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

A New CL for IRAF Based On Python

P. Greenfield, R. L. White

Space Telescope Science Institute, 3700 San Martin Dr., Baltimore, MD 21218

Abstract. We have developed a new CL for IRAF to allow us to develop scripts with capabilities not easily obtained in the current IRAF CL. The new CL is based on Python, a powerful, freely available object-oriented scripting language with an large and growing user base. We can run virtually all existing IRAF tasks with full functionality including interactive graphics and image display. Since there is a module for Python that provides IDL-like array manipulation, we expect to eventually combine the strengths of the IRAF and IDL environments into one system.

1. Introduction

We have long felt there was an important role for a more powerful Command Language (CL) for IRAF. A suitably powerful CL could become the data analysis software language of first resort, one that both astronomers and programmers would consider using first and where writing applications in a traditional compiled language such as C, Fortran or SPP would be infrequently required. Nevertheless, when necessary, it should be easy to incorporate compiled code into the CL either as a task, or by calling newly added functions within a CL program. Furthermore, the CL should be useful not only as a tool for astronomical data reduction, but also for many other tasks including accessing databases, the web, and text processing. Since we have a large investment in existing IRAF tasks it is essential that a new CL provide full access to existing IRAF task functionality in a style that current IRAF users will find comfortable.

Developing a new scripting language that satisfies these general requirements is not easy. We decided at the outset that the only practical way of developing a new IRAF CL was to adapt an existing scripting language to the task. By choosing suitably we would automatically get the wished-for attributes for free (except for the ability to run IRAF tasks).

The above requirements and the attributes of the existing CL that we wished to preserve resulted in a more specific list of requirements for an acceptable scripting language.

- A powerful but clear programming language,
- Support for object-oriented programming,
- Syntactic and functional extensibility,
- Good error handling,
- Access to a wide base of libraries (built-in and external),

- Easy access to code written in standard languages such as Fortran, C, C++, and Java,
- IDL-like data access and image manipulation capabilities,
- Easy access to GUI facilities,
- Open Source and freely distributable,
- Portable,
- Large user base and good prospects for long term support.

The popularity of IDL in the astronomical community (Shaw et al. 1999) has demonstrated that it is possible to write significant astronomical data analysis applications using an array-based interactive language. Having such a capability in a scripting language would enable it to serve for more than just scripting IRAF tasks; it would also be a language in which actual data analysis algorithms could be implemented directly.

Our requirement for object-oriented features was born out of conviction that such a capability would make a new CL environment significantly more flexible and extendible.

The requirement for a large user base stems from a natural desire to develop using software that has a large community of people supporting it. The larger the community involved, the more talent and resources are available to support the language itself as well as libraries it uses. An orphaned language presents the problem of providing support for the language and its libraries with limited resources and expertise.

2. Why Python?

One reason we chose Python as the basis for our new CL is because it satisfied these requirements. But what about its competitors? While Python is a reasonably well known scripting language, its user base is smaller than that of two better known scripting languages, namely Tcl and Perl. Nevertheless, we believe that the following characteristics dictated that Python be chosen over Tcl and Perl for our purposes.

First, we felt that readability was an essential requirement. Since we expect that the new CL will be used more by astronomers than by professional software developers, we need a language that is both quick to learn and comparatively easy to read and understand for those who cannot devote large amounts of time to becoming an expert with it. In that regard we judged Python to be far superior to Perl. Python is also better for writing object-oriented programs since it was designed to be an object-oriented language from the start—though one can write procedural programs just as easily.

While Tcl is generally more readable than Perl, its support for numeric types is weak. It is not a language we considered a natural choice for developing numerical algorithms. It also is not as well suited as Python for object-oriented programming.

Array-based numeric operations are not part of the base Python system. But there is an extension module (Numeric) which does implement an efficient array-based functionality that can provide many of the features found in IDL and Matlab.

Although Python's user community is smaller than that of Perl or Tcl, it is substantial and it is continuing to grow rapidly. It has been adopted by major organizations including Hewlett Packard, Red Hat, Industrial Light and Magic, and many others. While no language's future is guaranteed, we expect that Python will be well supported for many years to come.

3. Technical Challenges

With the adoption of Python, the primary technical challenge was reduced to being able to run IRAF tasks with full functionality. One condition that we set was that we should not require any changes in the IRAF system proper. The new CL must coexist with the existing system and CL.

Task Control and Execution. To run IRAF tasks, we must emulate IRAF task control and communications faithfully enough that tasks "believe" they are communicating with the old IRAF CL. IRAF tasks run as connected subprocesses and multiplex several I/O channels onto the executable's stdin and stdout channels. We were able to adapt an existing module written in Python to start and communicate with an IRAF connected process, and we wrote Python code to handle the multiplexing/demultiplexing of IRAF I/O between Python and the IRAF task.

Traversing IRAF Packages and Interpreting Parameter Files. To make tasks easy to use requires being able to interpret the IRAF package structure from package CL scripts (and thus being able to determine what packages are available, what tasks are contained within, and where the executables are located). We wished not to maintain any parallel configuration information but, rather, to use the information contained within IRAF itself. IRAF parameter files are an integral part of the IRAF system; any CL that wishes to use IRAF tasks must be able to interpret IRAF parameters in the same way the IRAF CL does. This includes handling typing issues, range or value checking and various other parameter attributes.

Graphics and Image Display. The existing IRAF CL manages the graphics of IRAF tasks. Retaining graphics functionality requires being able to render the graphics stream from the task and being able to handle IRAF CL gcur mode for interactive graphics. We implemented a Python/IRAF graphics kernel using Tkinter and OpenGL libraries. While image display does not go through the CL, imcur interactions do, so it is necessary for a CL to be able to interact with the image display server (e.g., ximtool or saoimage). This capability was provided by wrapping the NOAO Client Display Library (Fitzpatrick 1997) with SWIG (Simplified Wrapper Interface Generator, Beazley 2000).

CL Emulation. Without CL emulation, a number of IRAF tasks that exist as CL scripts will not be accessible in the new CL. The IRAF CL has a complicated syntax and semantics, and emulating it is not trivial. Nevertheless, we have written a CL compiler that converts CL procedures to Python using one of the Python parsing modules (SPARK, Aycock 1998).

Familiar User Interface. While Python syntax is one that most programmers will find very straightforward, it is not one that all existing CL users will find convenient. We have provided a front end to the Python interpreter that allows the use of IRAF command mode syntax as well as standard mode syntax for interactive use.

We have found Python to be both very powerful and productive. It has been one of those rare instances where we have accomplished more than we set out to do with significantly less effort than expected. We have implemented all of the above capabilities in Python itself. The only occasions we have had to resort to using C have been for Xlib manipulations not available in Tk (such as focus or cursor manipulations) and to wrap the Client Display Library. Speed has been an issue only during initialization on slow workstations.

4. Future Plans

We see the development of a new CL—which we've dubbed "Pyraf"—as having four phases. The first (completed) demonstrates that it is possible to run IRAF tasks with full functionality. The second makes Pyraf usable as a scripting language for IRAF. This involves supplying some utility functions and settling on public interfaces to IRAF tasks and environment. We have nearly completed this phase. Phase 3 makes Pyraf usable as an interactive environment for the typical astronomer. This includes supporting IRAF CL command mode syntax, providing a parameter editor (a prototype exists, see De La Pena 2000) and OS type utilities (e.g., dir, chdir, page, etc.), and logging. This phase will form the basis of our beta release (expected in December 1999).

The final phase makes Pyraf usable for IDL-style analysis and programming. While there is existing functionality that allows much of this, we feel it is not quite mature enough to meet the expectations of most astronomers, particularly those who currently use IDL. Some improvements are needed to the Numerics module to fill holes in functionality and to make it more memory efficient. A full-featured, easy-to-use plotting package is needed (there is good progress in the Python community on that front). There is also work proceeding on enhancing the FITS I/O module (Barrett & Bridgman 2000). An additional year will be needed to bring Pyraf to the point of being a practical alternative to IDL as a data analysis language.

We are already beginning to use Python for new STScI tools. Initial projects include a GUI front end for Synphot, a revised dither package, and an ACS display tool. The first release is expected in the summer of 2000.

References

Aycock, J. 1998, Proc. 7th International Python Conference, 100

Barrett, P. & Bridgman, T. 2000, this volume, 67

Beazley, D. 2000, this volume, 49

De La Pena, M. 2000, this volume, 63

Fitzpatrick, M. 1997, in ASP Conf. Ser., Vol. 145, Astronomical Data Analysis Software and Systems VII, ed. R. Albrecht, R. N. Hook, & H. A. Bushouse (San Francisco: ASP), 200

Shaw, R. A. White, R. L. & Greenfield, P. 1999, this volume, 24