Scripting and Visualization

- **Scripting**
  - Python basics (Peter Teuben)
  - IDL basics (GDL) (Mark Vogelsberger)
- **Visualization**
  - Python::Matplotlib, tvtk, mayavi (Matthew Turk)
  - Vis5d , IDL widgets (Anatoly Spitkovsky)
  - PovRay, SecondLife (Derek Richardson)
  - Visit (Chris Lindner)
  - OpenGL (Hanno Rein)

S&V questions

- S: Is speed important?
- V: Specific for one type of data (point vs. grid, 1-2-3-dim?)
- V: Can it be driven by external programs (**ds9, paraview**)
- V: Can it be scripted? (**partiview**)
- V: Can it make animations? (**glnemo2**)
- V: Installation dependancies? Hard to install?

Scripting

(leaving out bash/tcsh)

- **Scheme** ((((((= a 1))))))
- **Perl** $^_a!
- **Tcl/Tk** [set x [foo [bar]]]
- **Python** a.trim().split(':')[5]
- **Ruby** @a = (1,2,3)
- **Cint** struct B { float x[3], v[3];}
- IDL wh = where(r(delm) EQ 0, ct)
- Matlab t=r(:,1)+r(:,2)/60+r(:,3)/3600;

PYTHON

http://www.python.org

What's all the hype about?

- 1990 at UvA by Guido van Rossum
- Open Source, in C, portable Linux/Mac/Win/...
- Interpreted and dynamic scripting language
- Extensible and Object Oriented
  - Modules in python itself
  - Modules to any language (C, Fortran, ...)
    - Many libraries have interfaces: *gsl, fitsio, hdf5, pgplot,....*
- SciPy environment (numerical, plotting)

The Language

http://docs.python.org/tutorial

- Types:
  - Scalars: X=1 X="1 2 3"
  - Lists: X=[1,2,3] X=range(1,4)
  - Tuples: X=(1,2,3,'-1','-2','-3') vx=float(X[3])
  - Dictionary: X={"nbody":10, "mode":"euler", "eps":0.05} X["eps"]
- Lots of builtin functions (and modules) to operate on types
- Control flow (!!! **indentation creates the control** !!!)

```
If i==5:
    i=i+1
elif i>10:
    i=i-1
else:
   i=0
```

```
for w in ["aap", "noot", "mies"]:
    print w[0],w[-2:]
```

```
def asqrt(x):
    if x<0: return math.sqrt(-x)
    return math.sqrt(x)
```

```
while i<10:
    i=i+1
    if bad(i): break
    good(i)
```

Example: nbody1.py

```
#! /bin/env python
#
#
#  Flavor 1:  only intrinsic python lists/arrays

import sys, os, math

def read_table(file):
    """ return the regular part of a table as a matrix"""
    f = open(file)                  # open file
    lines = f.readlines()           # get list of all lines
    f.close()                       # close
    t = []                          # prepare empty list of rows
    nc = 0                          # number of columns
    for line in lines:
        if line[0] == '#': continue          # skip comments
        words = line.split()
        if len(words) == 0: continue     # skip blank lines
        if nc == 0: nc = len(words)      # remeber # columns
        row=[]                           # empty row
        for w in words:
            row.append(float(w))         # fill the row
        t.append(row)                    # add row to table
    return t
```

```
def pyth2d(file='pyth2d.mpv'):
    """create the example MPV file for the experiments"""
    f = open(file,'w')
    f.write("3   1  3   0 0\n")
    f.write("4  -2 -1   0 0\n")
    f.write("5   1 -1   0 0\n")
    f.close()
    print 'File %s written' % file
```

```
class nbody:
    """ A class to experiment with nbody  in python"""
    def __init__(self,file):
        f = open(file)
        lines = f.readlines()
        f.close()
        self.n = 0
        self.t = 0.0
        self.m=[]
        self.x=[]
        self.vx=[]
        self.ax=[]
        self.y=[]
        self.vy=[]
        self.ay=[]
        nw = 0
        for line in lines:
            if line[0] == '#': continue
            words = line.split()
            if len(words) == 0: continue
            if nw == 0: nw = len(words)
            if nw != 5: continue
            self.x.append(float(words[1]))
            self.y.append(float(words[2]))
            self.vx.append(float(words[3]))
            self.vy.append(float(words[4]))
            self.ax.append(0.0)
            self.ay.append(0.0)
            self.n = self.n + 1
    def force(self,eps=0.0):
        """compute new accellerations based on current positi
        with given softening"""
        eps2 = eps*eps
        for i in range(0,self.n):
            self.ax[i] = 0.0
            self.ay[i] = 0.0
            xi=self.x[i]
            yi=self.y[i]
            for j in range(0,self.n):
                if i==j: continue
                dx = self.x[j]-xi
                dy = self.y[j]-yi
                p = 1.0/math.sqrt(dx*dx+dy*dy+eps2)
                p = self.m[j]*p*p*p;
                self.ax[i] = self.ax[i] + p*dx
                self.ay[i] = self.ay[i] + p*dy
    def euler(self,dt):
        """take a single forward Euler step"""
        for i in range(0,self.n):
            self.x[i]  = self.x[i]  + dt*self.vx[i]
            self.y[i]  = self.y[i]  + dt*self.vy[i]
            self.vx[i] = self.vx[i] + dt*self.ax[i]
            self.vy[i] = self.vy[i] + dt*self.ay[i]
        self.t = self.t + dt
```

Finally... work can be done

```
def bench(nstep=100, tstop=1.0, eps=0.1, file='pyth2d.mpv'):
    a = nbody(file)
    step = tstop/nstep
    print "step=",step
    a.force(eps)
    a.list()
    for i in range(0,nstep):
        a.euler(step)
        a.force(eps)
    a.list()
```

Examples in: PiTP's papers/exercises

Example: nbody1.py

- Simple Euler 3-body Pythagoras problem, 100000 steps of 0.00001 to T=1:
  - Classic C: 0.9 sec
  - Classic C (float): 1.2 sec - wow
  - Classic python-1: 3.5 sec - not bad
  - Numpy python-2: 13.3 sec - yikes!
  - Numpy python-3: 21.0 sec - wahooooooo!!!!

What is going on here ???

```
for i in range(0,self.n):
        self.x[i]  = self.x[i]  + dt*self.vx[i]
```

```
self.x  = self.x  + dt*self.vx
```

Example: nbody1.py

- Simple euler 3-body pythagoran problem, N steps of 1/N to T=1:
-

	N=3	10	40	250
	Steps=1e5	1e4	1e3	1e2
– Classic C:	0.9	0.16	0.12	0.44
– Classic python-1:	3.5	3.3	5.2	20.1
– Numpy python-2:	13.3	14.2	22.4	89.5
– Numpy python-3:	21.0	6.2	2.4	2.0

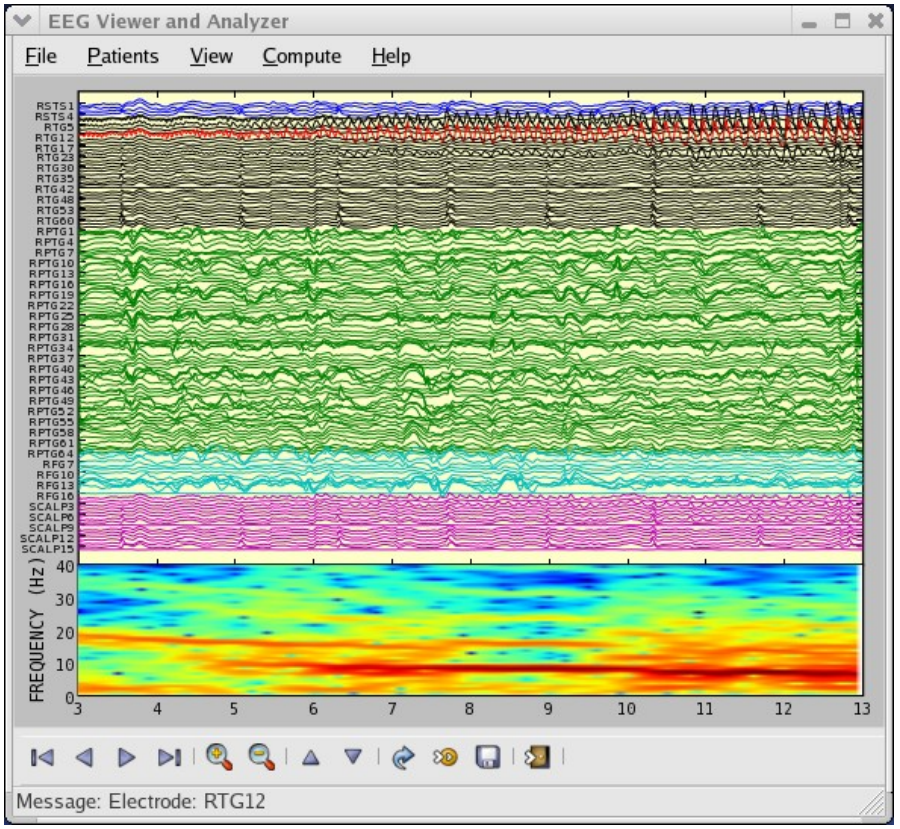
See
http://www.scipy.org/PerformancePython
for an example solving Laplace equation

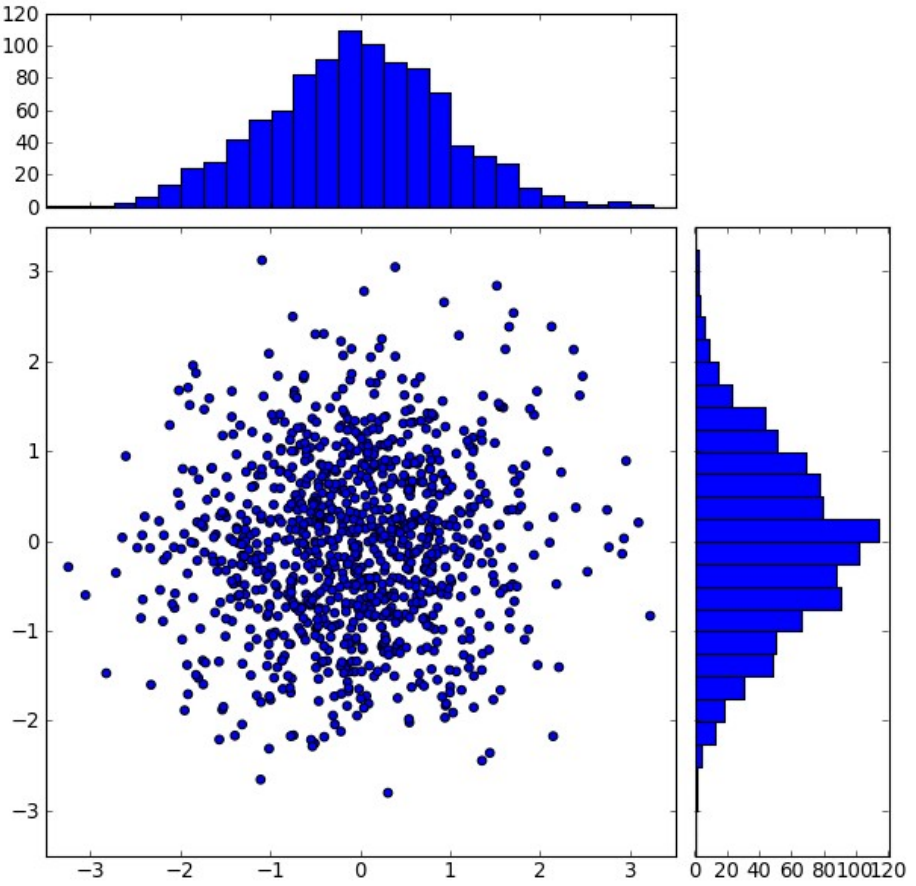
Typical Edit/Debug session

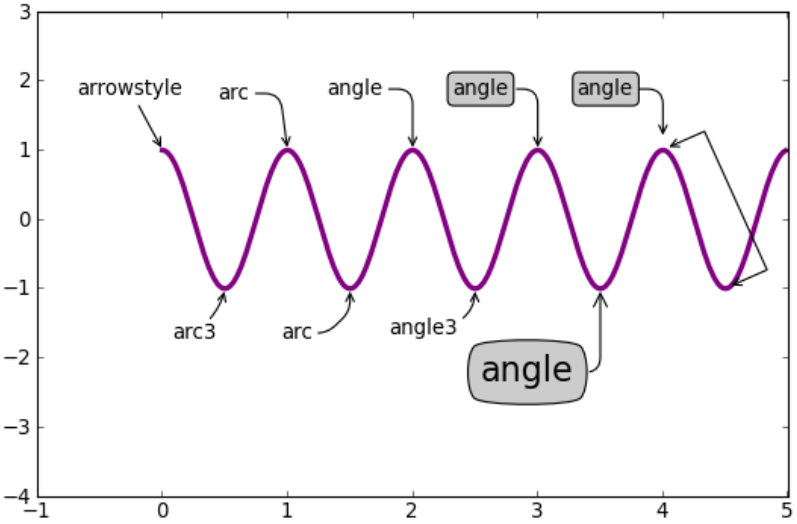
ipython>>
>> **import nbody1**
>> **nbody1.pyth2d()**
>> **a=nbody1.nbody('pyth2d.mpv')**
>> **a.bench()**
>> **reload(nbody1)**
>> **b=nbody1.nbody('pyth2d.mpv')**
>> **%time b.bench()**
CPU times: user 3.50 s, sys: 0.00 s, total: 3.50 s
Wall time: 3.51 s
>> **import nbody3 as nb**
>> **c=nb.nbody('pyth2d.mpv')**
>> **d=c.bench()**
>> **from pylab import plot**
>> **plot(d.x,d.vx,'.')**

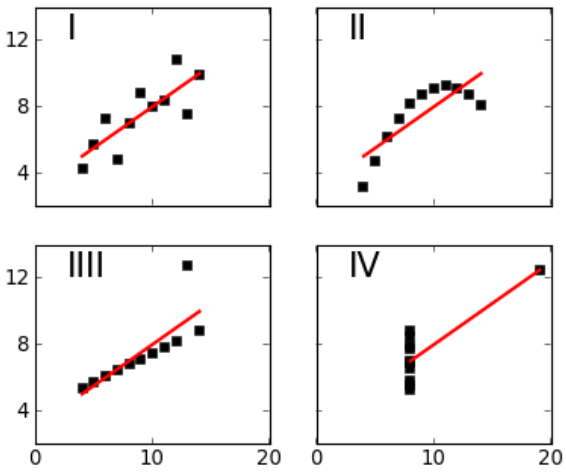
pyth2d.mpv:

```
3   1  3   0 0
4  -2 -1   0 0
5   1 -1   0 0
```


EEG Viewer and Analyzer
File
Patients
View
Compute
Help
FREQUENCY (Hz)
Message: Electrode: RTG12




arrowstyle
arc
angle
angle
angle
arc3
arc
angle3
angle


I
II
III
IV

Installation

- PiTP's **install.python** script,
  install.python prefix=/home/python
- Individual module install
  - Use $PYTHON_PATH in your local space
  - Use system's /usr/lib/python (need admin access)
- **ipython** is a module, not standard python (there is also **ipythonx**)
-

Not discussed, though mentioned

- Paraview - client / server
- VisIVO - point & grid – also client / server
- xmgrace - 2D – dynamic - python, client/server
- gnuplot, sm, xplt, pyx
- ds9 – just images and cubes, client/server
- splash – sph only
- s2plot - a 3D pgplot lookalike