The SciPy Documentation Project

Joseph Harrington (jh@physics.ucf.edu) – *University of Central Florida,* USA

The SciPy Documentation Project seeks to provide NumPy and SciPy with professional documentation and documentation access tools in a variety of formats. As our first effort, we have sponsored the SciPy Documentation Marathon 2008, whose goal is to produce reference documentation for the most-used portions of NumPy. I present an overview of current efforts and future plans.

Introduction and Motivation

A serious shortcoming of NumPy/SciPy is the state of its documentation. Not only does this hurt productivity and make working in NumPy very frustrating at times, it prevents adoption of the packages by many who might profitably use them. This means our pool of volunteers is much smaller than it might be, and that solving the documentation problem might thus enable projects of which today we can only dream.

Following a particularly difficult semester teaching a data analysis class in Python for the first time, I became motivated to begin the problem's resolution in the Spring 2008 semester. Below I discuss my motivating case, the requirements and design for a project to document NumPy and SciPy, our current (Summer 2008) timeline and plans, the people involved, results to date, our challenge to future development, and some issues to think about for the future.

I teach a course called Astronomical Data Analysis to upperclass physics and astronomy majors. The course initially used the Interactive Data Language, an expensive and somewhat clunky proprietary language that is popular in the astronomy community. The class is very challenging, and the focus is data analysis and how the physics of astronomical detectors affects data. Although most of the exercises are programming tasks, it is not a course in computer programming. Our use of programming is kept simple and even though about half the students typically have little programming experience, they manage to learn enough to write good and clear code (an emphasis of the course). It is perhaps a benefit that at this level the programs are kept simple and the focus is kept on analysis. The IDL course was highly effective. For example, students often reported having taught their advisors important aspects of data analysis of which the faculty advisors apparently had not been aware, and of becoming resources for their fellow students in external summer programs, graduate school, etc.

In Fall 2007, having moved institutions, I took the opportunity to switch the class to NumPy. I was pleased that most of the students in my class were comparably capable to those at my former institution. As resources for learning NumPy, they received the *Guide to NumPy*, the scipy.org web site, the mailing lists, me, source code, and the astronomy tutorial. They were also encouraged to work together. It was not enough, and the class suffered badly. Students often spent 2 hours to find and learn a simple routine. This prevented their rapid acquisition of the language, which the class depended upon. Despite their (and my) heroic efforts, none completed the final project, something that even the worst students had always been able to do. This included several students experienced in languages like C++. The problem was simple: the reference documentation was in many cases not even poor, it was nonexistent, and the other resources were insufficient to make the language accessible in the time currently expected of students and many other programmers. The conclusion was simple: I must either have documentation the next time I taught it or go back to IDL.

Requirements and Design

If I can't find a reindeer, I'll make one instead. -T. Grinch, 1957

I was unwilling to return to IDL, but at the same time as a pre-tenure professor with a very active research program, my time was far too constrained to take on the job myself. So, I would be leading a project, not doing one. My resources were the SciPy community, some surplus research funding, and about 8 months.

The needs were similarly clear: reference pages and a reference manual for each function, class, and module; tutorial user guide; quick "Getting Started" guide; topic-specific user guides; working examples; "live" collection of worked examples (the scipy.org cookbook); tools to view and search docs interactively; variety of access methods (help(), web, PDF, etc.); mechanisms for community input and management; coverage first of NumPy, then SciPy, then other packages.

Completing the reference documentation for the routines used in the course would be sufficient to teach the class in NumPy. The requirements for the reference documentation were: a reference page per function, class, and module ("docstrings"); thorough, complete, professional text; each page accessible one level below the lowest-level expected user of the documented item; examples, cross references, mathematics, and images in the docstrings; pages for general topics (slicing, broadcasting, indexing, etc.); a glossary; a reference manual compiled from the collected pages; organized by topic areas, not alphabetically; indexed.

Timeline and Plans

I hired Stéfan van der Walt, a NumPy developer and active community member, to spearhead the writing effort and to coordinate community participation. Since it was clear that the labor needed would exceed that of a few people, we agreed that some infrastructure was needed to ensure quality documentation and even to make it possible for many non-developers to participate. This included: wiki for editing and reviewing docs, with a link to SVN, ability to generate progress statistics, etc.; templates and standards for documentation; addition of indexing, math, and image mechanisms to NumPy doc standard; production of ASCII, PDF, and HTML automatically; and workflow for editing, review, and proofing. Work on the infrastructure took about 5 weeks and was complete by June 2008.

We decided first to address NumPy, the core package needed by everyone, and to provide reference pages, a "Getting Started Guide", and a tutorial user guide. We would also need doc viewing tools beyond Python's help() function. These would initially be web browsers and PDF readers, but perhaps eventually more specialized tools would appear. We realized that we would not write the latter tool set, but that if we provided an easily parseable documentation standard, other users likely would provide them. Finally, SciPy would need the same set of documents.

Starting all projects in parallel would dilute our volunteer effort to such a degree that the UCF course would not have the documents it would need. Many efforts would likely not reach critical mass, and would fail. Also, the count of NumPy pages surprised us, as it exceeded 2300 pages. A general triage and prioritization of the needs of the course were necessary. The surviving pages of the triage are called NumPyI below:

Date	Goal
June 2008	NumPy page triage and prioritization, infrastructure
by September 2008	NumPyI 50%+ to "Needs review" status, included in release
by January 2009	NumPyI 100% to "Needs review" status, included in release
by June 2009	NumPyI Proofed, included in release
by September 2009	SciPy 25%+ to "Needs review", included in release

Results of the first stage of the project appear below. That effort raised the question of when/if/how/by whom would the "unimportant" part of NumPy be documented? It was also not clear that an effort of tens of volunteers could write a monolithic tutorial user manual. A user manual requires continuity and coherence throughout, to a much greater degree than a collection of docstrings. One possibility would be to divide the manual into around ten chapters and to allow teams to propose to write the chapters and communicate with one another to attempt to keep the document as a whole uniform and coherent.

People

The core documentation team included: Joe Harrington - organization, plans, standards and processes, funding, whip; Stéfan van der Walt (on UCF contract) - Reliable and available expert, interface with SVN, PDF and HTML document generation, standards and processes, wiki, chief writer; Emmanuelle Gouillart - wiki, wiki hosting; Pauli Virtanen - wiki, writing; Perry Greenfield - numpy.doc pages

We began the community effort with the Summer Documentation Marathon 2008, in reference to the "sprints" sponsored periodically to address particular needs of NumPy development. Volunteers immediately signed up on the documentation wiki and began writing docstrings, and Stéfan finished the UCF priority list early in the summer. As of this writing (August 2008), the following have signed up on the wiki:

Shirt	Name	Shirt	Name
y	Bjørn Ådlandsvik	y	David Huard
	René Bastian	y	Alan Jackson
	Nathan Bell	y	Teresa Jeffcott
	Joshua Bloom		Samuel John
	Patrick Bouffard	y	Robert Kern
	Matthew Brett		Roban Kramer
	Christopher Burns		Vincent Noel
y	Tim Cera	y	Travis Oliphant
	Johann Cohen-Tanugi		Scott Sinclair
	Neil Crighton	y	Bruce Southey
	Arnar Flatberg		Andrew Straw
	Pierre Gerard-Marquardt	y	Janet Swisher
y	Ralf Gommers		Theodore Test
y	Keith Goodman		James Turner
y	Perry Greenfield	y	Gael Varoquaux
y	Emmanuelle Gouillart	y	Pauli Virtanen
y	Joe Harrington		Nicky van Foreest
	Robert Hetland	y	Stéfan van der Walt
y=	earned a T-shirt! Hooray and thanks!		

During the middle of the summer, we decided to offer an incentive to attract more writers. Teresa Jeffcott at UCF produced a humorous graphic, and writers contributing over 1000 words or equivalent work in wiki maintenance, reviewing, etc. would receive one at the SciPy 2008 conference or by mail. A number of writers signed up in response, and the offer remains good until we withdraw it. We encourage even those who simply want the shirt to sign up and write 1000 (good) words, an amount that many volunteers have contributed in a single week.

Results - Summer 2008

As of this writing, the status of the NumPy reference documentation is:

Status	%	Count
Needs editing	42	352
Being written / Changed	33	279
Needs review	20	164
Needs review (revised)	2	19
Needs work (reviewed)	0	2
Reviewed (needs proof)	0	0
Proofed	3	24
Unimportant		1531

The quality of the finished documents is easily comparable to commercial package documentation. There are many examples, they work, and the doctest mechanism ensures that they will always work. These should not be the *only* tests for a given function, as our educational tests do not necessarily exercise corner cases or even all function options. The PDF document has 309 pages. NumPyI should double that number, and it will triple from there when all documentation is complete. As of this writing, the doc wiki is the best source for NumPy documentation. It is linked from the scipy.org website under the Documentation page.

A Challenge

We would like to kill the doc problem going forward. "Normal" professional software projects, free or not, write docs and code together. *Good* software projects write docs *first* and use them as specifications. NumPy had to start fast to unite Numeric and numarray, but that era is now over (thank goodness). We thus challenge the developers to include no new functions or classes in NumPy or SciPy without documentation. We further urge those patching bugs to consider the lack of a docstring to be a major bug and to fix that problem when they fix other bugs. Those fixing bugs are particularly knowledgeable about a function, and should easily be able to write a docstring with a limited time commitment. This is particularly vital for the "unimportant" items, where there is likely to be less interest from the wider community in completing the documentation. Of course, we never wish to provide a barrier to fixing bugs. We simply state the case and ask developers to use their judgement. Likewise, if reasonable docs come with a new contribution, the details of doc standards compliance can be waived for a while, or the contribution can be accepted on a SVN branch and held until the docs pass review.

Going Forward

Since NumPy is fundamental, its reference pages come first. Then we will put SciPy on the doc wiki.

We need more writers. NumPy developers should consider addressing the "unimportant" pages, as others may lack the knowledge or motivation for doing them. The authors of specialized sections of SciPy should contribute docs for their work (initially in SVN). In some cases they are the only ones able to communicate effective use of their packages to users.

We may implement separate technical and writing reviews. It may be best to limit the reviewers to a small group, to maintain consistency and a high standard. Certainly in no case should a reviewer contribute text to a docstring, as all parts of each docstring must be seen by at least two brains.

We may need a different approach for the user guides, and we would like to start work on tools, SciPy, and user guides. For this, we need still more volunteers. So we ask you, dear reader, to go to the doc wiki, sign up, and WRITE! The time you spend will be greatly exceeded by the time you save by having docs available. Many thanks to all contributors!