Astronomical Data Analysis Software and Systems IX
ASP Conference Series, Vol. 216, 2000
N. Manset, C. Veillet, and D. Crabtree, eds.

PyFITS, a Python FITS Module

P. E. Barrett

Space Telescope Science Institute, SESD/DPT, Baltimore, MD 21218

W. T. Bridgman

NASA/Goddard Space Flight Center, RITSS, Greenbelt, MD 20771

Abstract. PyFITS is a Python module for reading, writing, and manipulating FITS files. The module uses Python's object-oriented features to provide quick, easy, and efficient access to FITS files. The use of Python's array syntax enables immediate access to any FITS extension, header cards, or data items. The FITS module is written in Python for maintainability and portability and uses C-extension modules, Numeric and Record, for efficient access to the data. These and other features, and future developments are discussed in this paper.

1. Python Programming Language

Python is an interpreted, object-oriented programming language and is often compared to Tcl, Perl, Scheme, or Java. It can be used interactively by entering `python` at the shell prompt or as a script by executing `python script.py`. Python combines remarkable power with very clear syntax and has modules, classes, exceptions, very high-level dynamic data types, and dynamic typing. There are interfaces to many system calls and libraries, as well as to various windowing systems (X11, Motif, Tk, Mac, MFC). New built-in modules are easily written in C and C++. Python is also usable as an extension language for applications that need a programmable interface.

The Python implementation is portable: it runs on various flavors of UNIX, on Windows, DOS, OS/2, Amiga, etc. Python is copyrighted but freely usable and distributable, even for commercial use. Access to the source code can be found through the Python Home Page.

2. PyFITS Module

PyFITS is a Python module for reading, writing, and manipulating FITS files. At the highest level, a FITS file is treated as a list of *header-data units* or HDUs, so HDU access is by array index or, optionally, extension name (e.g. `hdu[0]` and `hdu['PRIMARY']` access the primary HDU). This latter feature of PyFITS enables dictionary or associative-array type access to HDUs.

Each HDU contains two parts, a `.header` and `.data` attribute, though the data attribute may contain no data. The header part of the HDU is a list of cards containing a keyword and optionally a value and comment. Like HDUs, cards are accessed by array index or keyword name (e.g. `hdu.header[0]` and `hdu.header['EXTENSION']` access the first keyword's value).

The data part of the HDU is an array or a list of records depending on the extension type. For example, the data in the primary HDU has array behavior, while data in a binary table is a one-dimensional array of C-like structures or Records. It is therefore possible to access the entire data array or table, or just a multi-dimensional slice, by using array syntax. For example, `hdu.data[10:-10, 10:-10]` gets and trims a 10 pixel border from an image array, or `hdu.data[:,'X']` gets column 'X' from a binary table.

3. Sample Python Session

```
# python
Python 1.5.1 (#1, Mar 21 1999, 22:49:36) [GCC egcs on linux-i386]
Copyright 1991-1995 Stichting Mathematisch Centrum, Amsterdam
>>>
>>> #  Load FITS and Numeric module
>>> from FITS import *
>>> from Numeric import *
>>>
>>> #  Open data file and show number of extensions
>>> fits = FITS('mydata.fits')
>>> len(fits)
4
>>> print fits[0].header
SIMPLE  =                    T / primary header
BITPIX  =                    8 / array data type
NAXIS   =                    0 / number of array dimensions
EXTEND  =                    T / There may be standard extensions
DATE    = '10/05/93'           / FITS creation date
EVENT   = 'EVENTS  '           / A3D table containing event record
HISTORY = 'rp30019301.toe (hdr=pspc.hdr, gti=rp30019301.gti' /
TITLE   = 'TV COLUMBAE'        /
OBJECT  = 'TV COLUMBAE'        /
QPOENAME= './rp30019301.qp'    / IRAF QPOE file name
>>>
>>> print fits[0].header['OBJECT']
TV COLUMBAE
>>> print fits[0].data
None
>>> #  Read, trim (10 pixels), and filter (between 0 - 1000) image
>>> trim = clip(fits['SCI,1'].data[10:-10,10:-10], 0 ,1000)
>>>
>>> #  Create cumulative distribution of image histogram
>>> accum = add.accumulate(histogram(trim))
>>>
>>> #  Get value at 90% level (ie. last nonzero value)
>>> val90 = nonzero(less(accum, 0.9*len(accum)))[-1]
>>>
>>> #  Compute background rate from exposure time
>>> rate = (accum[-1] - val90)/fits['PRIMARY'].header['EXPTIME']
>>>
```

```
>>> #  Open new FITS file
>>> outfits = FITS('new.fits', 'w')
>>>
>>> #  Save background subtract image to 1st extension
>>> ext = ImageHDU(data=trim, name='SCI,1')
>>>
>>> #  Set RATE keyword in header
>>> ext.header['RATE'] = rate
>>>
>>> #  Append extension and close files
>>> outfits.append(ext)
>>> fits.close()
>>> outfits.close()
```

4. Implementing PyFITS

PyFITS is mostly coded in Python using Python classes, since this allows for the fastest development and most portability. PyFITS contains 13 classes: Boolean, Card, Header, Array, PrimaryHDU, ImageHDU, GroupsHDU, Field, Table, TableHDU, BinTableHDU, File, and FITS. Boolean, Card, Header, and Field are low level classes and provide access to header card records and data arrays. The Array and Table are base classes for the header-data units. The other HDU classes inherit from one of these two base classes. The highest level class is the FITS class. An associated class that allows file access is the File class. Note that a FITS object does not necessarily have to be associated with a FITS file. Use of a filename is only necessary when permanent storage is required.

The two parts of PyFITS that are not coded in Python are the Numeric and Record types. These modules are imported by PyFITS and used to access image and table HDU data. During the development of PyFITS, it was found that Python did not have an efficient mechanism to manipulate binary tables, so a C extension module was written to read and write binary tables as an array of records from native machine format to big- or little-endian byte formats.

Python was designed from the start to be extensible and to provide operator overloading, so programming hooks are available for the addition of new extension types. The complex data type was added to Python by using these facilities and without any changes to the Python syntax. The Record module uses the same hooks and language syntax to create a multi-dimensional array of C-like structures. The first index refers to the record (e.g. `hdu.data[0,:]` or `hdu.data[0]` gets the first record of a binary table), while the second index refers to the items in a record (e.g. `hdu.data[:,0]` gets the first column of a binary table). The number of dimensions of a Record is one less than the number of indices, so a FITS binary table is one-dimensional.

One nice implementation feature of PyFITS is the lazy instantiation of the `.data` attribute, which means that PyFITS delays the reading of the data part of the HDU, until it is accessed. This provides two benefits when reading large files: (1) files are read and scanned quickly allowing quick access to header information; and (2) computer memory is used more efficiently.

5. Future Developments

Memory-mapping
As the number of pixels in imaging detectors become ever larger and the efficiency of photon counting detectors become ever greater, the size of FITS files will continue to grow proportionally. Such large (64 MB and larger) image arrays and binary tables will be difficult to analyze on personal computers and workstations with limited memory. One option is to use memory-mapping which maps regions of the data on disk into memory. The data mainly resides on disk, freeing up precious memory and is read from or written to disk, only when it is being accessed.
Enhanced Object Support for Numeric
Another potential area of development is enhanced object support for the Numeric module which is currently used to access image array data. The goal here is to enable data access via one extension type, namely the Numeric module, instead of two as PyFITS is currently implemented. The benefit to the user is simplicity. For example when accessing the column of a binary table the resulting object will have numeric array behavior as the user would expect.

6. Conclusions

PyFITS is a Python extension module which enables astronomers to easily and efficiently manipulate FITS files either interactively using the Python command-line prompt or as part of a large executable program. Most of the program is coded in Python, making it easy to enhance and maintain, while the data access layer has been coded in C, making it fast and efficient. A possible next step in the development of this project would be to add a GUI for file browsing.

The development of PyFITS provides a good example of modern programming principles and design. The implementation uses (1) an object-oriented programming language which enables a modular design, reuse of code via inheritance, and operator overloading; (2) a very-high-level language (VHLL), like Python for rapid program development (development times are typically a fact of ten faster than using compiled languages), while using a low-level language, like C, for speed and efficiency (only for those parts of the code that really need it); and (3) an interpreted (scripting) language for fast development and ease of use.

For information about using Python in scientific research, see:

Computing in Science and Engineering
`http://www.aip.org/cip`

Python Matrix SIG mailing list
`http://www.python.org/sigs/matrix-sig`

Astronomical Python mailing list
`Majordomo@STScI.Edu`
`subscribe astropy`