import *
- a=arange(60)
- b=a.reshape(10,6) # make 2-D matrix
- c=a.reshape(3,4,5) # make 3-D (tensor)
- b.shape # current dimensions
- b.size # total number of elements
- b.ndim # dimensionality
- b.dtype # type of value stored
- b.astype("")



| Type | Bit-Width | Character |
|---|---|---|
| **bool_** | boolXX | '?' |
| byte | intXX | 'b' |
| short | | 'h' |
| intc | | 'i' |
| **int_** | | 'l' |
| longlong | | 'q' |
| intp | | 'p' |
| ubyte | uintXX | 'B' |
| ushort | | 'H' |
| uintc | | 'I' |
| uint | | 'L' |
| ulonglong | | 'Q' |
| uintp | | 'P' |
| single | floatXX | 'f' |
| **float_** | | 'd' |
| longfloat | | 'g' |
| csingle | complexXX | 'F' |
| **complex_** | | 'D' |
| clongfloat | | 'G' |
| **object_** | | 'O' |
| **str_** | | 'S#' |
| **unicode_** | | 'U#' |
| void | | 'V#' |

# NumPy

- a=zeros((nx,ny,...))
- a=fromfunction(lambda i,j:i+j,(4,5))
- a=arange(0,20,.1)    #
- a*10     # multiply each element !
- b=sin(a)  # sin() of each element
- c=a[c>0]   # condition, returns elements >0
- c.sort()   # sort values in-place
- c.mean(),var(),std(),prod()   # average, variance, standard dev, product
- inner(a,b), outer(a,b)        # inner and outer matrix products
- dot(a,b), cross(a,b)           # dot and cross products (similar to above)
- histogram(a,bins,range)     # compute a histogram of ‘a’

# SciPy

- Clustering package (scipy.cluster)
- Constants (scipy.constants)
- Fourier transforms (scipy.fftpack)
- Integration and ODEs (scipy.integrate)
- Interpolation (scipy.interpolate)
- Input and output (scipy.io)
- Linear algebra (scipy.linalg)
- Maximum entropy models (scipy.maxentropy)
- Miscellaneous routines (scipy.misc)
- Multi-dimensional image processing (scipy.ndimage)
- Orthogonal distance regression (scipy.odr)
- Optimization and root finding (scipy.optimize)
- Signal processing (scipy.signal)
- Sparse matrices (scipy.sparse)
- Sparse linear algebra (scipy.sparse.linalg)
- Spatial algorithms and data structures (scipy.spatial)
- Special functions (scipy.special)
- Statistical functions (scipy.stats)
- Image Array Manipulation and Convolution (scipy.stsci)

# matplotlib (pylab)

- Matlab-like plotting library
- http://matplotlib.sourceforge.net/users/pyplot_tutorial.html

```
ipython --pylab         # special mode for interaction with pylab

x=arange(0,4*pi,0.05)  # from numpy

y=sin(x)               # easy to apply a function to a list of values

plot(x,y)              # plot x,y and open a display window
```

**OR**

```
python

from pylab import *    # <-- only if you don't use ipython

x=arange(0,4*pi,0.05)

y=sin(x)               # easy to apply a function to a list of values

plot(x,y)              # plot x,y and open a display window

show()                 # opens the plot window (blocks on some machines)
```

# matplotlib (pylab)

```
ipython --pylab    # special mode for interaction with pylab

x=arange(0,4*pi,0.05)  # from numpy

y=sin(x)               # easy to apply a function to a list of values

y2=cos(x)

figure(1)              # start a new figure

subplot(211)           # make a 1x2 set of plots and move to 1st

plot(x,y)              # plot x,y and open a display window

ylabel("sin(x)")

subplot(212)           # start on the 2nd subplot

plot(x,y2)             # second plot

ylabel("cos(x)")

xlabel("x")
```

# Homework 5

- Problem 1 - Tell me what your plan is for a class project. Specifically: what will the program do, what sort of data will it work with, and what will it do with that data ? Again, it's ok if you're forced to change some of the details as your project develops, but you need to start with a plan.

- Problem 2:

    1. Install numpy/matplotlib/[scipy] on your computer.

    2. Write a simple CSV plotting program : It's very easy to export data from a spreadsheet as a CSV (comma separated value) file. While spreadsheets are great at creating business charts, they are less good at generating scientific plots, and value manipulations are somewhat limited. Your program should:

        A. Read a user-specified CSV file.

        B. The user should pick which columns from the file to use for X and Y

        C. Make a simple line or scatter plot of the data using matplotlib. The plot can appear on the screen or be saved as a picture to a file. Your choice. You only need to turn in the .py program, not the output.

    - A couple of sample CSV files are provided on the class website.

    - Tip: Careful investigation of the manuals for numpy and python may reveal some possible solutions to make part of this assignment easier, but you may also just use string/file manipulation.