Introduction: The NAIF has long supported distributions of the SPICE library for Fortran, C and proprietary interpreted languages such as IDL and MATLAB. With the growing use of open source interpreted languages, such as Python in the scientific community [1], the need for a Python wrapper or interface to SPICE has led to the development of several open source Python wrappers for SPICE by motivated individuals and groups. One of these python wrappers is called SpiceyPy [2], which is licensed under the MIT software license and implements a majority of the CSPICE API. SpiceyPy is available on the Python Package Index (PyPI) and the git repository is hosted at https://github.com/AndrewAnnex/SpiceyPy. What follows is a discussion of SpiceyPy and its impact so far in the scientific community.

History: Python support for SPICE has been in development in official and unofficial capacities since as early as 2007 [3]. Although the official version has yet to be released by the NAIF, unofficial versions such as PySPICE first became available in early 2010 [4]. PySPICE was implemented via a custom parser that generated wrapper functions using the Python/C API, which created compatibility issues when attempting to support newer Python versions (Python 3). The desire to have cross-version and cross-platform support led to the initial development of SpiceyPy by the first author of this abstract in early 2014. Around the same time, a project by Philipp Rasch called Spiceminer [5] followed a similar trajectory in using ctypes but focused on a higher-level API that used a small subset of SPICE functions, but importantly Spiceminer implemented the CSPICE Cell data structure as well as code that downloaded the appropriate SPICE distribution for users system during installation which are noted in the code of SpiceyPy and some SpiceyPy specific changes have been contributed back to the Spiceminer codebase as both projects are MIT-licensed. After two years of casual development, SpiceyPy version 1.0.0 was released in the March of 2016, with a newer minor version (1.1.0) released in October of that year [2].

Implementation: CSPICE was wrapped in SpiceyPy by compiling CSPICE into a shared library on the desired platform and implementing the wrapping code in Python using the ctypes FFI package included in the Python standard library. This allowed for wrapper functions to be implemented in pure Python code that could be trivially written to be cross-compatible with legacy Python (versions 2.7.X) and newer version of Python (3.4, 3.5, 3.6). Version compatibility was also aided by using the popular Python library Six [6]. Additional benefits were realized by bridging C to Python idioms for error code and exception handling as well as implementing compatibility with the popular Numerical Python (NumPy) package for vector and matrix data types. Ease of installation was ensured by publishing the library to the Python Package Index (PyPI) and the Anaconda™ Cloud hosted by Continuum Analytics, enabling simple installation using the pip or conda install commands. By using pip, SpiceyPy can be installed by running “pip install spiceypy”.

API details: SpiceyPy includes wrapper functions for approximately 97% of the CSPICE N65 documented functions, which includes over 500 functions. Of those functions that have been wrapped, around 90% have tests implemented to validate that the wrapper functions are working as intended. SpiceyPy does not cover most of the event kernel (EK) SPICE functions due to lack of user demand and currently does not wrap most of the Geometry Finder (GF) functions which rely on C callback function. Function names in SpiceyPy are equivalent to those in the CSPICE API, neglecting the “_c” suffix in the function name. For example, “sincpt_c” from CSPICE can be called in SpiceyPy as “sincpt”. Functions like “sincpt” that return boolean “found flags” to indicate success or failure are wrapped using a Python decorator function to throw a Python exception if the flag evaluates as false to better align with Python idioms. Similarly, all functions are wrapped with a decorator function that interrogates the SPICE error handling system to migrate error messages into Python exceptions. SPICE data structures like Cells, Ellipses, and Planes are implemented, and Cells are indexable and iterable in Python like Python lists.

Development Methods: SpiceyPy was developed using a few methods that greatly aided the tasks of maintaining the codebase and supporting end users. The code for SpiceyPy is version controlled using the Git VCS system, and hosted on a public GitHub repository. Extensive tests, mostly translated from the NAIF CSPICE API documentation into Python, are included in the code repository for nearly all supported functions to demonstrate and ensure continued functionality of the package. Function docstrings are also written for each function, detailing parameter data types and a short description about the function with a link to the corresponding documentation for the C version of that function hosted by the NAIF.
The project utilizes the popular Travis CI and AppVeyor cloud-based continuous integration (CI) services to build and test new revisions of the SpiceyPy codebase on *NIX and Microsoft Windows™ operating systems for every new commit to the codebase. Code coverage statistics are reported using the Coverage alls cloud-based CI service. Documentation was written in reStructuredText (RST) format and is hosted using the ReadTheDocs cloud-based CI service [7] which generate documentation from included text files and function docstrings. Although GitHub provides a mechanism for users to issue bug reports and feature requests, a Gitter IM chat room was created for the project to facilitate quick user communication with the maintainer of the project and other users. Critically, these services are utilized at no cost to the maintainer. These services minimize the maintenance effort and validates functionality on a continual basis.

Usage by the Community: SpiceyPy has been cited in three different scientific papers by distinct authors in different journals since the 1.0.0 release in March 2016 [8-10]. This number however is not a good proxy for inferring how many users there are as some users are not using SpiceyPy for scientific publications. Gauging the direct usage of SpiceyPy is difficult without a survey of users, however some statistics are gathered by various services that provide some details. Downloads from the Python Package Index (PyPI) are logged and queryable through a publically available Google BigQuery™ table “thes-psf:pypi.downloads” [12]. A query was run against this table with results limited to installations using the pip CLI tool to filter out PyPI mirrors between the dates 03/01/2016 and 03/31/2017 which processed through 169 GB of data. The results aggregated download counts of SpiceyPy for each version of Python and each version of SpiceyPy and are presented below in table 1.

![Table 1: Counts of SpiceyPy downloads from pip grouped by Project Version and Python Version between 03/01/2016 and 03/31/2017. Dates below SpiceyPy versions indicate month of release. NA indicates where versions of SpiceyPy have not been released for that version of Python.](image)

Most users are utilizing legacy python installations (versions 2.7 and below). The usage of Python 3 is spread across the various minor releases, with the smallest amount attributed to Python 3.6 which was not released until late 2016. These results appear to show a growing use of Python 3 with a greater number of downloads for all Python 3 versions of SpiceyPy 1.1.0 than for legacy Python versions.

The Anaconda™ Cloud also displays download statistics for conda packages which show a smaller number of users utilize the conda package distributions. The number of downloads for 64-bit distributions as of April 2017 for SpiceyPy 1.1.0 are listed below in table 2, and are grouped by operating system.

![Table 2: Counts of SpiceyPy 1.1.0 downloads from the Anaconda™ Cloud. NA indicates where versions of SpiceyPy have not been released for that version of Anaconda™.](image)

From these counts, there is a notable preference for newer Python versions by Anaconda™ users on macOS™ and Linux systems. Future Directions: Development of SpiceyPy will continue as new issues are reported by end users and as new versions of SPICE are released, such as N66. The changes made by N66 will likely be added to SpiceyPy version 2.0.0, allowing versions 1.0.0 and 1.1.0 to remain tied with N65. Additional changes may include updating incomplete wrapper functions to explicitly return a runtime exception instead of providing limited or broken functionality to make it clearer to users what still needs work. A refactor of the SpiceyPy to utilize the Python CFFI package is also been prototyped which may simplify some of the function wrapper code as well as making SpiceyPy more compatible with the PyPy implementation of Python [12].

Conclusions: SpiceyPy is a mostly feature complete open source wrapper for CSPICE implemented by private persons outside of the NAIF. It is utilized by several researchers in various fields and is beginning to appear in the scientific literature. SpiceyPy will continue to be developed upon periodically as the NAIF releases new versions of SPICE.


Acknowledgements: I would like to thank the multiple users and contributors to SpiceyPy for all of their comments, suggestions, bug reports, and commits which improved the project markedly.