Astronomical Data Analysis Software and Systems VIII
ASP Conference Series, Vol. 172, 1999
D. M. Mehringer, R. L. Plante, and D. A. Roberts, eds.

PyFITS, a FITS Module for Python

P. E. Barrett[1,2], W. T. Bridgman[1]

NASA/Goddard Space Flight Center, Greenbelt, MD 20771

Abstract. PyFITS is a module for reading, writing, and manipulating FITS files using the interactive, object-oriented language, Python. The module is composed of two files: a generic low-level C library for manipulating multidimensional arrays of C-type structures and a high-level Python module. FITS files can be manipulated at several different levels, beginning with the header-data unit at the highest level to rows and columns of binary tables at the lowest level. In addition, header-data units and columns of binary tables are accessible by index or name. PyFITS also interfaces to NumPy, the Python numerical array module.

1. What is Python?

python, (*Gr. Myth.* An enormous serpent that lurked in the cave of Mount Parnassus and was slain by Apollo). 1. any of a genus of large, non-poisonous snakes of Asia, Africa and Australia that crush their prey to death. 2. popularly, any large snake that crushes its prey. 3. totally awesome, bitchin' language that will someday crush the $s out of certain other so-called VHLL's ;-)

Python is an interpreted, interactive, object-oriented programming language and is often compared to Tcl, Perl, Scheme, or Java. It can be used either interactively, by typing `python` at the shell prompt, or as a script, by typing `python script.py`. It 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 or C++. Python is also usable as an extension language for applications that need a programmable interface.

The Python implementation is portable: it runs on many brands of UNIX, on Windows, DOS, OS/2, Mac, Amiga, etc. If your favorite system isn't listed here, it may still be supported, if there's a C compiler for it. Ask around on the `comp.lang.python` newsgroup – or just try compiling Python yourself. 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[3].

---

[1]Universities Space Research Association, Seabrook, MD

[2]Current address: Space Telescope Science Institute, Baltimore, MD

[3]`http://www.python.org/`

For more information on extending Python, see Pirzkal & Hook (1999) in these proceedings.

2. What is PyFITS?

PyFITS is python module for reading, writing, and manipulating FITS files. A FITS file is treated as a 'list' of *header-data units* or HDUs. Access to the HDU is by an integer index or using the extension name as a dictionary key. The primary HDU can be accessed using the keyword 'PRIMARY'. This keyword or dictionary feature of PyFITS enables access to a HDU without having to know the absolute index of the extension in the FITS file. Hence, software which uses this feature of PyFITS is likely to be more robust and portable than software that does not.

Each HDU contains two parts, a *header* and *data* unit, though the data unit may contain no data. The header part of the HDU is a list of records containing the keyword, the value, and a comment. Access to the header unit is via list behavior, so records can be accessed, assigned, appended, deleted, inserted, etc. by using either its integer index or its keyword. The benefits of keyword access to the header are as noted above.

The data part of the HDU is treated as an array or a list of records depending on the extension type. For example, the primary array obviously has array behavior, while the binary table extension is treated as a 1-dimensional array of C-structures or records. It is therefore possible to access an entire array or extension or just a small slice. For example, when accessing a binary table extension, a single record or list of records can be accessed using array notation. It is also possible to access just a single column of a binary table by using an integer index or the column name. All of these features allow quick and efficient access to all aspects of the data in a FITS file.

The listing below is a sample session demonstrating the use of Python with the PyFITS module.

```
% python
>>> from fitsio import *
>>> fits = Ffile('a_fits_file.fits', 'r')
>>> len(fits)                    # number of HDUs in file
4
>>> fits[0].hdr                  # print primary header
SIMPLE  =                    T  /  FITS STANDARD
BITPIX  =                   16  /  Binary Data
NAXIS   =                    2  /  Number of image axes
NAXIS1  =                  512  /  Dimension of axis 1
NAXIS2  =                  512  /  Dimension of axis 2
BSCALE  =          1.000000E0  /  Real = tape*BSCALE + BZERO
BZERO   =          0.000000E0  /  Real = tape*BSCALE + BZERO
DATE    = '10/05/93'  /  FITS creation date
IRAFNAME= './rp30019301_im1.imh'  /  Output IRAF image name
TELESCOP= 'ROSAT    '  /  telescope (mission) name
INSTRUME= 'PSPC     '  /  instrument (detector) name
RADECSYS= 'FK5      '  /  WCS for this file (e.g., Fk4)
EQUINOX =          2.000000E3  /  equinox (epoch) for WCS
CTYPE1  = 'RA---TAN'  /  axis type for dim. 1 (e.g., RA---TAN)
CTYPE2  = 'DEC--TAN'  /  axis type for dim. 2 (e.g., DEC--TAN)
...
END
>>> fits[0].hdr['BITPIX']       # print value of BITPIX
16
>>> a = fits[0].data            # assign primary array to
>>>                             #   variable a
```

```
>>> x = fits[1].data[:][0]      # assign column 1 of extension 1
>>>                             #   (a binary table) to array x
>>> event = fits['EVENT']       # assign EVENT extension to
>>>                             #   variable event
>>> y = event[:]['Y']           # assign column Y of binary
>>>                             #   table EVENTS to array y
>>>
```

This sample demonstrates opening a FITS file and displaying the header keywords, along with the basic syntax of access the binary data tables in the file.

3. Implementing PyFITS

PyFITS is mostly coded in Python using Python classes, since this allows for the fastest development and most portability. PyFITS contains 12 classes: FBool, Fcard, Fhead, Fdata, Ffield, FHDU, FPrimary, FRandomGroupHDU, FASCIITableHDU, FBinTableHDU, FITS, and Ffile. FBool, Fcard, Fhead, Fdata, and Ffield are low level classes and provide access to header card records, data arrays, etc.. The FHDU class is a base header-data unit class. The other HDU classes inherit the basic behavior of the FHDU class. The highest level class is the FITS class itself. An associated class that allows file access is the Ffile class. Note that a FITS object does not necessarily have to be associated with a FITS file. The file is only used for permanent storage.

The one part of PyFITS that has not been coded in Python is the record array module. This module is imported by PyFITS and is used to read and write data in binary tables. During the development of PyFITS, Python did not have an efficient mechanism to read and write binary tables and a Python implementation, though usable, was not very fast or efficient. Hence, a C-extension module was written to read and write an array of records from native machine format to big- or little-endian byte formats. This is a general purpose module capable of reading multi-dimensional arrays of C-structures or records. The module can be used independently of PyFITS to read and write binary data and has links to import and export data to the NumPy module for matrix and array math.

Since this module is fully integrated into Python, it allows fast and efficient access to a binary data file using array notation. It is possible to access the data as a single record, a list of records, a single item, a list of items, or any combination of these. The module also allows items in a record to be named and then accessed by name, instead of by column or field index.

4. Conclusion

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 factor 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.

References

Pirzkal, N. & Hook, R. N. 1999, this volume, 479