

Using Python for fun and profit

~~Why you shouldn't be using IDL~~

Why python is Awesome.

Hard Code

Easier Code

Pure GUI

Game

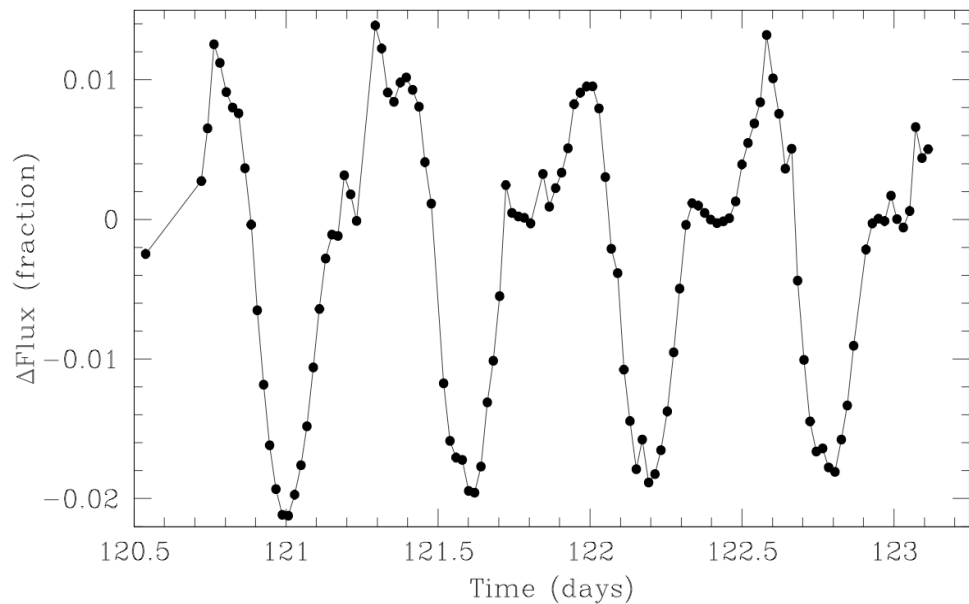
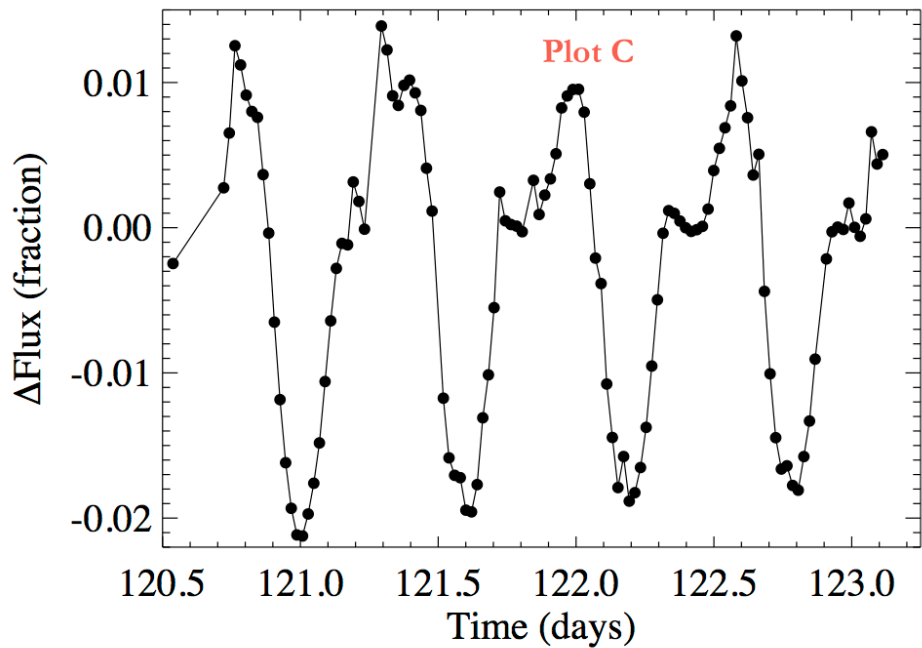
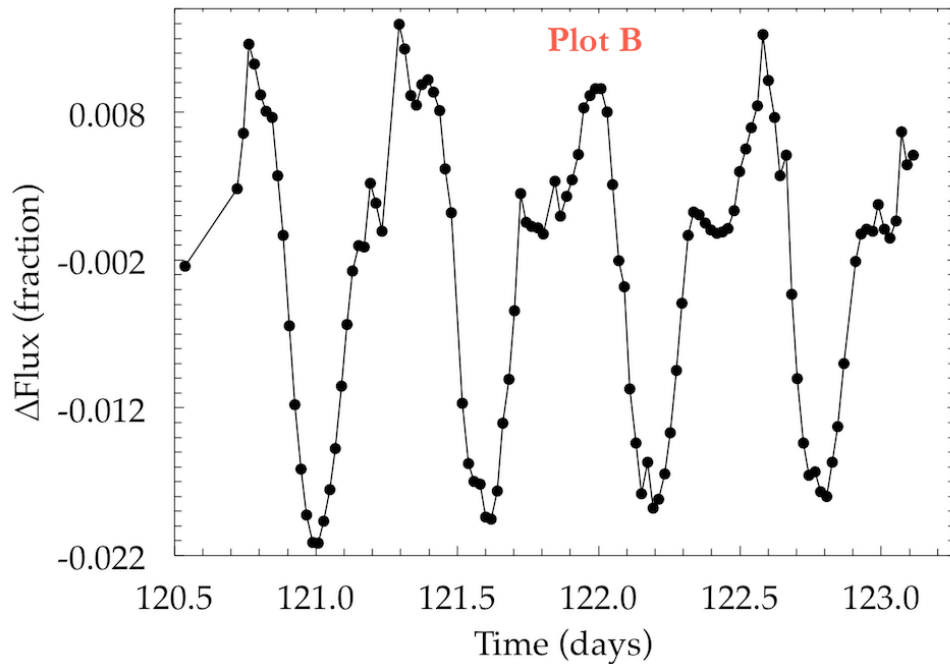
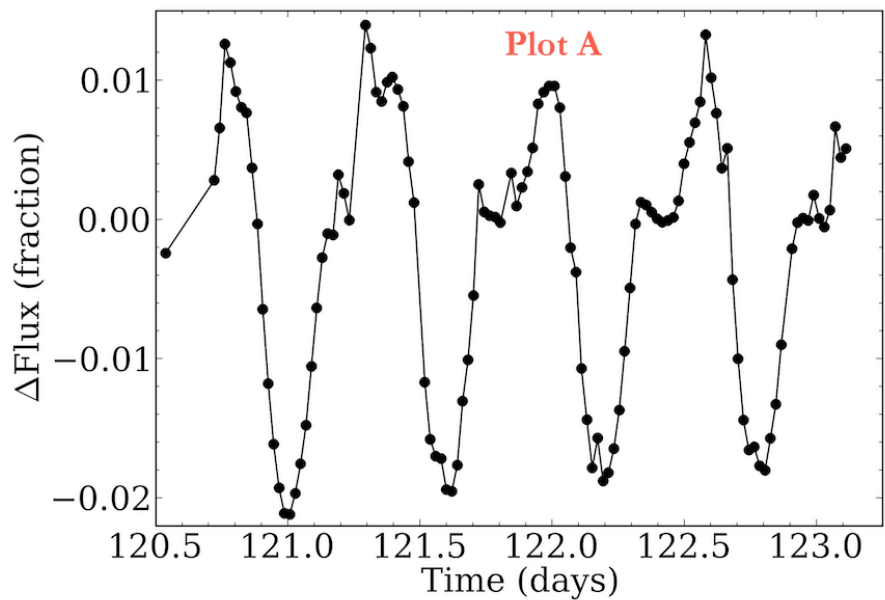
C++
FORTRAN

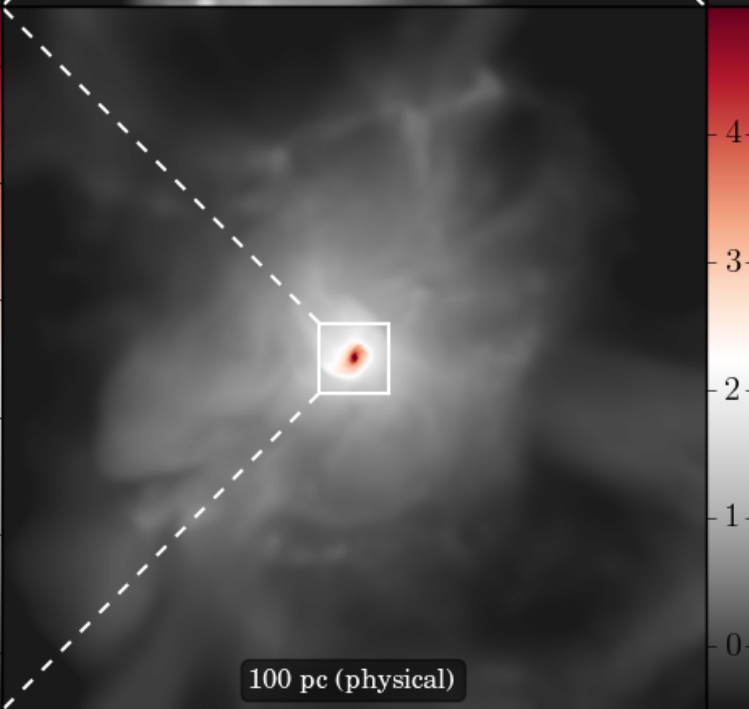
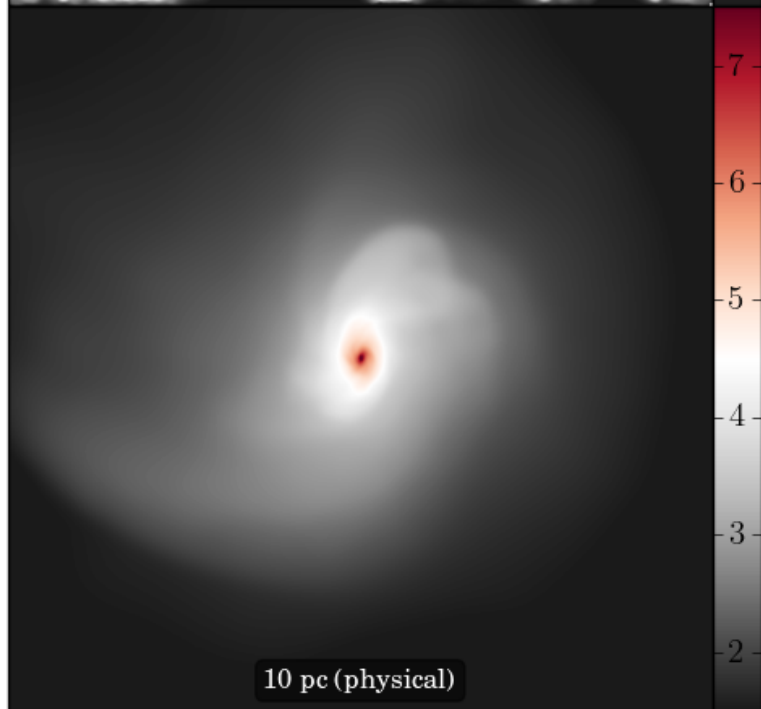
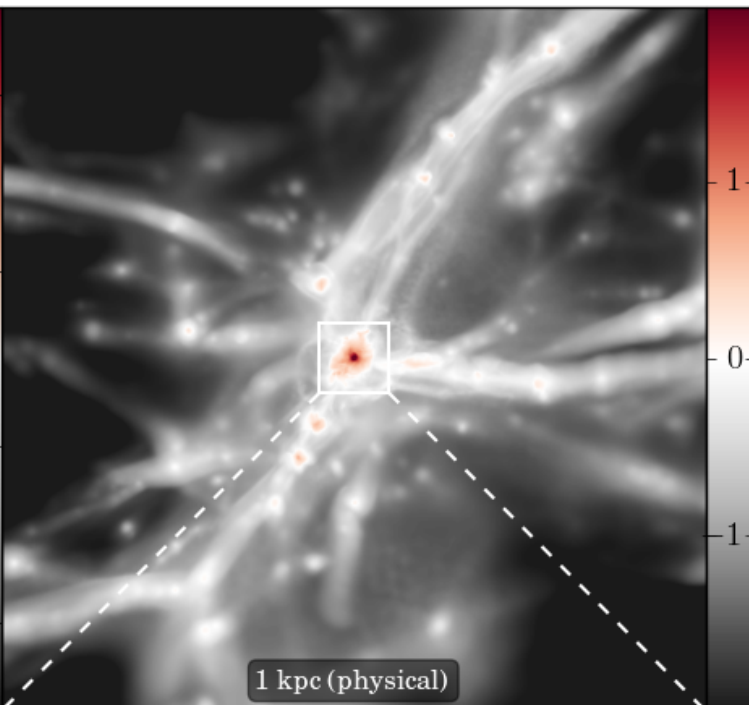
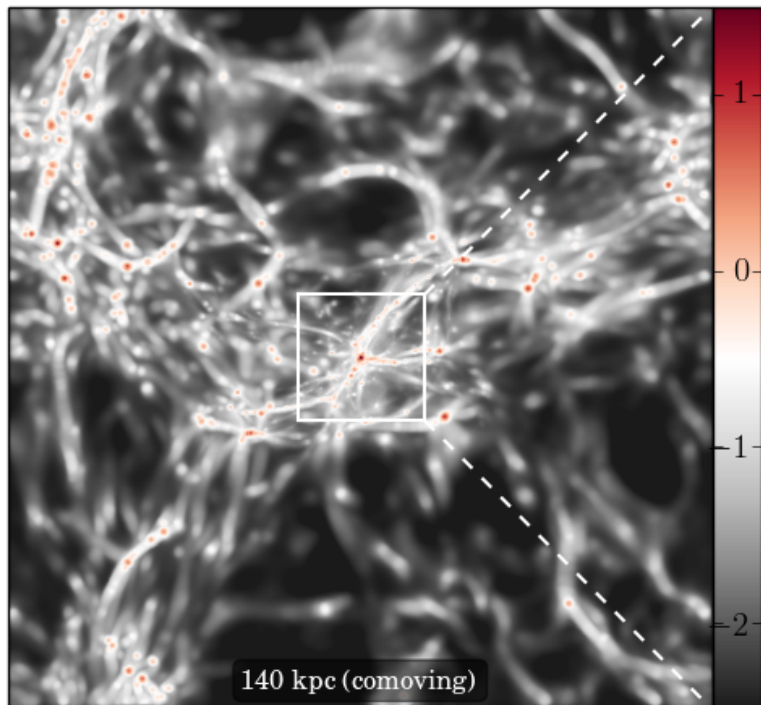


Wolfram
Mathematica



Where do *you* need to be
to do astrophysics research?





Log Number Density [cm^{-3}]

Log Number Density [cm^{-3}]

```
In [ ]: from mpl_toolkits.axes_grid1 import ImageGrid

scales = ['140 kpc (comoving)', '1 kpc (physical)', '10 pc (physical)', '100 pc (physical)']
ratio = [.1788, .1, None, .1]
zoom = ['right', 'down', None, 'left']
clims = [(-2.5,1.5), (-2.,2.), (1.5,7.5), (-0.5,5.)]
ticks = [(-2,-1,0,1), (-1,0,1), (2,3,4,5,6,7), (0,1,2,3,4)]
cpad = [-17, -17, -15, -16]
clabel = [False, True, False, True]
bbox_props = dict(boxstyle="round", fc="k", ec="k", alpha=0.5)
zc = 'w'
zls = '--'
zlw = 1.5

fig = plt.figure(1, (12., 12.), dpi=600)
grid = ImageGrid(fig, 111, # similar to subplot(111)
                 nrows_ncols = (2, 2), # creates 2x2 grid of axes
                 axes_pad=0.0, # pad between axes in inch.
                 cbar_mode = 'each', cbar_size='7%', cbar_pad=0.
                 )

for i in range(4):
    x = imzoom[i][0]
    y = imzoom[i][1]
    im = imzoom[i][2]
    ax = grid[i]
    img = ax.imshow(im, cmap=plt.cm.RdGy_r, origin='lower')
    ax.xaxis.set_visible(False)
    ax.yaxis.set_visible(False)
    img.set_clim(clims[i])

    cb = plt.colorbar(img, cax=grid.cbar_axes[i])
    cb.set_ticks(ticks[i])
    cb.ax.tick_params(left='on', pad=cpad[i],
                     labelsize=15, labelcolor='k', labelleft='on', labelright='off')
    if clabel[i]: cb.set_label('Log Number Density [cm-3]{-3}{-3}')

    ax.text(0.5, 0.025, scales[i], color='w', ha='center', va='bottom', size=12,
           transform=grid[i].transAxes, bbox=bbox_props)

    if ratio[i]:
        axmin, axmax = ax.get_xlim()
        axlength = axmax - axmin
        mid = axlength/2
        s = ratio[i] * axlength
        s00 = [mid - s/2, mid - s/2]
        s01 = [mid - s/2, mid + s/2]
        s11 = [mid + s/2, mid + s/2]
        ax.add_line(plt.Line2D(s00, s01, c=zc, lw=zlw))
        ax.add_line(plt.Line2D(s11, s01, c=zc, lw=zlw))
        ax.add_line(plt.Line2D(s01, s00, c=zc, lw=zlw))
        ax.add_line(plt.Line2D(s01, s11, c=zc, lw=zlw))
        if zoom[i] == 'right':
            ax.add_line(plt.Line2D([mid+s/2, axmax], [mid+s/2, axmax], c=zc, lw=zlw, ls=zls))
            ax.add_line(plt.Line2D([mid+s/2, axmax], [mid-s/2, axmin], c=zc, lw=zlw, ls=zls))
        elif zoom[i] == 'down':
            ax.add_line(plt.Line2D([mid-s/2, axmin], [mid-s/2, axmin], c=zc, lw=zlw, ls=zls))
            ax.add_line(plt.Line2D([mid+s/2, axmax], [mid-s/2, axmin], c=zc, lw=zlw, ls=zls))
        elif zoom[i] == 'left':
            ax.add_line(plt.Line2D([mid-s/2, axmin], [mid+s/2, axmax], c=zc, lw=zlw, ls=zls))
            ax.add_line(plt.Line2D([mid-s/2, axmin], [mid-s/2, axmin], c=zc, lw=zlw, ls=zls))

plt.show()
fig.savefig('figures/structure.png', bbox_inches='tight', dpi=100)
```

Reasons NOT to use Python

- I already know how to do this in IDL/SM/Excel/etc
- Legacy Code
- Legacy Professors
- Better resources for getting help with other languages
- Better code libraries

Reasons NOT to use Python

- I already know how to do this in IDL/SM/Excel/etc
- Legacy Code
- Legacy Professors
- Better resources for getting help with other languages
- Better code libraries

2008

Reasons NOT to use Python

- I already know how to do this in IDL/SM/Excel/etc
- Legacy Code
- Legacy Professors
- ~~Better resources for getting help with other languages~~
- ~~Better code libraries~~

You want it – Python has it (probably)



NumPy
Base N-dimensional
array package



SciPy library
Fundamental library
for scientific
computing



Matplotlib
Comprehensive 2D
Plotting

IP[y]:
IPython

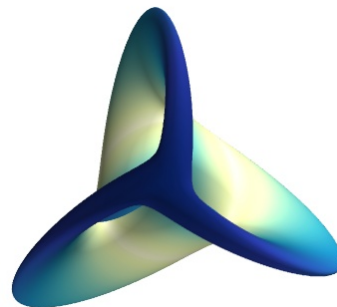
IPython
Enhanced
Interactive Console



Sympy
Symbolic
mathematics



pandas
Data structures &
analysis



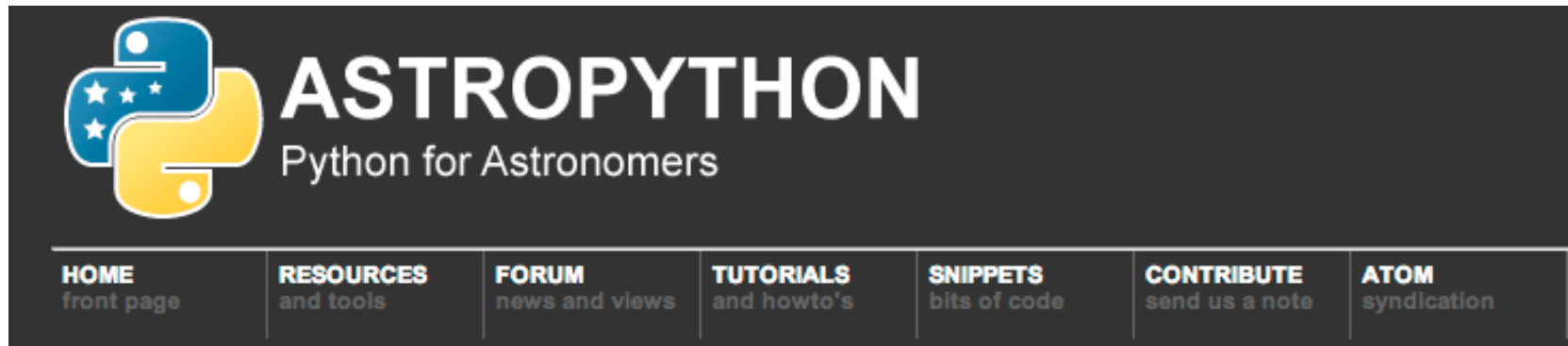
Mayavi – 3D visualization



H5py and PyTables:
hdf5 made easy

But what about Astronomy?

- An entire ecosystem of python tools specifically for python exists.



<http://www.astropython.org/>

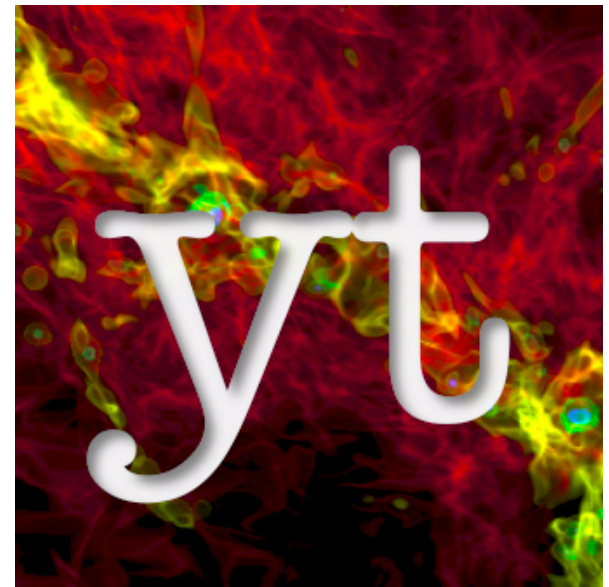
But what about Astronomy?



PyRaf – a python wrapper for IRAF



PyFITS – interface for FITS formatted images



Still can't find what you need?

- Perhaps Python's greatest strength is how easy it is to extend
- Works great as 'glue' to piece together your analysis pipeline
- Use Cython; `scipy.weave`; `f2py` to speed up computationally intensive parts of your code



- Parallel processing is possible

IPython Notebooks

- interactive, repeatable analysis
- Use as a living research journal
- quickly and easily share your analysis with others



The screenshot shows a web browser window with two tabs: "IPy IPython Dashboard" and "IPy 00_notebook_tour". The active tab displays the IPython Notebook interface for "00_notebook_tour", last saved on Jun 13 10:14 AM. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with various icons. The main content area features a heading "Rich displays: include anything a browser can show" and a note about the display_protocol notebook. Below this, the "Images" section shows an interactive code cell:

```
In [1]: from IPython.core.display import Image
        Image(filename='../source_static/logo.png')
```

The output of this cell is a rich display:

```
Out[1]: Image(self, data=None, url=None, filename=None, format='png', embed=None)
        Create a display an PNG/JPEG image given raw data.

        When this object is returned by an expression or passed to the
        display function, it will result in the image being displayed
        in the frontend.

        An image
        Parameters
        -----
        data : unicode, str or bytes
              The raw data or a URL to download the data from.
        url  : unicode
              A URL to download the data from.
```

Below the code cell, the Python logo is displayed. The "SVG images" section follows, with a code cell:

```
In [3]: from IPython.core.display import SVG
        SVG(filename='python-logo.svg')
```

The output of this cell is a rich display of the Python logo:

```
Out[3]: 
```