Astropy: Building Blocks for Astronomy Software

Erik M. Bray

Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218, USA

Introduction

The Astropy Project is a community effort to develop an open source Python package of common data structures and routines for use by other, more specialized astronomy software in order to foster interoperability.

The project encompasses the "core" **`astropy`** Python package, "affiliated packages" that strive to implement Astropy's coding standards and interoperability with other affiliated packages, and a broader community aimed at implementing Pythonic solutions to astronomy computing problems while minimizing duplication of effort.

Here we present an overview of the key features of the core package and some of the free and open source technologies and services we use to manage a large project with a diverse community of contributors.

Astropy Affiliated Packages

An affiliated package is a Python package that is not part of the core **`astropy`** package, but is under the umbrella of the Astropy Project. Some affiliated packages may be moved into the core package once they reach a reasonable level of maturity. Others may be more specialized or have enough external dependencies (GUI libraries, for example) that they make more sense as separate packages. In general we hope that becoming an affiliated package will be seen as a good way for new and existing packages to gain exposure.

Some currently featured affiliated packages include Montage-wrapper, Ginga, APLpy, astroML, and Astropysics. A few additional affiliated packages that are currently under development are photutils, astroquery, specutils, and kcorrect.

Citing Astropy

If you use Astropy for work/research presented in a publication, we would be grateful if you could include the following citation:

The Astropy Collaboration, Robitaille T. P., Tollerud E. J., Greenfield P., et al. 2013, ArXiv e-prints. 1307.6212

which is a citation to this paper, on which this poster is based: http://arxiv.org/abs/1307.6212

Core `astropy` capabilities

Development status: Active; less stable API · Reasonably stable · Mature

Core data structures and transformations		
`astropy.units`	Reasonably stable	Handles defining and converting between physical units, and performing arithmetic with physical quantities (numerical scalars and arrays with associated units) using a powerful `Quantity` class.
`astropy.constants`	Active; less stable API	Contains a number of physical constants useful in astronomy and physics. Constants are representing as `Quantity` objects with additional metadata describing their provenance and uncertainties.
`astropy.nddata`	Active; less stable API	Provdes an `NDData` class for managing n-dimensional array-based data with attached metadata, errors, and masking. Also provides new convolution routines that treat NaN values properly.
`astropy.table`	Reasonably stable	Provides a powerful `Table` class for storing and manipulating heterogeneous tables of data in a way that is familiar to Numpy users, though with more flexibility than the data structures built into Numpy including support for metadata and masking.
`astropy.time`	Reasonably stable	Provides functionality for manipulating times and dates. Specific emphasis is placed on supporting time scales (e.g. UTC, TAI, UT1) and time representations (e.g. JD, MJD, ISO 8601) that are used in astronomy. Wraps the ERFA library of time and calendar routines, which is a fork of SOFA released under a less restrictive license.
`astropy.coordinates`	Active; less stable API	Provides classes and transformation functions for representing celestial coordinates and for converting between standard systems in a uniform way.
`astropy.wcs`	Reasonably stable	Contains utilities for managing WCS transformations in FITS files. These transformations map the pixel locations in an image to their real-world units. This sub-package is an import of the already widely used PyWCS package.
`astropy.modeling`	Active; less stable API	Provides a framework for representing models performing model evaluation and fitting. It supports 1D and 2D models and fitting with parameter constraints, as well as an easy to use API for defining new models and fitting algorithms.
File I/O		
`astropy.io.fits`	Mature	A port of the already widely used PyFITS package, supporting reading and writing FITS files including support for many non-standard FITS conventions.
`astropy.io.ascii`	Mature	A port of the popular asciitables package with support from reading from and writing a wide range of plain text formats to and from Astropy `Table` objects.
`astropy.io.votable`	Mature	A port of the VOTable package for reading and writing the IVOA VOTable format to and from Astropy `Table` objects, as well as validating them.
`astropy.io.misc`	Reasonably stable	Includes optional support for tables stored in HD5 where `h5py` is available.
Astronomy computations and utilities		
`astropy.cosmology`	Reasonably stable	Contains classes for representing cosmologies, and utility functions for calculating commonly used quantities that depend on a cosmological model. This includes distances, ages and lookback times corresponding to a measured redshift or the transverse separation corresponding to a measured angular separation.
`astropy.stats`	Active; less stable API	This sub-package holds statistical functions and algorithms used in astronomy, some of which are more specialized than one would find in `scipy.stats`.
`astropy.vo`	Active; less stable API	Handles simple access for Virtual Observatory (VO) services. Currently, only Simple Cone Search Version 1.03 as defined in IVOA Recommendation (February 22, 2008) is supported.

Community and Collaboration

A primary guiding principle behind Astropy has been that it is developed by and for the astronomy community, and that it strives to include contributions from users and developers from around the world and from all sub-disciplines. It also aims to encourage best practices from software engineering in the hope of
creating a clean, consistent, and stable API; thorough and well-maintained documentation; and constant feedback through careful issue tracking and continuous integration testing. All of our development practices and guidelines are documented so that other projects—whether affiliated with Astropy or not—may adopt them without having to develop their own standards from scratch.

In order to better enable collaboration we have adopted the GitHub code hosting service which provides an issue tracker and a git source code repository. GitHub is designed around collaboration and enables a low barrier of entry to submitting new code and bug fixes to open source projects via "pull requests".

Astropy's development process ensures that any bug fixes or new features that are implemented are accompanied by tests, and any relevant updates to the documentation. Free documentation building and hosting by Read the Docs ensures that the latest documentation builds and is accessible online. All new pull requests are tested against multiple Python and NumPy versions using the continuous integration system Travis. Comprehensive testing against multiple OS platforms (Linux, MacOS X, and Windows) is made possible by Jenkins with hosting from Shining Panda.

Having an accessible development workflow has attracted source code contributions from over 50 developers, and many more have contributed design, feedback, and testing.

Technologies and Services We Use

Python, NumPy, SciPy, Matplotlib, Sphinx,
Git, GitHub, Read the Docs, Travis, Jenkins, Shining Panda,
py.test (not pictured)