# Python Tools for Astronomical Data Analysis and Visualization

## Sources for information:

**numarray:** http://www.stsci.edu/resources/software_hardware/numarray
**manual:** http://www.stsci.edu/resources/software_hardware/numarray/manualPDF

**PyFITS:** http://www.stsci.edu/resources/software_hardware/pyfits
**manual**: http://www.stsci.edu/resources/software_hardware/pyfits/Users_Manual.pdf

**matplotlib:** http://matplotlib.sourceforge.net/

**numdisplay:** http://stsdas.stsci.edu/numdisplay

**PyRAF:** http://www.stsci.edu/resources/software_hardware/pyraf

**Python:** www.python.org

Quick overview of Python: http://stsdas.stsci.edu/pyraf/doc/python_quick_tour/

**Scipy:** http://scipy.org (Hosts large number of libraries adapted for use in Python. not currently compatible with numarray, but work is underway to make it compatible and to improve documentation)

The scripts used for the canned examples can be found in the examples directory of the matplotlib source distribution, or through links on the matplotlib screenshots web page: http://matplotlib.sourceforge.net/screenshots.html

The following scripts were used:

align_text.py
log_shot.py
histogram_demo.py
barchart_demo.py
scatter_demo2.py
fill_demo.py
mathtext_demo.py
layer_images.py
axes_demo.py
anim_tk.py
finance_work2.py

**Conferences:**

PyCon: March 23-25, 2005 (Washington DC). The main Python conference
Scipy: scientific python. Annually held at Caltech the beginning of September.

## demo.py

```
# Note that this demo depends on the settings in
# .matplotlibrc changed from their default values
# See the attached example

import pyfits

# Obtain data from a FUSE dataset
fits = pyfits.open('fuse.fits')

# Get summary of FITS file contents
fits.info()


# See what the table contains
fits[1].columns.info('names')

# Access the table extension
tab = fits[1].data
flux = tab.field('flux')*10.**13
wave = tab.field('wave')
# show the dimensionality and their sizes
print flux.shape
print flux

# import plotting package
from matplotlib.matlab import *

# plot
plot(wave, flux)
# use toolbar to zoom, pan, and print
# Note that in interactive mode, what is printed is the return value
#   of the plot command. It can be assigned to a variable for future
#   manipulation.

# Add labels with Greek/math text (LaTex syntax)
xlabel(r'$\lambda  (\angstrom)$',size=13)
ylabel('Flux')
title('FUSE Spectrum')

# Overplot smoothed spectrum as dashed line
from numarray.convolve import boxcar
# Note need to reshape flux to a simple 1-d array for 1-d boxcar
sflux = boxcar(flux.flat, (100,))
hold(True)  # in order to overlay other plots
plot(wave, sflux, '--r')  # --r means dashed red line
# Overplot every 100 points in smoothed spectrum
# as circles. Array indexing produces a "sliced" array. Absence of first
# two numbers means start at beginning and go to array end. Last value
# (100) means skip every 100 points. So these subsample the original
arrays
# by a factor of 100.
subwave = wave.flat[::100] # Flatten (Otherwise need to index both
dimensions)
subflux = flux.flat[::100]
# Overplot green circles ('og') at every 100 values
plot(subwave,subflux,'og')
# Add black ('k') errorbar
errorbar(subwave, subflux, yerr=0.05*subflux, fmt='.k')
# add legend
legend(('unsmoothed', 'smoothed', 'every 100'))
text(1007, 0.5, "Hi There")
```

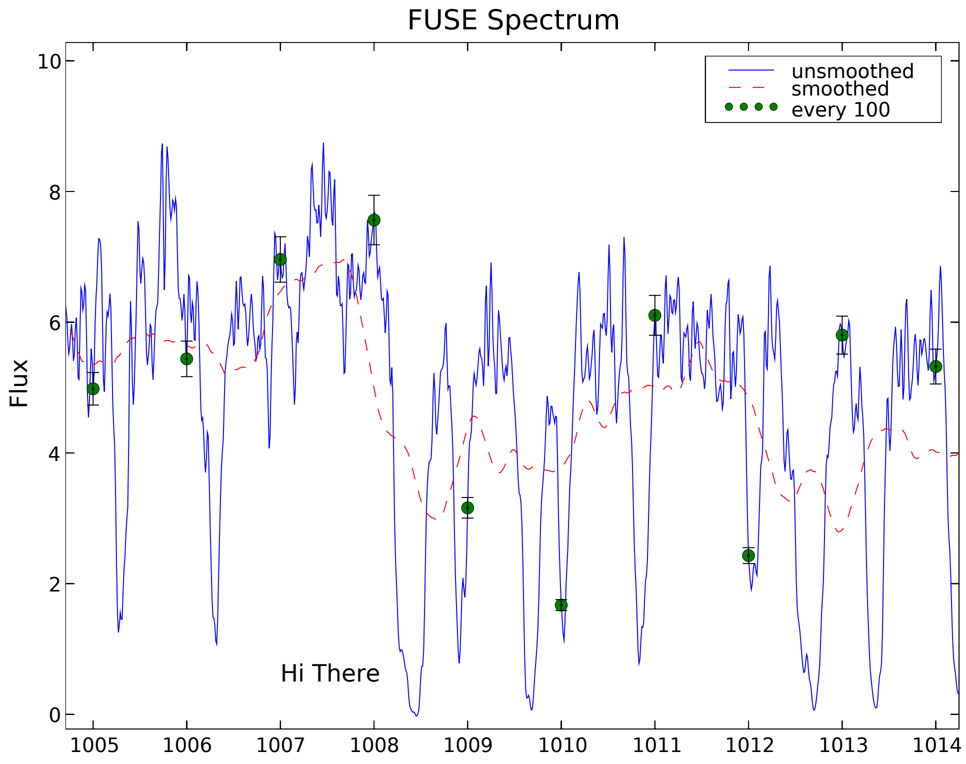
```
# save to png and postscript files
savefig('fuse.png')
savefig('fuse.ps')

# Now for images
fits = pyfits.open('chandra.fits')
#  Print image size and data type
fits.info()

im = fits[0].data.copy() # Copy since we will want to preserve
                         # original values

#  Open connection to DS9 and display image
import numdisplay as nd
nd.open()
nd.display(im)

#  Display full unsampled image using figimage
clf()
figimage(im)

# display every second pixel of 512 x 512 image
#   with max displayed value at 100
clf()  # currently figure images must be explicitly cleared
figimage(im[::2,::2],vmax=100)

#  Set pixels > 64 to 64 and redisplay
clf()
im[where(im>64)] = 64
figimage(im)

#  Display original full resampled image using imshow
clf()
im = fits[0].data
imshow(log(im+1)) # Log scaled
hold(False) # Stop overplot mode

#  Display central 512 x 512 resampled image using grayscale
imshow(log(im+1),cmap=cm.gray)

# add X and Y axis labels
xlabel('East -->')
ylabel('North -->')

#  print some header information for title
hdr = fits[0].header
for card in hdr.ascardlist()[42:52]: print card
title(hdr['object']+'\n'+hdr['observer'])

#  create new HDU, add some comments, save to file
hdu = fits[0].copy()
hdu.data = im
hdu.header.add_comment('512 x 512 clipped image of Kepler SNR')
hdu.writeto('kepler.fits')

# now write a slightly different version
hdu.data[1,1] = 999
hdu.header.update('newkwd','test') # add new keyword and value
hdu.writeto('kepler2.fits')

# compare old and new fits files
import fitsdiff
fitsdiff.fitsdiff('kepler.fits', 'kepler2.fits')
```

```
# clean up files
import os
os.remove('kepler.fits')
os.remove('kepler2.fits')
os.remove('fuse.png')
os.remove('fuse.ps')
```

```
#### MATPLOTLIBRC FORMAT
#
# This is a sample matplotlib configuration file.   It should be placed
# in your home dir (Linux and friends) or in the matplotlib data path,
# ie, where matplotlib installs its data files (fonts, etc).  On
# windows, this would be, for example, C:\Python23\share\matplotlib.
# On OSX, using the python framework, it will be (depending on your
# version number) in
#/System/Library/Frameworks/Python.framework/Versions/2.3/share/
#                                              matplotlib/.matplotlibrc

#
# By default, the installer will overwrite the existing file in the
# install path, so if you want to preserve your's, please move it to
# your HOME dir and set the environment variable if necessary.
#
# This file is best viewed in a editor which supports python mode
# syntax highlighting
#
# Blank lines, or lines starting with a comment symbol, are ignored,
# as are trailing comments.  Other lines must have the format
#
#   key : val   # optional comment
#
# Colors: for the color values below, you can either use
#  - a matplotlib color string, such as r, k, or b
#  - an rgb tuple, such as (1.0, 0.5, 0.0)
#  - a hex string, such as ff00ff  (no '#' symbol)
#  - a scalar grayscale intensity such as 0.75

#### CONFIGURATION BEGINS HERE
backend      : TkAgg    # the default backend
numerix      : numarray   # Numeric or numarray
interactive  : True     # see
http://matplotlib.sourceforge.net/interactive.html
toolbar      : toolbar2   # None | classic | toolbar2
timezone     : UTC        # a pytz timezone string, eg US/Central or
Europe/Paris

# Set the verbose flags
verbose.level  : silent     # one of silent, helpful, annoying
verbose.fileo  : sys.stdout # a log filename, sys.stdout or sys.stderr


# Where your matplotlib data lives if you installed to a non-default
# location.  This is where the matplotlib fonts, bitmaps, etc reside
#datapath : /home/jdhunter/mpldata


### LINES
# See http://matplotlib.sourceforge.net/matplotlib.lines.html for more
# information on line properties.  Note antialiased rendering looks
# better, but can be slower.  If you want fast antialiased rendering,
# use the Agg backend (or TkAgg, GTKAgg, WxAgg)
lines.linewidth   : 0.5     # line width in points
lines.linestyle   : -       # solid line
lines.color       : b       # blue
lines.marker      : None    # the default marker
lines.markerfacecolor  : b       # blue
lines.markeredgecolor  : k       # black
lines.markeredgewidth  : 0.5     # the line width around the marker
                                 #    symbol
```

```
lines.markersize  : 6            # markersize, in points
lines.antialiased : True         # render lines in antialised (no
                                 # jaggies)
lines.data_clipping : False      # Use data clipping in addition to
                                 # viewport
                                 # clipping.  Useful if you plot long
                                 # data
                                 # sets with only a fraction in the
                                 # viewport

### Patches
# Patches are graphical objects that fill 2D space, like polygons or
# circles.  See
# http://matplotlib.sourceforge.net/matplotlib.patches.html for more
# information on patch properties
patch.linewidth        : 1.0     # edge width in points
patch.facecolor   : b
patch.edgecolor   : k
patch.antialiased      : True    # render patches in antialised (no
                                  # jaggies)

### FONT
#
# font properties used by text.Text.  see
# http://matplotlib.sourceforge.net/matplotlib.fonts.html for more
# information on font properties.  The 6 font properties used for font
# matching are given below with their default values.
#
# The font.family property has five values: 'serif' (e.g. Times),
# 'sans-serif' (e.g. Helvetica), 'cursive' (e.g. Zapf-Chancery),
# 'fantasy' (e.g. Western), and 'monospace' (e.g. Courier).  Each of
# these font families has a default list of font names in decreasing
# order of priority associated with them.
#
# The font.style property has three values: normal (or roman), italic
# or oblique.  The oblique style will be used for italic, if it is not
# present.
#
# The font.variant property has two values: normal or small-caps.  For
# TrueType fonts, which are scalable fonts, small-caps is equivalent
# to using a font size of 'smaller', or about 83% of the current font
# size.
#
# The font.weight property has effectively 13 values: normal, bold,
# bolder, lighter, 100, 200, 300, ..., 900.  Normal is the same as
# 400, and bold is 700.  bolder and lighter are relative values with
# respect to the current weight.
#
# The font.stretch property has 11 values: ultra-condensed,
# extra-condensed, condensed, semi-condensed, normal, semi-expanded,
# expanded, extra-expanded, ultra-expanded, wider, and narrower.  This
# property is not currently implemented.
#
# The font.size property has 11 values: xx-small, x-small, small,
# medium, large, x-large, xx-large, larger, smaller, length (such as
# 12pt), and percentage.  larger and smaller are relative values.
# percentage is not yet implemented.
#
font.family         : sans-serif
font.style          : normal
font.variant        : normal
font.weight         : medium
font.stretch        : normal
font.size           : medium
font.serif          : New Century Schoolbook, Century Schoolbook L,
Utopia, ITC Bookman, Bookman, Bitstream Vera Serif, Nimbus Roman No9 L,
Times New Roman, Times, Palatino, Charter, serif
font.sans-serif     : Lucida Grande, Verdana, Geneva, Lucida, Bitstream
Vera Sans, Arial, Helvetica, sans-serif
font.cursive        : Apple Chancery, Textile, Zapf Chancery, Sand,
cursive
font.fantasy        : Comic Sans MS, Chicago, Charcoal, Impact, Western,
fantasy
font.monospace      : Andale Mono, Bitstream Vera Sans Mono, Nimbus Mono
L, Courier New, Courier, Fixed, Terminal, monospace


### TEXT
# text properties used by text.Text.  See
# http://matplotlib.sourceforge.net/matplotlib.Text.html for more
# information on text properties
text.color          : k       # black

### AXES
# default face and edge color, default tick sizes,
# default fontsizes for ticklabels, and so on
axes.hold           : False    # whether to clear the axes by default
                               # on
axes.facecolor      : w       # background color; white
axes.edgecolor      : k       # edge color; black
axes.linewidth      : 1.0     # edge linewidth
axes.grid           : False   # display grid or not
axes.titlesize      : 14      # fontsize of the axes title
axes.labelsize      : 12      # fontsize of the x any y labels
axes.labelcolor     : k       # black

### TICKS
tick.major.size     : 4      # major tick size in points
tick.minor.size     : 2      # minor tick size in points
tick.major.pad      : 4      # distance to major tick label in points
tick.minor.pad      : 4      # distance to the minor tick label in
                             # points
tick.color          : k      # color of the tick labels
tick.labelsize      : 10     # fontsize of the tick labels

### Grids
grid.color       :   k       # grid color
grid.linestyle   :   :       # dotted
grid.linewidth   :   0.5     # in points

### FIGURE
figure.figsize   : 8, 6    # figure size in inches
figure.dpi       : 80      # figure dots per inch
figure.facecolor : 0.75    # figure facecolor; 0.75 is scalar gray
figure.edgecolor : w       # figure edgecolor; w is white

### images
image.aspect : free             # free | preserve
image.interpolation  : bilinear  # see help(imshow) for options
image.cmap   : jet              # gray | jet
image.lut    : 256              # the size of the colormap lookup
                                # table
image.origin : lower            # lower | upper

### SAVING FIGURES
# the default savefig params can be different for the GUI backends.
# Eg, you may want a higher resolution, or to make the figure
# background white
```

```
savefig.dpi       : 100      # figure dots per inch
savefig.facecolor : w        # figure facecolor; 0.75 is scalar gray
savefig.edgecolor : w        # figure edgecolor; w is white

tk.window_focus   : False    # Maintain shell focus for TkAgg
```


FUSE Spectrum
Flux
unsmoothed
smoothed
every 100
Hi There
0
2
4
6
8
10
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014

KEPLER SNR
DR STEPHEN HOLT