# SunPy (0.3): Python for Solar Physics Data Analysis

S. Christe[1], V. Keith Hughitt[1], J. Ireland[1], F. Mayer[3], D. Perez-Suarez[4], A. Shih[2], S. Mumford[6], R. Hewett[7], M. D. Earnshaw[5], SunPy Developers

[1]NASA GSFC, [2]ADNET Systems, NASA GSFC, [3]Technische Universitat Wien, Austria, [4]Trinity College Dublin, Ireland, [5]Blackett Laboratory, Imperial College, UK, [6]University of Sheffield, UK, MIT[7]

[6]The Catholic University of America, NASA GSFC

## Overview

The SunPy project is an effort to create an open-source software library for solar physics using the Python programming language.

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

**http://sunpy.org**

SunPy is in active development and has just released version 0.3! To browse the functionality currently available checkout the live documentation at

**http://sunpy.readthedocs.org/**

## Why Python?

Python is an interpreted, interactive, free and open-source object-oriented language which is well suited for scientific applications. It incorporates modules, exceptions, dynamic typing, very high level dynamic data types, and classes. Some benefits include:

- Free as in beer!
- *Many* different sources of help available – tutorials, code snippets, forums. Used by many different communities.
- Large following in the scientific community and access to numerous multi-disciplinary libraries (e.g. AstroPy, SpacePy).

## Datatype Architecture

Although there are many different data sources in solar physics, there are many fewer *datatypes.* For example, AIA and XRT both produce images of the Sun. In SunPy, data from these instruments are examples of *maps* - spatially aware images of the Sun. Objects that handle data from AIA and XRT have the same core map functionality since they are the same *datatype*. However, the AIA and XRT objects also contain functionality specific to their instrument. The *datatype* architecture ensures a more integrated data analysis environment.

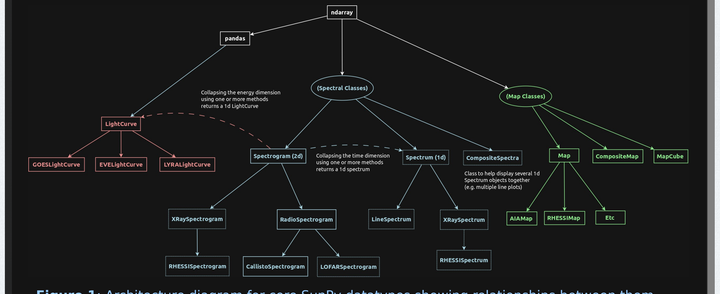


**Figure 1**: Architecture diagram for core SunPy datatypes showing relationships between them. Boxes with dashed outlines indicated planned classes.

## Examples

### (Over)Plotting a SunPy map

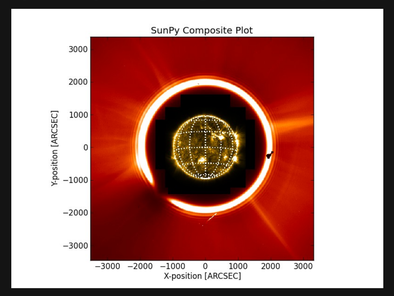


**Figure 2**: SDO's AIA image with a context LASCO image.

One of the major focuses of development for SunPy has been to provide simple and powerful plotting capabilities. 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. We provide an example fits file for the following commands but SunPy can read in fits files from any of th following instruments; SDO/AIA, Yohkoh/SXT, SOHO/EIT, SOHO/LASCO, SOHO/MDI, STEREO/EUVI, STEREO/COR PROBA2/SWAP, and Hinode/XRT.

```
import sunpy
import matplotlib.pyplot as plt
aia = sunpy.Map(sunpy.AIA_171_IMAGE)
Lasco = sunpy.Map('22121375.fts')
cmap = sunpy.map.CompositeMap([lasco, aia])
cmap.plot()
cmap.draw_grid(index=1, grid_spacing=30)
plt.show()
```

### Plotting GOES light-curve data

SunPy is currently capable of working with several types of light-curve data including GOES, EVE, and LYRA. Files can be read in locally, or if dates are provided SunPy will first download the data corresponding to the specified date range and then load the data in.

```
# Example: plotting GOES data
from sunpy.lightcurve import GOESLightCurve
from sunpy.time import TimeRange
goes = GOESLightCurve.create(
    TimeRange('2012/08/07', '2012/08/08'))
goes.peek()
```

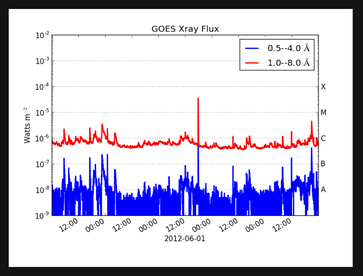


**Figure 3**: GOES light-curve from July 8, 2012

### Querying the VSO

SunPy offers a powerful interface to the VSO to search for and download data. We provide twq interfaces/ A legacy interface which is similar to the IDL vso query tool.

```
from sunpy.net import vso
Client = vso.VSOClient()
Qr = client.query_legacy(tstart='2001/01/01 00:00', tend='2001/01/02 01:00',
instrument='EIT')
res = client.get(qr, path='~/Desktop/{file}.fits')
```

As well as a more powerful interface for more complex queries. This query system allows for full use of logical operators. So one can search for times with both AIA and STEREO images or search a time interval for observations at either of two particular wavelengths.

```
qr = client.query(vso.attrs.Time('2001/1/1', '2001/1/2'), vso.attrs.Instrument('eit'))
qr = client.query(vso.attrs.Time('2001/1/1', '2001/1/2'), vso.attrs.Instrument('eit') |
vso.attrs.Instrument('mdi'))
qr = client.query(vso.attrs.Time('2001/1/1', '2001/1/2'), vso.attrs.Instrument('eit'),
vso.attrs.Wave(171,171) )
```

### Accessing feature and event metadata from the HEK

```
# Example: Querying HEK for flares
from sunpy.net import hek
client = hek.HEKClient()
date = hek.attrs.Time(
    (2012, 6, 1), (2012, 6,  6)
)
result = client.query(date,
hek.attrs.EventType('FL'),
  ( (hek.attrs.Event.Coord1 > 50) and
    (hek.attrs.FL.PeakFlux > 1000.0)    )
hek.attrs.FRM.Name == 'Flare Detective - Trigger
Module')
```

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).

SunPy can also perform complex queries not offered by the traditional interfaces. The code on the left 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), with a peak flux greater than 1000.0, and west of 50 arcseconds.

## 0.3 versus 0.2

SunPy 0.3 includes major improvements in the following areas

- All sunpy datatype now hold the raw data as an object variables making it clearer and easier to access (e.g. map.data, lc.data)
- Major improvements to the backend for opening files into datatypes.
- Major improvements in documentation.
- Lots of bugs squashed!

**Project Statistics**

- 2360 code commits
- 18,168 lines (estimated 3 person-years, $160k)
  - 13,042 lines of code
  - 5,126 lines of comments (lots of comments!)
- 27 contributors

- Project Growth (compared to last year)
  - +33% increase in number of commits
  - +109% increase in number of contributors!

## What's Next?

SunPy has been funded by the Google Summer of Code 2013! Two students , Simon Liedtke & Michael Malocha will be working on the following projects

- A database system for indexing and searching solar data files on your computer. This system will link automatically update itself when the sunpy vso module is used to download files.
- Adding better integration for Heliophysics Event Knowledge base (HEK) and new integration of HELIO. This will include querying for events, but also other capabilities like overploting the events’ positions/contours onto maps.

## Feedback?

Any comments, suggestions, code, would be appreciated.

email: steven.d.christe@nasa.gov
User group: sunpy@googlegroups.com
Code: https://github.com/sunpy/sunpy