SunPy: Python for Solar Physics Data Analysis

V. Keith Hughitt[1], S. Christe[2], J. Ireland[1], A. Shih[2], F. Mayer[3], M. D. Earnshaw[4], C. Young[1], D. Perez-Suarez[5], R. Schwartz[6]

[1]ADNET Systems, NASA GSFC, [2]NASA GSFC, [3]Technische Universitat Wien, Austria, [4]Blackett Laboratory, Imperial College, UK,
[5]Trinity College Dublin, Ireland, [6]The Catholic University of America, NASA GSFC

Overview

SunPy is a Python library for solar physics data analysis. This poster will discuss some of the latest SunPy developments along with future plans for development.

SunPy is an open source effort and all of our code is freely available at:

http://sunpy.org

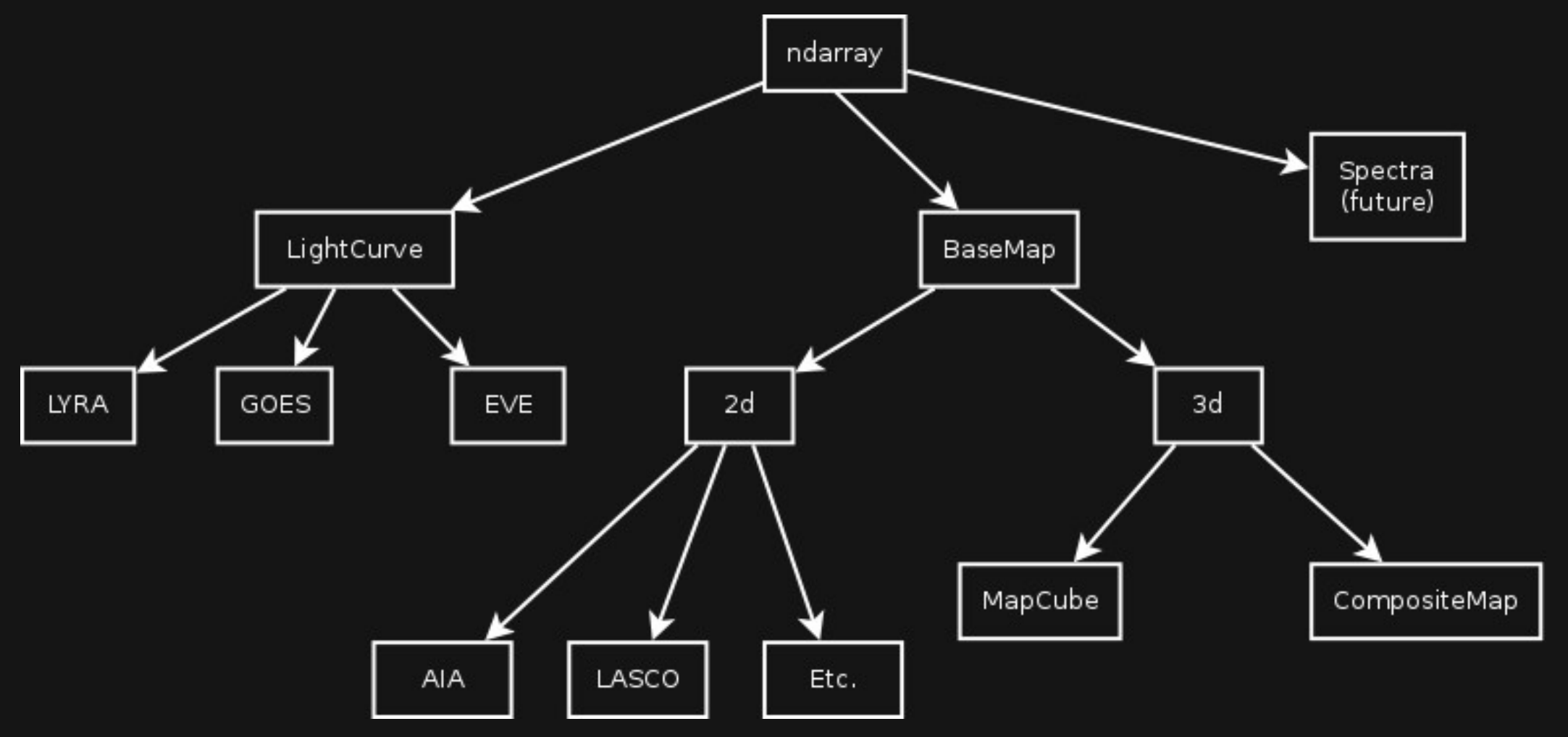
Why Python?

Each programming environment has it's own advantages and disadvantages. Python was chosen because it excels above other languages for scientific applications such as SunPy. Some benefits include:

•Easy
•Free and open-source
•Large following in the scientific community (numerous libraries already available)

Map Architecture

One of the core data structures in SunPy is the Map – a spatially-aware data array (e.g. an image). Below is a depiction of the basic architecture of the Map class and it's sub-classes, all of which inherit from numpy.ndarray..



The related nature of the SunPy classes makes it easy to convert from one to the other. For example, a MapCube behaves like an array of Maps; calling 'cube[0]' for a MapCube instance named 'cube' returns a Map instance.

Examples

Plotting a SunPy map

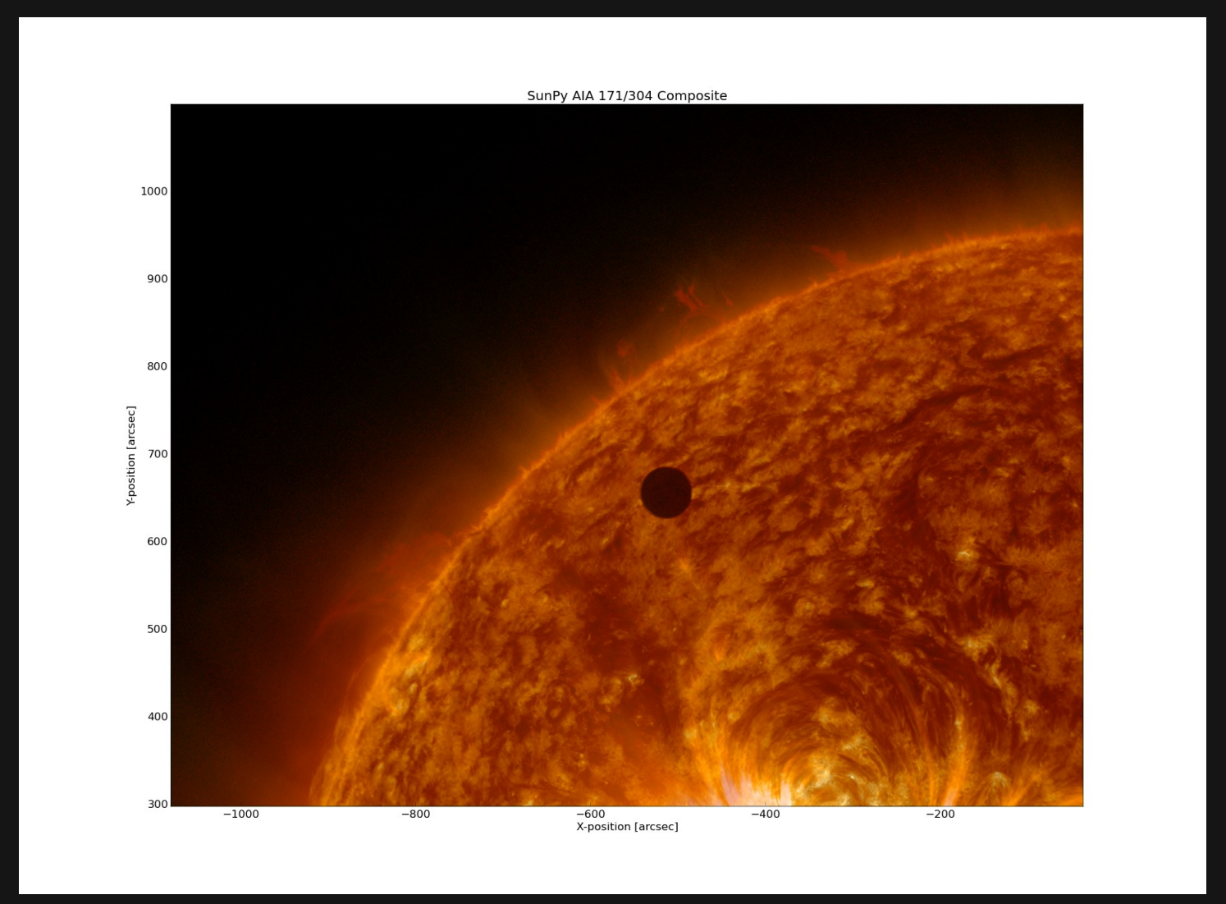


Figure 2: Venus transit as seen by SDO's AIA instrument.

One of the major focuses of development for SunPy has been to provide simple and powerful plotting capabilities. SunPy makes it easy to composite both single and composite maps. SunPy map objects are inspired by and work similarly to SSW map objects. By using Matplotlib for the underlying plotting, users gain access to the rich functionality provided by Matplotlib and are able to use any of the plot commands it supports.

```
# Example: composite image plot
import sunpy
maps = sunpy.make_map('2012_06_05*.jp2')
maps.set_alpha(1, 0.5)
maps.show(title='SunPy AIA 171/304 Composite')
```

Plotting GOES light-curve data

One of the major focuses of development for SunPy has been to provide simple and powerful plotting capabilities. SunPy makes it easy to composite both single and composite maps. SunPy map objects are inspired by and work similarly to SSW map objects. By using Matplotlib for the underlying plotting, users gain access to the rich functionality provided by Matplotlib and are able to use any of the plot commands it supports.

```
# Example: downloading and plotting GOES data
from sunpy.instr import goes
fp = goes.get_file('2012/06/05','2012/06/06')
data = goes.parse_file(fp[0])
goes.show(data.get('time_tag'),
          data.get('A_FLUX'),
          data.get('B_FLUX'))
```

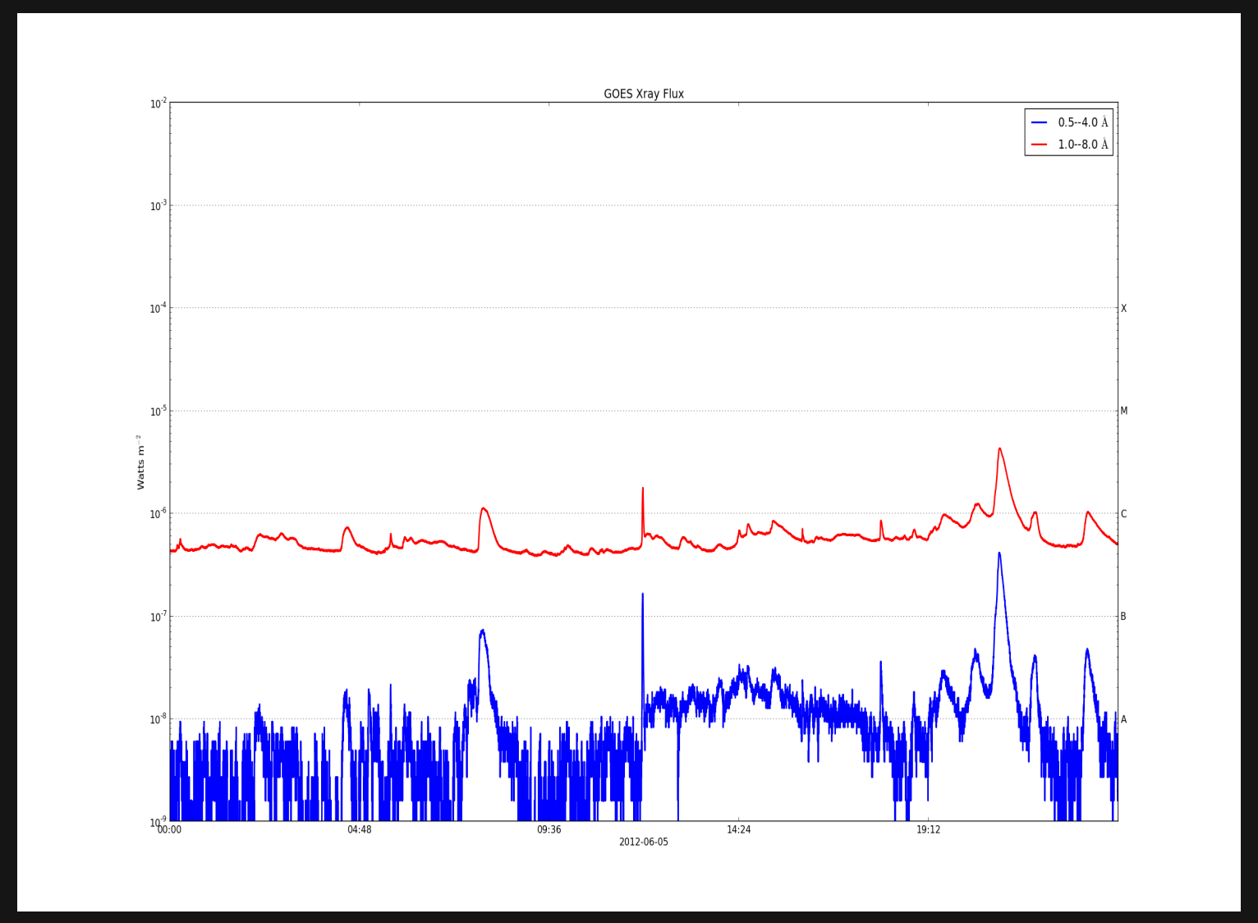
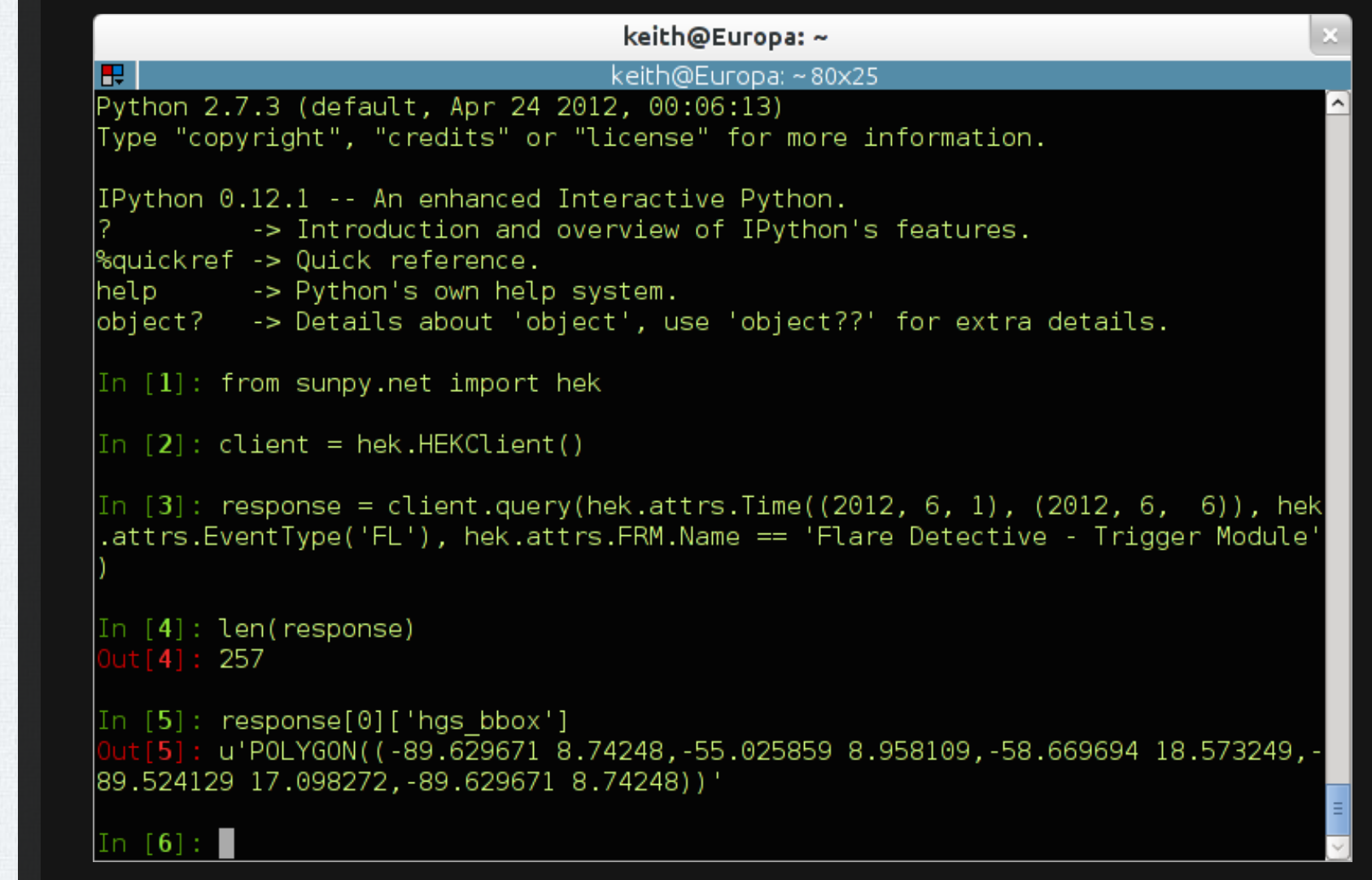


Figure 3: GOES light-curve from June 05, 2012.

Accessing feature and event metadata from the HEK



```
keith@Europa: ~
keith@Europa: ~ 80x25
Python 2.7.3 (default, Apr 24 2012, 00:06:13)
Type "copyright", "credits" or "license" for more information.

IPython 0.12.1 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: from sunpy.net import hek

In [2]: client = hek.HEKClient()

In [3]: response = client.query(hek.attrs.Time((2012, 6, 1), (2012, 6,  6)), hek
.attrs.EventType('FL'), hek.attrs.FRM.Name == 'Flare Detective - Trigger Module'
)

In [4]: len(response)
Out[4]: 257

In [5]: response[0]['hgs_bbox']
Out[5]: u'POLYGON((-89.629671 8.74248,-55.025859 8.958109,-58.669694 18.573249,-
89.524129 17.098272,-89.629671 8.74248))'

In [6]: 
```

SunPy offers support for interacting with various online data services: users can query and download data from the Virtual Solar Observatory (VSO), create movies and screenshots using Helioviewer.org, or search for feature and event detections using the Heliophysics Event Knowledgebase (HEK).

Aside from making queries similar to those that can be made using the IDL or web interfaces of these services, SunPy can also perform complex queries not offered at the traditional interfaces. Below is an example of an HEK query for flares detected between 2012/06/01 and 2012/06/06 using the "Flare Detective - Trigger Module" feature recognition method (FRM). The result is a list of event structures, each of which corresponds to a single detection.

```
# Example: Finding flares the occurred for a given time using the HEK
from sunpy.net import hek
client = hek.HEKClient()
date = hek.attrs.Time((2012, 6, 1), (2012, 6,  6))
response = client.query(date, hek.attrs.EventType('FL'),
hek.attrs.FRM.Name == 'Flare Detective - Trigger Module')
len(response)
response[0]['hgs_bbox']
```

EIT Wave research using SunPy

For a real-world example of SunPy being used to do research, see the poster by Ireland et. al., *"Automatic Detection and Characterization of EIT Waves Observed by AIA Data"*; SPD poster session 201. Solar & Stellar, Tuesday, June 12, 2012 9:00 AM - 6:30 PM.

Modules

SunPy includes a number of different modules, each with their own focus. Below is a brief overview of each of the major modules .

gui	Graphical interface components including a Plotman-like visualization tool based based on Matplotlib.
image	image processing routines.
inst	Instrument-specific functionality.
io	File input/output functionality.
map	N-dimensional matrix used to represent solar data
net	Network functionality (VSO, HEK, Helioviewer, etc)
sun	Solar constants.
wcs	World Coordinate System transformations.

What's Next?

SunPy was started little over a year ago, but already much progress has been made. Initial efforts have focused on core functionality relating to acquiring and reading in data and plotting. Future efforts will continue to refine this work, and also add new features such as:

•Improved support for light-curve data structures using the Pandas time-series library.

•Support for working with spectral data types including GOES, EVE and LYRA data (initial support for all three of these already exists)

•Performance improvements for working with large data sets.

Feedback?

Any comments or suggestions would be appreciated.

The author can be contacted at:

keith.hughitt@nasa.gov