
Astropy

*A community developed core
package for Astronomy in Python*

Axel Donath
MPIK Heidelberg

Herbsttagung der Deutschen Astronomischen
Gesellschaft, Bamberg 2014



What is Astropy?

How the Astropy project started

June 9th 2011 on the Astropy mailing list ...

- [\[AstroPy\] PyAstronomy](#) *Stefan Czesla*
 - [\[AstroPy\] Proliferating py-astro-libs](#) *Marshall Perrin*
 - [\[AstroPy\] Proliferating py-astro-libs](#) *Wolfgang Kerzendorf*

On Jun 9, 2011, at 12:54 PM, Stefan Czesla wrote:
Dear all,

we would like to let you know about our recent release of a -- hopefully -- useful contribution to Python's astronomy community, namely, our PyAstronomy package (yes, there have been more inspired names...). It consists of

On 10/06/11 8:25 AM, Marshall Perrin wrote:

Hopefully without sounding too critical of you in particular, I'm going to ask: do we as a community really need /yet another/ separate python library for astronomy and yet another attempt at building a core set of routines ported from the IDL library?

The problem


- astLib
- astrolib
- astropysics
- pyephem
- ephempy
- pyast
- pyastro
- apwlib
- chiantipy
- pymidas
- pandora
- pyspec
- pyraf
- atpy
- aplpy
- pynovas
- cosmopy
- cosmology
- kapteyn
- cosmics.py
- pyguide
- astrogui
- pyastronomy
- pywcs
- pyfits
- sunpy
- ...more?


The solution

- 🌀 Start the Astropy project ... a community effort to develop a **core package for Astronomy in Python** and **foster interoperability** between Python astronomy packages.
- 🌀 **Astropy core package** ... useful for most astronomers.
- 🌀 **Astropy affiliated packages** ... for more specialised astronomy applications. Leverage Astropy infrastructure (packaging, testing, documentation) by using a package template.
- 🌀 Astropy project coordinators:
 - Perry Greenfield, STScI
 - Thomas Robitaille, MPIA
 - Erik Tollerud, Yale

3 years later ...

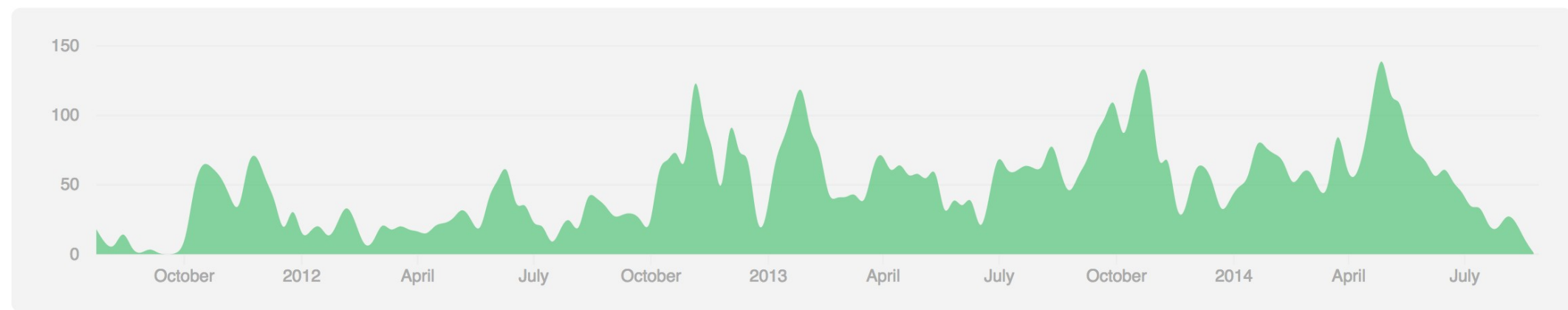
...it has worked out very well!

 [GitHub, Inc. \[US\]](https://github.com/showcases/science) <https://github.com/showcases/science>

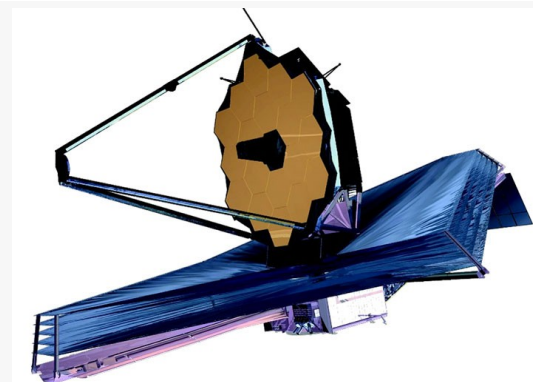
 **astropy / astropy**

Astropy is a community effort in the astrophysics community to develop a single core package for Astronomy in Python. Three years after the [first commit](#) more than [75 researchers](#) have contributed over [1600 pull requests](#) making this one of the most heavily used open source projects in astronomy.

[Python](#) ★ 587 stars [starred by 1 person you know](#)



STScI has gotten substantial funding (several FTEs for 4+ years) to develop generic data analysis tools for Python to support **JWST** data analysis needs ... most effort will go to Astropy and affiliated packages.



Astropy core package

Astropy core status

- 🌀 3 major public releases (first release February 2013)
- 🌀 Latest stable version: 0.4.1 (released August 2014)
- 🌀 **Astropy 0.2 paper** in Astronomy & Astrophysics journal
(a nice sign that an astronomy science journal accepts a paper only presenting a software, not a science application)
- 🌀 GSoC students (2 students in 2013 and 6 in 2014)
- 🌀 Already **9,600** commits from **90** contributors.

Astropy core functionality

Core data structures and transformations

- Constants (**astropy.constants**)
- Units and Quantities (**astropy.units**)
- N-dimensional datasets (**astropy.nddata**)
- Data Tables (**astropy.table**)
- Time and Dates (**astropy.time**)
- Astronomical Coordinate Systems (**astropy.coordinates**)
- World Coordinate System (**astropy.wcs**)
- Models and Fitting (**astropy.modeling**)

Connecting up: Files and I/O

- Unified file read/write interface
- FITS File handling (**astropy.io.fits**)
- ASCII Tables (**astropy.io.ascii**)
- VOTable XML handling (**astropy.io.votable**)
- Miscellaneous Input/Output (**astropy.io.misc**)

Astropy core functionality

Astronomy computations and utilities

- Convolution and filtering ([astropy.convolution](#))
- Cosmological Calculations ([astropy.cosmology](#))
- Astrostatistics Tools ([astropy.stats](#))
- Virtual Observatory Access ([astropy.vo](#))

Nuts and bolts of Astropy

- Configuration system ([astropy.config](#))
- I/O Registry ([astropy.io.registry](#))
- Logging system
- Python warnings system
- Astropy Core Package Utilities ([astropy.utils](#))

Code examples

Taken from the Astropy documentation <http://docs.astropy.org>

astropy.units

The `astropy.units` package provides the `Unit` and `Quantity` class:

```
>>> from astropy.units import Unit, Quantity
>>> Unit('km / h')
Unit("km / h")
>>> Quantity(42, 'km / h')
<Quantity 42.0 km / h>
```

The `astropy.units` namespace also contains a lot of unit objects and you can use arithmetic to create units and quantities:

```
>>> from astropy import units as u
>>> len(dir(u))
2627
>>> u.meter
Unit("m")
>>> 42 * u.meter
<Quantity 42.0 m>
>>> 9.8 * u.m / u.s ** 2
<Quantity 9.8 m / s2>
```

`Quantity` is a `numpy.ndarray` sub-class, so you can make quantities that represent arrays of values with the same unit:

```
>>> [1., 2., 3.] * u.m
<Quantity [ 1., 2., 3.] m>
>>> import numpy as np
>>> np.array([1., 2., 3.]) * u.m
<Quantity [ 1., 2., 3.] m>
```

Quantities have a value and a unit:

```
>>> q = 42.0 * u.meter
>>> q.value
42.0
>>> q.unit
Unit("m")
```

And a few other useful properties:

```
>>> u.eV.__doc__
'Electron Volt'
>>> u.eV.physical_type
'energy'
>>> u.eV.find_equivalent_units()
# didn't fit on the slide ...
# prints a table containing `Joule` and `erg`.
```

astropy.constants

Contains astronomical and physical constants for use in Astropy or other places.

A typical use case might be:

```
>>> from astropy.constants import c, m_e
>>> # ... define the mass of something you want the rest energy of as m ...
>>> m = m_e
>>> E = m * c**2
>>> E.to('MeV')
<Quantity 0.510998927603161 MeV>
```

The following constants are available:

Name	Value	Unit	Description
G	6.67384e-11	m ³ / (kg s ²)	Gravitational constant
L_sun	3.846e+26	W	Solar luminosity
M_earth	5.9742e+24	kg	Earth mass
M_jup	1.8987e+27	kg	Jupiter mass
M_sun	1.9891e+30	kg	Solar mass

And lots more ...

CTA Design study

astropy.coordinates

The **coordinates** package provides classes for representing a variety of celestial/spatial coordinates, as well as tools for converting between common coordinate systems in a uniform way.

```
>>> from astropy import units as u
>>> from astropy.coordinates import SkyCoord

>>> c = SkyCoord(ra=10.5*u.degree, dec=41.2*u.degree, frame='icrs')
>>> c = SkyCoord(10.5, 41.2, 'icrs', unit='deg')
>>> c = SkyCoord('00h42m00s', '+41d12m00s', 'icrs')
>>> c = SkyCoord('00 42 00 +41 12 00', 'icrs', unit=(u.hourangle, u.deg))
>>> c
<SkyCoord (ICRS): ra=10.5 deg, dec=41.2 deg>
```

Get coordinates for astrophysical objects:

```
>>> SkyCoord.from_name("M42")
<SkyCoord (ICRS): ra=83.82208 deg, dec=-5.39111 deg>
```

List of coordinate systems:

Frame class	Frame name
ICRS	icrs
FK5	fk5
FK4	fk4
FK4NoETerms	fk4noeterms
Galactic	galactic

astropy.table

astropy.table provides functionality for storing and manipulating heterogeneous tables of data in a way that is familiar to **numpy** users. A few notable features of this package are:

- Initialize a table from a wide variety of input data structures and types.
- Modify a table by adding or removing columns, changing column names, or adding new rows of data.
- Handle tables containing missing values.
- Include table and column metadata as flexible data structures.
- Specify a description, units and output formatting for columns.
- Interactively scroll through long tables similar to using `more`.
- Create a new table by selecting rows or columns from a table.
- Perform *Table operations* like database joins and concatenation.
- Manipulate multidimensional columns.
- Methods for *Reading and writing Table objects* to files
- Hooks for *Subclassing Table* and its component classes

```
>>> from astropy.table import Table
>>> a = [1, 4, 5]
>>> b = [2.0, 5.0, 8.2]
>>> c = ['x', 'y', 'z']
>>> t = Table([a, b, c], names=('a', 'b', 'c'), meta={'name': 'first table'})
```


astropy.io.fits

The **astropy.io.fits** package provides access to FITS files. FITS (Flexible Image Transport System) is a portable file standard widely used in the astronomy community to store images and tables.

Once the **astropy.io.fits** package is loaded using the standard convention[1], we can open an existing FITS file:

```
>>> from astropy.io import fits
>>> hdulist = fits.open('input.fits')
```

The **HDUList** has a useful method **HDUList.info()**, which summarizes the content of the opened FITS file:

```
>>> hdulist.info()
Filename: test1.fits
No. Name Type Cards Dimensions Format
0 PRIMARY PrimaryHDU 220 () int16
1 SCI ImageHDU 61 (800, 800) float32
2 SCI ImageHDU 61 (800, 800) float32
3 SCI ImageHDU 61 (800, 800) float32
4 SCI ImageHDU 61 (800, 800) float32
```

© ITK Design Studio

astropy.io.ascii

[astropy.io.ascii](#) provides methods for reading and writing a wide range of ASCII data table formats via built-in *Extension Reader classes*. The emphasis is on flexibility and ease of use.

The following shows a few of the ASCII formats that are available, while the section on [Supported formats](#) contains the full list.

- **Basic** : basic table with customizable delimiters and header configurations
- **Cds** : [CDS format table](#) (also VizieR and ApJ machine readable tables)
- **Daophot** : table from the IRAF DAOPHOT package
- **FixedWidth** : table with fixed-width columns (see also [Fixed-width Gallery](#))
- **Ipac** : [IPAC format table](#)
- **HTML** : HTML format table contained in a `<table>` tag
- **Latex** : LaTeX table with data value in the `tabular` environment
- **Rdb** : tab-separated values with an extra line after the column definition line
- **SExtractor** : [SExtractor format table](#)

This table can be read with the following:

```
>>> from astropy.io import ascii
>>> data = ascii.read("sources.dat")
>>> print data
obsid redshift  X    Y    object
-----
 3102    0.32 4167 4085 Q1250+568-A
   877    0.22 4378 3892 Source 82
```

Astropy affiliated packages

Astropy 'affiliated' packages

Can be:

- Functionality under development for core
- More specialized functionality
- Packages with incompatible licenses

Adhere to Astropy coding, testing, and docs guidelines

Use Astropy wherever possible (avoid duplication)

```
$ python setup.py test --help  
$ python setup.py build_sphinx --help
```

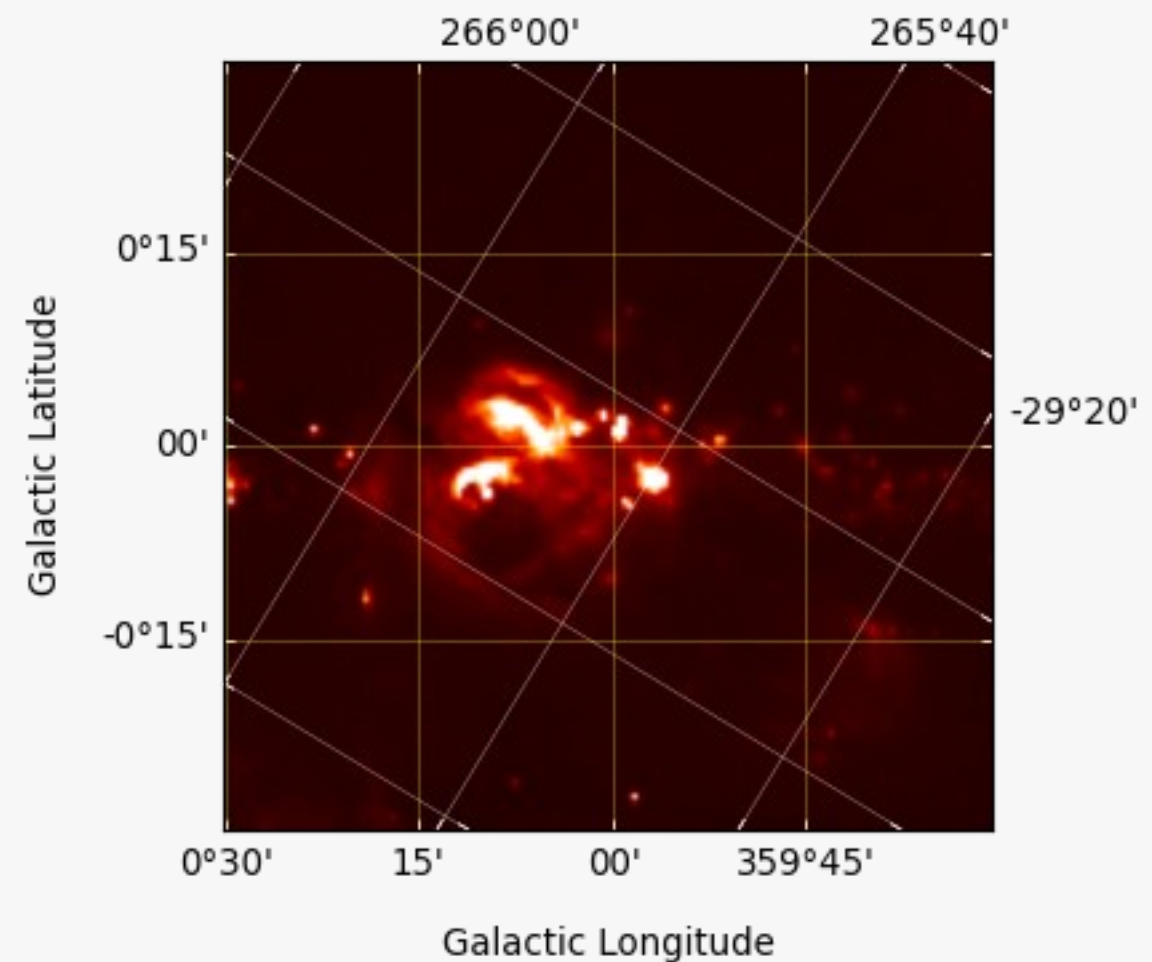
Existing affiliated packages

- 🌀 APLpy – Plotting of Astronomical images
- 🌀 astroML – Machine learning and data mining in Astronomy
- 🌀 astropysics – Utilities for reducing, analyzing, and visualizing data
- 🌀 astroquery – Querying of online databases
- 🌀 ccdproc – Basic reduction of CCD data
- 🌀 **gammapy – Gamma-ray astronomy**
- 🌀 ginga – Interactive FITS file viewer
- 🌀 montage-wrapper – Wrapper for the Montage image mosaicking engine
- 🌀 **photutils – Photometry tools**
- 🌀 pydl – Library of IDL astronomy routines in Python
- 🌀 pyVO – Virtual observatory tools (complements astropy.vo)
- 🌀 sncosmo – Simulating, fitting, and typing of Supernova light curves
- 🌀 specutils – Spectroscopic analysis tools
- 🌀 **wcsaxes – Extensible framework for plotting images with WCS**
- 🌀 ... many more coming ...

WCsAxes

<https://github.com/astrofrog/wcsaxes>

- 🌀 Example of an affiliated package that **will** be included into the Astropy core soon, because it's useful for most astronomers.
- 🌀 WCsAxes is a framework for making plots of Astronomical data in Matplotlib.
- 🌀 WCS = "World coordinate system"
= projections of the sky onto images.
- 🌀 Provides **WCsAxes** class, which subclasses **matplotlib.axes.Axes** and knows about Astropy **coordinates** and **quantities**.

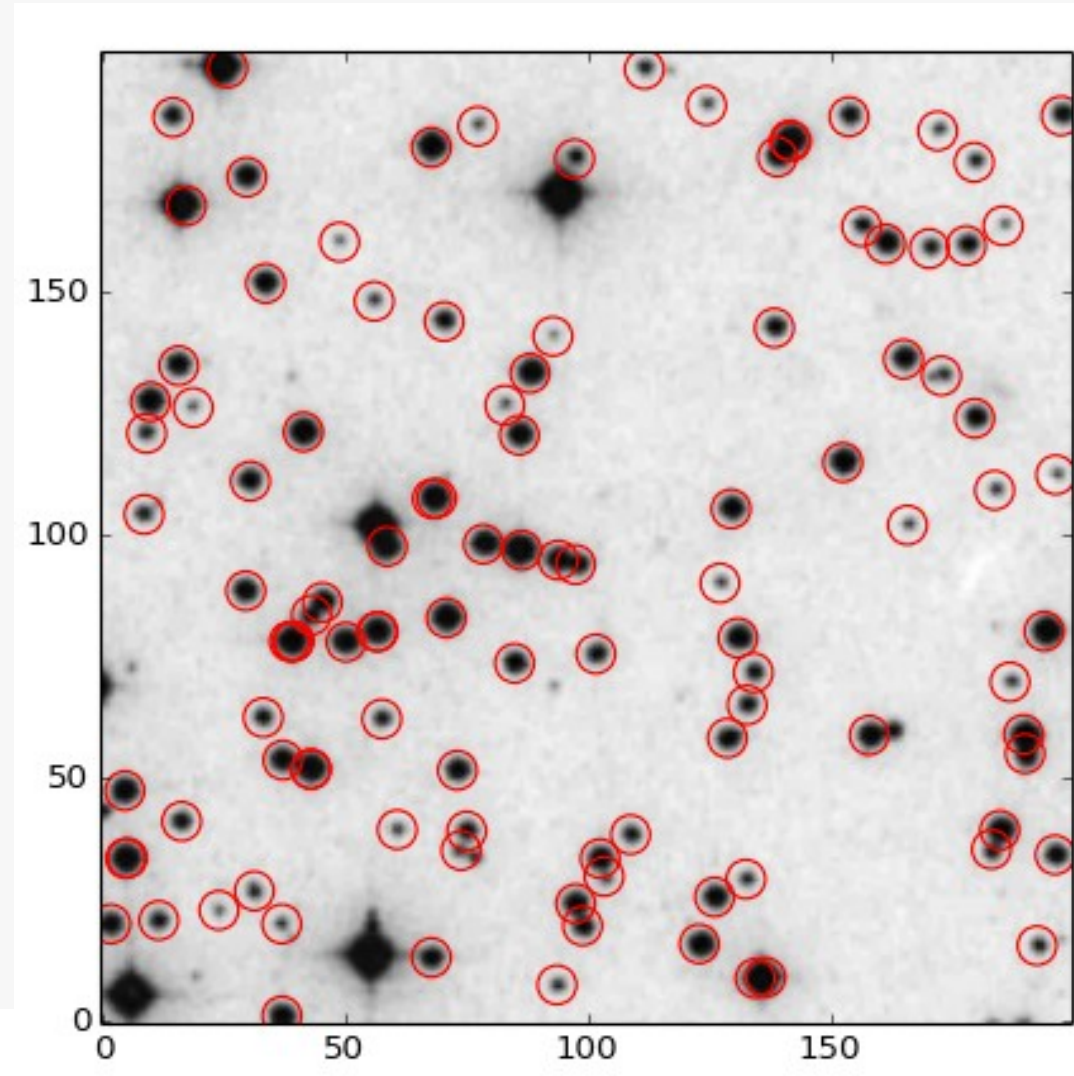


photutils

<https://github.com/astropy/photutils>

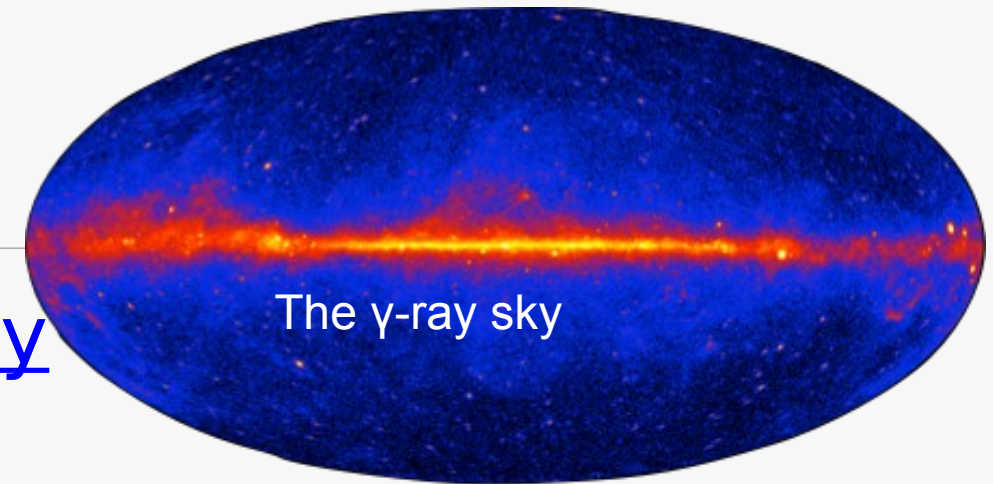
- 🌀 Example of an Astropy affiliated package that **might** be included into the Astropy core, because it's useful for a large fraction of astronomers.
- 🌀 Source detection and characterisation in astronomical images.
- 🌀 Aperture and PSF photometry
- 🌀 Uses `scipy.ndimage` and `scikit-image`

```
sources = daofind(image, fwhm, threshold)
positions = zip(sources['xcen'], sources['ycen'])
apertures = CircularAperture(positions, radius)
fluxes = aperture_photometry(image, apertures)
```

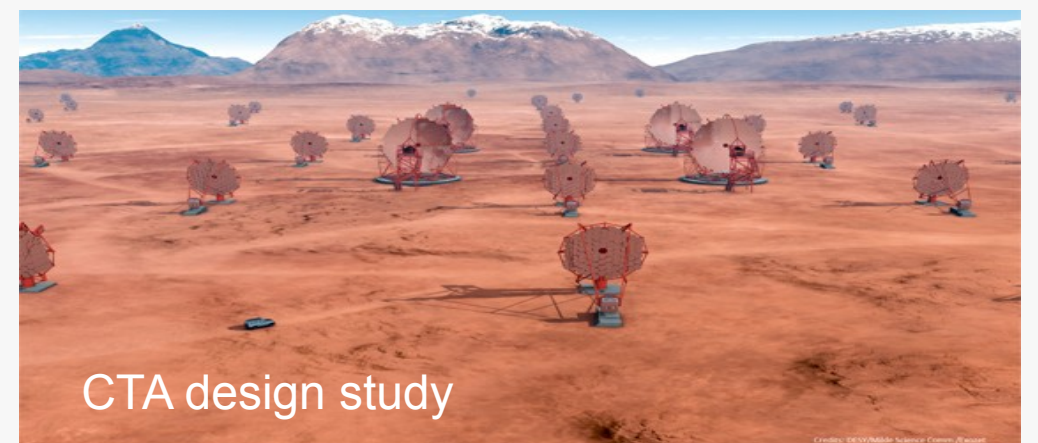
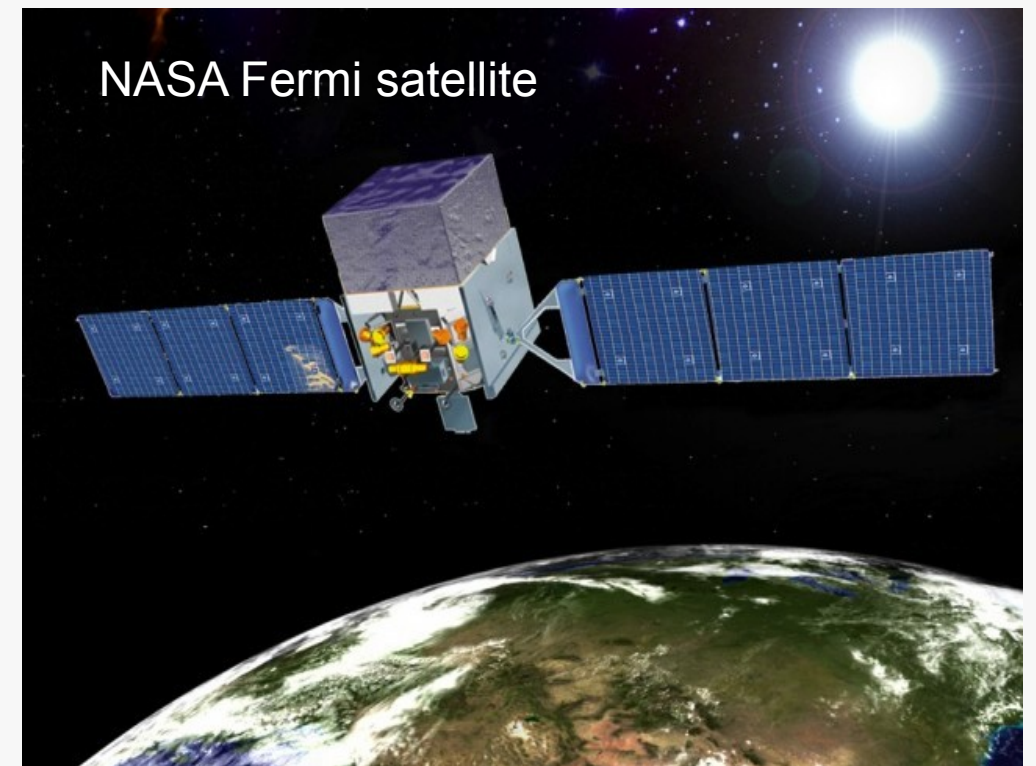


γ π A **Python** package for **gamma-ray** astronomy

<https://github.com/gammapy/gammapy>



- 🌀 Example of an Astropy affiliated package that **will never** be included into the Astropy core, because it's useful only for a sub-community of astronomers.
- 🌀 Tools to simulate and analyse astronomical gamma-ray data from ground- and space-based telescopes
- 🌀 0.1 release a few weeks ago... hopefully useful for the small community of gamma-ray astronomers



Next steps for Astropy

- 🌀 **Improvements and additions to the Astropy core package.**
 - Finish modeling, coordinates, ...
 - Add generalised WCS, imageutils, sphere, ...
 - Use quantities throughout Astropy ...
- 🌀 **Astropy 1.0 release scheduled for December 2014.**
 - Release cycle: 6 month, 2 year LTS
- 🌀 **Extend the ecosystem of Astropy affiliated packages.**
 - A future of open and reproducible science in Astronomy?
- 🌀 **“Python in Astronomy” workshop**
 - 20-24 April 2015, Lorentz Center, Leiden
 - <http://python-in-astronomy.github.io/>

Thank you for your attention!

Web: <http://www.astropy.org>

Docs: <http://docs.astropy.org>

Code: <http://github.com/astropy>

Twitter: @astropy



Astropy

*A community developed core
package for Astronomy in Python*



Axel Donath
MPIK Heidelberg

Herbsttagung der Deutschen Astronomischen
Gesellschaft, Bamberg 2014

Introduction:

Hello, first I'd like to introduce myself shortly:

- + Name Axel Donath
- + PHD Student working @ MPIK Heidelberg
- + Field of research is Gamma-Ray astronomy
- + Involved in the HESS and CTA Galactic Plane Survey
- + Not talk about my research, but talk about Astropy for the next 20 minutes

What is Astropy?

What is Astropy?

+ Before answering question, go back a few years and see what happened on the Python Astronomy mailing list.

How the Astropy project started

June 9th 2011 on the Astropy mailing list ...

- [\[AstroPy\] PyAstronomy](#) Stefan Czesla
 - [\[AstroPy\] Proliferating py-astro-libs](#) Marshall Perrin
 - [\[AstroPy\] Proliferating py-astro-libs](#) Wolfgang Kerzendorf

On Jun 9, 2011, at 12:54 PM, Stefan Czesla wrote:
Dear all,

we would like to let you know about our recent release of a -- hopefully --
useful contribution to Python's astronomy community, namely, our PyAstronomy
package (yes, there have been more inspired names...). It consists of

On 10/06/11 8:25 AM, Marshall Perrin wrote:

*Hopefully without sounding too critical of you in particular, I'm
going to ask: do we as a community really need /yet another/ separate
python library for astronomy and yet another attempt at building a
core set of routines ported from the IDL library?*

Stefan Cezla introduced an new Python package,
But the response was not overwhelming as he
might have expected.

A day later Marshall Perrin responded...

This coment started an extensive discussion on
the need of new packages and proliferating of
Python astronomy libraries.

Quote from the mailing list Illustrates the problem
During that time.

The problem

- astLib
- astrolib
- astropysics
- pyephem
- ephempy
- pyast
- pyastro
- apwlib
- chiantipy
- pymidas
- pandora
- pyspec
- pyraf
- atpy
- aplpy
- pynovas
- cosmopy
- cosmology
- kapteyn
- cosmics.py
- pyguide
- astrogui
- pyastronomy
- pywcs
- pyfits
- sunpy
- ...more?

Overview about all available python öibraries at that time

- + PyWCS for WCS transformation
- + kapteyn for coordinates transformations
- + pyfits for reading/writing FITS files
- + atpy for handling of data tables
- + etc.

Many different Python packages for astronomy, with different purposes and different, often single, authors

On astropy mailing list idea born to start a new package... yet another package?

No but rather start a package, that bundles all the basic functionallity into one single open source python package

The solution

- 🌀 Start the Astropy project ... a community effort to develop a **core package for Astronomy in Python** and **foster interoperability** between Python astronomy packages.
- 🌀 **Astropy core package** ... useful for most astronomers.
- 🌀 **Astropy affiliated packages** ... for more specialised astronomy applications. Leverage Astropy infrastructure (packaging, testing, documentation) by using a package template.
- 🌀 Astropy project coordinators:
 - Perry Greenfield, STScI
 - Thomas Robitaille, MPIA
 - Erik Tollerud, Yale

Idea was to start the astropy python package, which should be....

Should consist of:

+ Astropy core package, which bundles basic functionality, that is useful for most astronomers

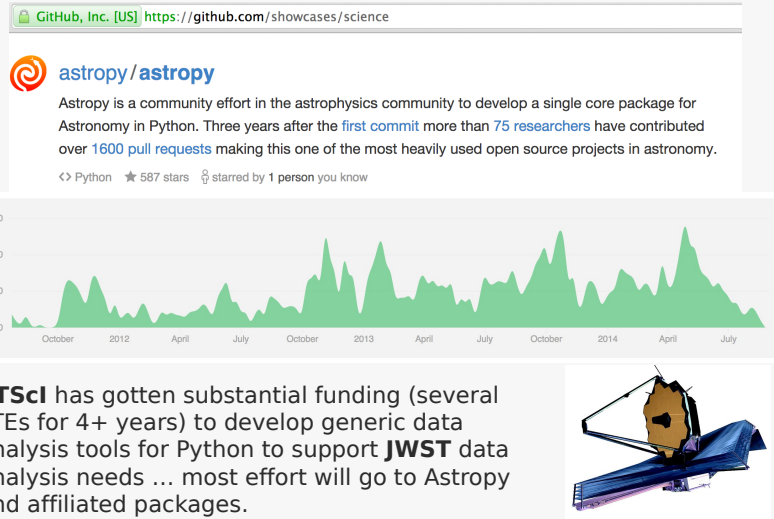
+ So called “affiliated packages”, which contain special functionality for certain fields of astronomy.

Main coordinators...

Going back to presence...

3 years later ...

...it has worked out very well!



Astropy - A community developed core package for Astronomy

6

Astropy has found a home on GitHub and has worked out very well!

3 yrs after the first commit 75 researchers have contributed over 1600 pull requests to the project..

Timeline: Show 100 commits...

Recently the Sctci....

To conclude introduction, I'd like to explain what my connection to Astropy is...

- + Use it for my daily work, I'm a user
- + Also developer, GSoC Project for Astropy
- + Co-Developer for an astropy affiliated package for Gamma-Ray astronomy

Astropy core package

Let's have a closer look at the package...

Astropy core status

- 🌀 3 major public releases (first release February 2013)
- 🌀 Latest stable version: 0.4.1 (released August 2014)
- 🌀 **Astropy 0.2 paper** in Astronomy & Astrophysics journal
(a nice sign that an astronomy science journal accepts a paper only presenting a software, not a science application)
- 🌀 GSoC students (2 students in 2013 and 6 in 2014)
- 🌀 Already **9,600** commits from **90** contributors.

Status overview:

- + 3 major public releases
- + Latest stable version 0.4
- + Astropy 0.2 paper published in A&A, Note: Kind of quality proof, that a software paper is published...
- + In 2013 and 2014 Astropy was part of the GSoC project, where students work on open source software, as a “summer job”.
- + Most recent numbers 9600, commits from over 90 contributors

Astropy core functionality

Core data structures and transformations

- Constants (`astropy.constants`)
- Units and Quantities (`astropy.units`)
- N-dimensional datasets (`astropy.nddata`)
- Data Tables (`astropy.table`)
- Time and Dates (`astropy.time`)
- Astronomical Coordinate Systems (`astropy.coordinates`)
- World Coordinate System (`astropy.wcs`)
- Models and Fitting (`astropy.modeling`)

Connecting up: Files and I/O

- Unified file read/write interface
- FITS File handling (`astropy.io.fits`)
- ASCII Tables (`astropy.io.ascii`)
- VOTable XML handling (`astropy.io.votable`)
- Miscellaneous Input/Output (`astropy.io.misc`)

Overview:

+Core functionality is divided into several submodules

constants: collection of physical and astronomical constants with, value unit and references...

Units: Classes for defining variables and arrays with units, unit conversions etc.

Nddata: handling of multi-dimensional datasets, with masks

Table: Provides data structures for table data and Functions to access and manipulate them.

Coordinates: Contains classes for representing, celestial and spatial coordinates, and function to format and convert them.

WCS: Former PyWCS package, handles projections of world coordinates onto images etc.

Modeling: Framework for modeling and fitting data.

Several modules for handling astronomical file formats.
e.g. FITS, ascii, VOTables, etc..

+ Later go into more detail and show some code examples

Astropy core functionality

Astronomy computations and utilities

- Convolution and filtering ([astropy.convolution](#))
- Cosmological Calculations ([astropy.cosmology](#))
- Astrostatistics Tools ([astropy.stats](#))
- Virtual Observatory Access ([astropy.vo](#))

Nuts and bolts of Astropy

- Configuration system ([astropy.config](#))
- I/O Registry ([astropy.io.registry](#))
- Logging system
- Python warnings system
- Astropy Core Package Utilities ([astropy.utils](#))

Here is the second part of the list of core functionality:

Furthermore Convolution and filtering: with proper NaN handling and a different kernel types

Cosmology: Cosmological calculations, redshifts, comoving distances, etc.

Astrostats: Statistical function useful for astronomers, kappa-sigma clipping for background estimation, or outlier robust mean and variance estimates

Nuts and Bolts:

Logging and warning system

And utility functions

Code examples

Taken from the Astropy documentation <http://docs.astropy.org>

To illustrate what the different modules can be used for, I'd like to take a closer look at a few of them and show some code examples...

astropy.units

The `astropy.units` package provides the `Unit` and `Quantity` class:

```
>>> from astropy.units import Unit, Quantity
>>> Unit('km / h')
Unit("km / h")
>>> Quantity(42, 'km / h')
<Quantity 42.0 km / h>
```

The `astropy.units` namespace also contains a lot of unit objects and you can use arithmetic to create units and quantities:

```
>>> from astropy import units as u
>>> len(dir(u))
2627
>>> u.meter
Unit("m")
>>> 42 * u.meter
<Quantity 42.0 m>
>>> 9.8 * u.m / u.s ** 2
<Quantity 9.8 m / s2>
```

`Quantity` is a `numpy.ndarray` sub-class, so you can make quantities that represent arrays of values with the same unit:

```
>>> [1., 2., 3.] * u.m
<Quantity [ 1., 2., 3.] m>
>>> import numpy as np
>>> np.array([1., 2., 3.]) * u.m
<Quantity [ 1., 2., 3.] m>
```

Quantities have a value and a unit:

```
>>> q = 42.0 * u.meter
>>> q.value
42.0
>>> q.unit
Unit("m")
```

And a few other useful properties:

```
>>> u.eV.__doc__
'Electron Volt'
>>> u.eV.physical_type
'energy'
>>> u.eV.find_equivalent_units()
# didn't fit on the slide ...
# prints a table containing `Joule` and `erg`.
```

astropy.constants

Contains astronomical and physical constants for use in Astropy or other places.

A typical use case might be:

```
>>> from astropy.constants import c, m_e
>>> # ... define the mass of something you want the rest energy of as m ...
>>> m = m_e
>>> E = m * c**2
>>> E.to('MeV')
<Quantity 0.510998927603161 MeV>
```

The following constants are available:

Name	Value	Unit	Description
G	6.67384e-11	m ³ / (kg s ²)	Gravitational constant
L_sun	3.846e+26	W	Solar luminosity
M_earth	5.9742e+24	kg	Earth mass
M_jup	1.8987e+27	kg	Jupiter mass
M_sun	1.9891e+30	kg	Solar mass

And lots more ...

CTA Design study

astropy.coordinates

The **coordinates** package provides classes for representing a variety of celestial/spatial coordinates, as well as tools for converting between common coordinate systems in a uniform way.

```
>>> from astropy import units as u
>>> from astropy.coordinates import SkyCoord

>>> c = SkyCoord(ra=10.5*u.degree, dec=41.2*u.degree, frame='icrs')
>>> c = SkyCoord(10.5, 41.2, 'icrs', unit='deg')
>>> c = SkyCoord('00h42m00s', '+41d12m00s', 'icrs')
>>> c = SkyCoord('00 42 00 +41 12 00', 'icrs', unit=(u.hourangle, u.deg))
>>> c
<SkyCoord (ICRS): ra=10.5 deg, dec=41.2 deg>
```

Get coordinates for astrophysical objects:

```
>>> SkyCoord.from_name("M42")
<SkyCoord (ICRS): ra=83.82208 deg, dec=-5.39111 deg>
```

List of coordinate systems:

Frame class	Frame name
ICRS	icrs
FK5	fk5
FK4	fk4
FK4NoETerms	fk4noeterms
Galactic	galactic

astropy.table

astropy.table provides functionality for storing and manipulating heterogeneous tables of data in a way that is familiar to **numpy** users. A few notable features of this package are:

- Initialize a table from a wide variety of input data structures and types.
- Modify a table by adding or removing columns, changing column names, or adding new rows of data.
- Handle tables containing missing values.
- Include table and column metadata as flexible data structures.
- Specify a description, units and output formatting for columns.
- Interactively scroll through long tables similar to using `more`.
- Create a new table by selecting rows or columns from a table.
- Perform *Table operations* like database joins and concatenation.
- Manipulate multidimensional columns.
- Methods for *Reading and writing Table objects* to files
- Hooks for *Subclassing Table* and its component classes

```
>>> from astropy.table import Table
>>> a = [1, 4, 5]
>>> b = [2.0, 5.0, 8.2]
>>> c = ['x', 'y', 'z']
>>> t = Table([a, b, c], names=('a', 'b', 'c'), meta={'name': 'first table'})
```


astropy.io.fits

The [astropy.io.fits](#) package provides access to FITS files. FITS (Flexible Image Transport System) is a portable file standard widely used in the astronomy community to store images and tables.

Once the [astropy.io.fits](#) package is loaded using the standard convention^[1], we can open an existing FITS file:

```
>>> from astropy.io import fits
>>> hdulist = fits.open('input.fits')
```

The [HDUList](#) has a useful method [HDUList.info\(\)](#), which summarizes the content of the opened FITS file:

```
>>> hdulist.info()
Filename: test1.fits
No. Name Type Cards Dimensions Format
0 PRIMARY PrimaryHDU 220 () int16
1 SCI ImageHDU 61 (800, 800) float32
2 SCI ImageHDU 61 (800, 800) float32
3 SCI ImageHDU 61 (800, 800) float32
4 SCI ImageHDU 61 (800, 800) float32
```

astropy.io.ascii

astropy.io.ascii provides methods for reading and writing a wide range of ASCII data table formats via built-in *Extension Reader classes*. The emphasis is on flexibility and ease of use.

The following shows a few of the ASCII formats that are available, while the section on [Supported formats](#) contains the full list.

- **Basic** : basic table with customizable delimiters and header configurations
- **Cds** : [CDS format table](#) (also Vizier and ApJ machine readable tables)
- **Daophot** : table from the IRAF DAOPHOT package
- **FixedWidth** : table with fixed-width columns (see also [Fixed-width Gallery](#))
- **Ipac** : [IPAC format table](#)
- **HTML** : HTML format table contained in a `<table>` tag
- **Latex** : LaTeX table with data value in the `tabular` environment
- **Rdb** : tab-separated values with an extra line after the column definition line
- **SExtractor** : [SExtractor format table](#)

This table can be read with the following:

```
>>> from astropy.io import ascii
>>> data = ascii.read("sources.dat")
>>> print data
obsid redshift X Y object
-----
3102 0.32 4167 4085 Q1250+568-A
877 0.22 4378 3892 Source 82
```

Astropy affiliated packages

Already mentioned: Not only core functionality, but so called affiliated packages, what's the purpose of these packages?

Astropy 'affiliated' packages

Can be:

- Functionality under development for core
- More specialized functionality
- Packages with incompatible licenses

Adhere to Astropy coding, testing, and docs guidelines

Use Astropy wherever possible (avoid duplication)

```
$ python setup.py test --help  
$ python setup.py build_sphinx --help
```

Existing affiliated packages

- 🌀 APLpy - Plotting of Astronomical images
- 🌀 astroML - Machine learning and data mining in Astronomy
- 🌀 astropyphysics - Utilities for reducing, analyzing, and visualizing data
- 🌀 astroquery - Querying of online databases
- 🌀 ccdproc - Basic reduction of CCD data
- 🌀 **gammapy - Gamma-ray astronomy**
- 🌀 ginga - Interactive FITS file viewer
- 🌀 montage-wrapper - Wrapper for the Montage image mosaicking engine
- 🌀 **photutils - Photometry tools**
- 🌀 pydl - Library of IDL astronomy routines in Python
- 🌀 pyVO - Virtual observatory tools (complements astropy.vo)
- 🌀 snocosmo - Simulating, fitting, and typing of Supernova light curves
- 🌀 specutils - Spectroscopic analysis tools
- 🌀 **wcsaxes - Extensible framework for plotting images with WCS**
- 🌀 ... many more coming ...

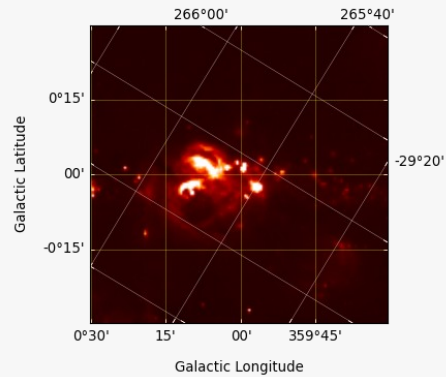
Here are a few examples:

I've chosen three example for affiliated packages

WCsAxes

<https://github.com/astrofrog/wcsaxes>

- Example of an affiliated package that **will** be included into the Astropy core soon, because it's useful for most astronomers.
- WCsAxes is a framework for making plots of Astronomical data in Matplotlib.
- WCS = "World coordinate system" = projections of the sky onto images.
- Provides **WCsAxes** class, which subclasses **matplotlib.axes.Axes** and knows about Astropy **coordinates** and **quantities**.



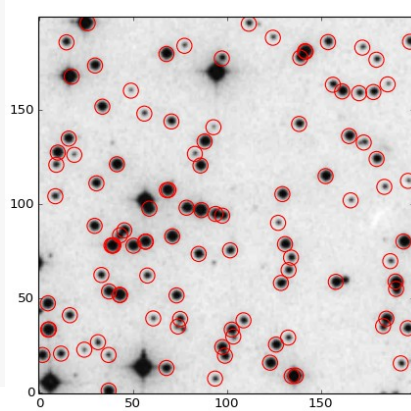
Overlay coordinate grids in different coordinate systems, add contours,

photutils

<https://github.com/astropy/photutils>

- 🌀 Example of an Astropy affiliated package that **might** be included into the Astropy core, because it's useful for a large fraction of astronomers.
- 🌀 Source detection and characterisation in astronomical images.
- 🌀 Aperture and PSF photometry
- 🌀 Uses `scipy.ndimage` and `scikit-image`

```
sources = daofind(image, fwhm, threshold)
positions = zip(sources['xcen'], sources['ycen'])
apertures = CircularAperture(positions, radius)
fluxes = aperture_photometry(image, apertures)
```



Worked on during the GsoC project

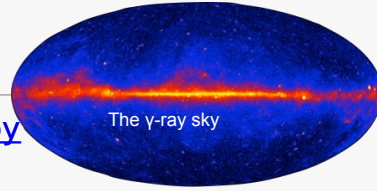
Catalog tools for detection and characterisation of sources in astronomical images, to perform photometry on images

Implementation of standard routines like DaoPhot,

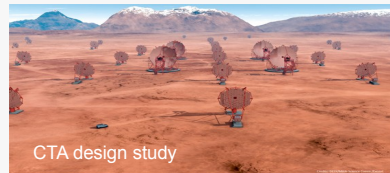
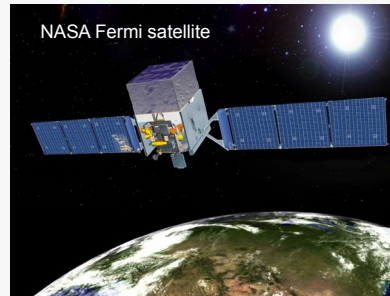
To avoid duplication of code there are additional dependencies, to `scikit-image` e.g.

Explain example!

Photometry with only a few lines of code



- Example of an Astropy affiliated package that **will never** be included into the Astropy core, because it's useful only for a sub-community of astronomers.
- Tools to simulate and analyse astronomical gamma-ray data from ground- and space-based telescopes
- 0.1 release a few weeks ago... hopefully useful for the small community of gamma-ray astronomers



Affiliated package I'm working on together with Christoph Deil.

Provides Tools, to simulate ...

Next steps for Astropy

- 🌀 **Improvements and additions to the Astropy core package.**
 - Finish modeling, coordinates, ...
 - Add generalised WCS, imageutils, sphere, ...
 - Use quantities throughout Astropy ...
- 🌀 **Astropy 1.0 release scheduled for December 2014.**
 - Release cycle: 6 month, 2 year LTS
- 🌀 **Extend the ecosystem of Astropy affiliated packages.**
 - A future of open and reproducible science in Astronomy?
- 🌀 **“Python in Astronomy” workshop**
 - 20-24 April 2015, Lorentz Center, Leiden
 - <http://python-in-astronomy.github.io/>

Conclusion:

Astropy already provides a lot of useful functionality, but there is still a lot of improvements to be done.

E.g.

Thank you for your attention!

Web: <http://www.astropy.org>

Docs: <http://docs.astropy.org>

Code: <http://github.com/astropy>

Twitter: @astropy

