Planetary Science Software Survey

2019

introduction

In Summer 2019 (late June-early September), collaborators[1] from organizations including the University of Colorado, USGS, Johns Hopkins University, the Naval Research Laboratory, Million Concepts, and OpenPlanetary conducted a survey[2] about software needs in the planetary science community, particularly in relation to Python use.

The results (58 responses) indicate that Python has achieved widespread adoption in planetary science, and may well now be the most commonly-used language in the field. They also point to many shortcomings of the current planetary science software ecosystem, including but not limited to Python. These results support contentions that software development efforts that seek to serve the maximum number of stakeholders in the community should be focused on Python. They also suggest a number of productive avenues and principles for near-term development efforts to improve usability, address common user pain points, and develop a more robust, well-integrated software ecosystem.

language adoption and use

Python is the most commonly used scientific language among our respondents. Over two thirds (42) of our respondents regularly use Python in their scientific work. The next most common languages are IDL (21), MATLAB (14), and FORTRAN (10). (Many respondents reported regular use of more than one programming language.)

python and array programming

The next most common languages are all array-based. In other words, Python is by far the most popular language among our respondents that is not first and foremost a numerical analysis language. Furthermore, Python libraries that provide strong support for array programming are very popular among our respondents. Numpy and scipy use are nearly ubiquitous among regular Python users, and pandas use is extremely common (about three

[1] Michael Aye, Angeline Burrell, Trent Hare, Jason Laura, Andrew Annex, MaryJon Barrineau, Michael St. Clair, and Chase Million contributed to survey design. This report was composed by Michael St. Clair.
[2] "Planetary Science Software Survey." 2019. docs.google.com/forms/d/1KHuQD4EhHq_w1_wudkunZhDJSH1SxnpbOcWSBwebrEU/

quarters of users). Even some respondents who do not regularly use Python report occasional use of these libraries.

This suggests that Python may be an especially appealing language because it is a general-purpose, high-level language with a very broad ecosystem that also has robust support for array programming. The central position of array programming in scientific computing generally and planetary scientists' use of Python in particular has major implications for community developers. APIs that readily follow numpy, scipy, and pandas idioms and interfaces with their data structures are likely to have the easiest learning curve for new users and the widest interoperability with existing workflows. Many Python users among our respondents complained that Python libraries with special support for planetary data have inadequate integration with other Python libraries or fail to offer a Pythonic user experience.

other libraries

We also inquired about the use of many other Python libraries we expected to be commonly used in planetary science (including space for write-ins). The most commonly-used libraries other than numpy, scipy, and pandas were astropy (21), gdal (17), spiceypy (14), scikit-image (11), and scikit-learn (9). Developers should consider capabilities, deficiencies, and conventions of these libraries when planning new software efforts. Furthermore, the success of spiceypy, a Python wrapper for the NAIF SPICE toolkit, testifies to community demand for highly usable Python wrappers for legacy planetary science tools built in other languages.

jupyter notebook

Two-thirds of the regular Python users among our respondents use Jupyter at least "occasionally," but only about a third use it "frequently" or "all the time." (Several respondents who do not regularly use Python also report occasional use of Jupyter, perhaps with Julia or R, or with analysis notebooks prepared by other parties.) These results indicate that, while Jupyter is well-known and widely-used in the community, it is not a part of everyday workflows for most Python users. Despite their high popularity in the Python developer community, notebook-based documentation and examples should perhaps not be considered a mandatory prerequisite to code release or necessary for code adoption.

anaconda

Over three-quarters of regular Python users report using the Anaconda distribution of Python. Several users who do not regularly use Python also stated they used Anaconda,

perhaps indicating use of Anaconda for R, the presence of an Anaconda installation in a shared working environment, or attempted and rejected adoption of Python via Anaconda. Developers should consider it likely but not certain that their users will be working in an Anaconda environment.

pain points

data access

We asked about 'time sinks' in working with planetary data in Python. Most of our respondents noted at least one task related to data ingestion. Over half of respondents listed PDS metadata handling and data ingestion in particular. The myriad data access methods, search interfaces, and formats used by planetary science data sources--even across different nodes of the PDS--present significant burdens. Wrappers or interfaces to planetary data sources are in high demand. Tools for metadata parsing or management of locally-mirrored data would also help mitigate these burdens.

More broadly, provision of interfaces that create interoperability across many data sources--most obviously as RESTful web service APIs--would be enormously beneficial to the community. Python is an excellent language choice for such projects due to powerful frameworks like Flask coupled to robust support for tools and standards such as OpenAPI and Postman. However, creation of such interfaces is a difficult and time-consuming proposition, and would likely require support from many community organizations.

data handling

About half of Python users reported that their Python experiences suffered from difficulty handling planetary data, including many types of GIS data, multispectral data, FITS images, and even keeping track of leap seconds. Tellingly, one respondent who identified as a college educator simply listed "opening up new data" as a major problem in teaching Python to student geoscientists. Several non-Python users also listed issues of this type (such as reading and writing CDF files), presumably as reasons they avoid Python. Development and distribution of Python tools with strong, idiomatic support for planetary data is crucially valuable to the community.

Over half of our respondents also reported spending a great deal of time on tooling for data exploration. Although this is an issue that transcends particular languages, Python, due to its popularity in commercial data science and data visualization, has many user-friendly data exploration tools that could be relatively easily extended to planetary data exploration. In

some cases, improved interfaces for data sources could also improve data exploration, for instance via easy access to PDS browse products.

user experience

Finally, roughly half of our Python-using respondents -- and some of our non-Python-using respondents -- reported problems with high learning curves or generally poor user experience. Respondents often linked these issues to packaging and software distribution ("dependency hell," "feels like beating your head against a wall"), library interoperability, and simple software discovery. Planetary science developers cannot, of course, fix every issue with the broader Python ecosystem. However, continued cooperation and outreach, along with the development of shared standards and libraries, can help mitigate these problems within the community.

limitations and future work

This survey's primary limitation was that responses were primarily self-selected in response to solicitations at PDW (the Planetary Data Workshop) and in PEN (the Planetary Exploration Newsletter). Furthermore, responses were anonymous, and we avoided collecting potentially identifying information beyond very general questions about professional time expenditure. It is therefore unclear what cross-section of the planetary science community responded to the survey. In particular, it seems likely that Python users and people with strong opinions about software were overrepresented among respondents.

We believe its implications for developers are nevertheless generally valid, in part because highly motivated and technically adept users often influence the development practices and tool choices of less experienced users. However, we recommend further user research in this area (beyond the invaluable feedback developers continually collect). More formal in situ user experience research or survey sampling methods might be especially useful, although time and expense are significant barriers to these types of work within a relatively small community like planetary science.