Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research xxx (xxxx) xxx

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

Python in Heliophysics Community (PyHC): Current status and future outlook

Julie Barnum [a,*], Arnaud Masson [b], Reinhard H.W. Friedel [c], Aaron Roberts [d], Brian A. Thomas [d]

[a] Laboratory for Atmospheric and Space Physics, University of Colorado Boulder, 1234 Innovation Dr, Boulder, CO 80303, USA
[b] Telespazio UK for the European Space Agency (ESA), European Space Astronomy Centre, Camino Bajo del Castillo s/n, 28692 Villafranca del Castillo, Madrid, Spain
[c] Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[d] National Aeronautics and Space Administration (NASA) Goddard Space Flight Center, 8800 Greenbelt Rd, Greenbelt, MD 20771, USA

Received 31 January 2022; received in revised form 20 September 2022; accepted 1 October 2022

Abstract

It's been four years since the formation of the Python in Heliophysics Community (PyHC). In that time, the community has made great strides towards embodying and implementing the ideals of a "Heliophysics Framework" put forth by Burrell et al. (2018). Specifically, the components of such a framework include: 1) centralization of current Python packages, 2) increasing accessibility and connectivity of these projects, 3) consideration of software attribution issues, and 4) the establishment and implementation of best practices and standards for code development. We describe the manner in which, and to what extent, PyHC has realized these four tenants. We then set forth suggestions for advancing PyHC's efforts, including ways in which we can improve our information architecture, how we can grow our community, both in terms of project sustainability and usage, as well as the social component of the community itself, how we can improve PyHC package integration, and finally, non-Python library considerations. The suggested improvements and additions therein advance PyHC's mission and strategic goals, while helping better integrate PyHC into the broader Heliophysics and Space Weather community efforts.

Keywords: Heliophysics; Open source software; Python; Software interoperability; Software sustainability; Information architecture

1. Introduction

An increasingly popular method for data analysis involves utilizing open-source software. One of the most widely used programming languages for developing open-source software tools is Python (van Rossum, 1995). Python is popular due to its "community-driven, open-source language, with a broad spectrum of well-developed packages" (Burrell et al., 2018). Several packages in the heliophysics and space weather community had begun incorporating Python into their workflow for software tool creation. However, much of that software development has been done as part of isolated, individual efforts. This led to a preponderance of Heliophysics Python tools that overlapped in functionality, increasing tool redundancy incidences, tools that were not well-integrated or interoperable, tools that employed varying levels of standards (if any were specifically applied), and tools that lacked community involvement in the development process. In short, the Heliophysics and Space Weather Python community needed a unified framework

* Corresponding author.
E-mail address: Julie.Barnum@lasp.colorado.edu (J. Barnum).

https://doi.org/10.1016/j.asr.2022.10.006
0273-1177/

Please cite this article as: J. Barnum, A. Masson, R. H. W. Friedel et al., Python in Heliophysics Community (PyHC): Current status and future outlook, Advances in Space Research, https://doi.org/10.1016/j.asr.2022.10.006

to integrate projects and encourage collaboration between developers(McGranaghan et al., 2017).

Burrell et al. (2018) described an initial plan for carrying out the above desire. The ideas expressed therein were based largely off of Astropy's successful framework (Astropy Collaboration et al., 2013). Burrell et al. (2018) stated that such a community would be achieved through centralization of current Python packages, increasing accessibility and connectivity of said projects, addressing software attribution issues, and the establishment and implementation of best practices for code development, as well as code standards. Via contending with these areas, the community aimed to help make Python software tools more visible and accessible, encourage communication between developers, reduce overlapping functionalities and duplication of code, all while encouraging better integration and interoperability amongst Python heliophysics packages and more easily identifying the community's future technical needs. Thus, the "Python in Heliophysics Community" (PyHC) was born in 2018.

PyHC is nowadays the overarching organization supporting the development of five core python packages that conform to the PyHC standards. These packages cover subdomains of heliophysics including solar physics, solar wind, magnetospheric, ionospheric and atmospheric physics. Tens of Python packages are also connected to PyHC and listed on the PyHC website.

This paper is composed of two main sections. The first one describes the current status of PyHC including PyHC community activities and how PyHC may be regarded as an Heliophysics Framework (Section 2). The following section focuses on possible improvements of PyHC, starting with its information architecture including: project documentation, website discoverability and community resources (Section 3.1). Possible improvements of PyHC software usage and sustainability, as well as community engagement, are then presented (Section 3.2). The last two subsections are related to a better integration of the various packages and their link to non-python libraries still used by the community.

2. Current state of PyHC

2.1. PyHC community activities

What began as a small group gathering in spring 2018 at the Triennial Earth-Sun Summit (TESS) (Ware et al., 2019), has evolved into a flourishing, active community. They are now 62 PyHC-affiliated projects, over 100 people subscribed to the PyHC Google mailing list, and a minimum of 20–30 people participating to bi-weekly telecons. Seven well-attended bi-annual meetings in the fall and spring have already been held with active participation, and a workshop in summer 2021 brought together PyHC leadership to discuss future technical priorities for the community. More recently, the first PyHC Summer School was held in June 2022.

The PyHC telecons (kicked off by the gathering at the TESS) have covered a wide range of subjects, including topics such as PyHC future technical priorities, project updates, introductions, and demos, discussions surrounding potential funding or publication opportunities, and learning about communities outside the PyHC (introducing opportunities for collaboration). The bi-annual meetings resulted from discussions during said telecons, where members expressed a desire to have a meeting to "clarify the path forward and establish priorities" (Ware et al., 2019). These meetings generally include some project updates, outside speakers relevant to the theme of the meeting, hackathon sessions wherein developers get together and dig into coding problems simultaneously, and, when able to host in-person meetings, unconferences (wherein the community holds discussions based on agreed upon topics that arise during the rest of the meeting). The very first of these meetings in fall 2018 resulted in the PyHC Standards (Annex et al., 2018), which guide community activities and project development; see Section 2.2 for further information on the PyHC standards.

PyHC has also increased its presence in meetings and conferences outside of those organized by PyHC. Each year since 2018, PyHC members have organized a poster session at the American Geophysical Union (AGU) meeting. Several of the PyHC members have also attended, participated in, and presented at the International Heliophysics Data Environment Alliance (IHDEA) (https://ihdea.net) annual meetings. Key members and leaders of PyHC are also leads on the International Space Weather Action Teams (ISWAT; https://iswat-cospar.org/) associated with the Committee on Space Research (COSPAR). PyHC members have also presented and led numerous other meetings and hack weeks. These activities help promote PyHC to the wider Heliophysics and Space Weather community.

2.2. PyHC standards

The following sections reference the PyHC standards several times. To create a clearer understanding of what these standards are, and how they fit into the grading system outlined in Section 2.3.4, the standards are summarized in Fig. 1 below. These standards and their definitions are pulled and abbreviated directly from Annex et al. (2018).

2.3. PyHC as a Heliophysics framework

PyHC has expanded greatly in members, projects, and activities. Where does the community stand in terms of meeting the needs of a Heliophysics Framework, as described in Burrell et al. (2018)? To achieve this goal, the following topics were deemed as necessary to ensure PyHC packages to be useful for the community, as well as to decrease redundancy and duplicated efforts: centralization of Heliophysics Python package information, pack-

Standards Grouping	Included Standards	Abbreviated Standard Definitions
Community	Open Development	Code must be made available and developed publicly.
	Duplication	Duplication of code and functionality and forking projects into new projects is discouraged.
	Collaboration	Contributions to packages must be encouraged. Packages must provide contribution guidelines.
	Code of Conduct	Projects must adopt a code of conduct that is compatible with the Contributor Covenant and make it publicly available.
Documentation	Documentation	Functions, classes, and modules must have docstrings provided in standard conventions. High-level documentation must be provided (e.g. guides and tutorials). Documentation must be in version control with code and be in readable form.
Testing	Testing	Must provide unit tests of individual components and integration tests (tests the interaction between components; must cover most of the code). Testing coverage should be measured. Automated, system, and acceptance testing are recommended.
Software Maturity	Packaging	All code must be organized and provided as part of installable Python packages.
	Releases	Projects should have consistent/stable releases, made available through PyPI and Conda. Unstable code must have release number less than 1.0.
	OS Support	Packages must strive to support all major operating systems (e.g., OS X, Linux, Windows).
	Version Control	All code must use version control, with a strong recommendation to use a distributed version control system (e.g., git).
	Coding Style	Projects must adopt the basic style recommendations of PEP 8.
	Dependencies	Projects should import the minimum number of packages necessary.
	Binaries	Binary files should be added to the package repository only when necessary in order to keep packages as light as possible.
Python 3	Python 3	All packages must be compatible or work towards being compatible with Python 3.
License	License	Projects must provide a license. Projects should use permissive licenses for open source scientific software.

Fig. 1. PyHC standards grouping (used for project self-evaluations) with their associated standards and definitions (as defined in Annex et al., 2018).

age accessibility, software attribution, best practices, and rigor of the provided tools (Burrell et al. (2018)). We'll now address each of these topics.

2.3.1. Centralization

Burrell et al. (2018) stated that the community needed a centralized location where one could locate information related to all currently used and maintained Heliophysics Python packages. A package refers to a collection of Python modules linked together under one name. Please note that throughout this paper, projects and packages are used interchangeably. To this end, one of the first activities tackled by the PyHC was to create the PyHC website (https://heliopython.org). The website now serves as a one-stop shop for those interested in discovering everything that PyHC has to offer.

Information regarding Heliophysics Python packages is covered by the PyHC site's Projects section. Therein, PyHC members and outsiders alike can find current, open-source Heliophysics Python packages. At present, there are 62 PyHC-affiliated projects included on this page, which run the gamut of Heliophysics domains (Solar, solar wind, Magnetosphere, as well as Ionosphere, Thermosphere, and Mesosphere studies). These packages include a wide range of functionalities (e.g. modeling, machine learning, plotting, coordinate transformations, wrappers for other libraries, and CDF writers). Information is also given regarding the overall functionality of each package, contact information, a link to the package's code repository and documentation, as well as "grades" (see Section 2.3.4) associated with various groupings of the PyHC standards (see Fig. 1). See Appendix A for a listing of packages, along with a short description of each package's functionality.

Packages are split into two categories:

- Core Packages
  - Each core package provides a wide range of functionalities in their area and conform to the PyHC community standards. The current core packages are PlasmaPy, pysat, pySPEDAS, SpacePy, and SunPy. Initially, the core packages were chosen by the original PyHC organizers. Currently, core package inclusion is based on a decision made by LASP and NASA PyHC leadership, as well as the current PyHC core packages.

- "Other Packages"
  - This category comprises packages that fit within the PyHC ethos, but are more specific in their functionality.

The PyHC website also boasts an increasing and diverse number of PyHC examples on the PyHC site's Gallery section. This page includes science use case examples from PyHC packages. The gallery is powered by Sphinx (Brandl, 2021), which is a tool used for creating documentation (in particular, Python documentation). Its markup language is based in reStructuredText, a plaintext, markup syntax and parser system (Goodgerplo, 2022). Sphinx generates and displays Jupyter notebooks (Kluyver et al., 2016), which are web applications that generate and share computational documents. The Jupyter Notebooks in the PyHC Gallery use Binder links to allow an end user to interact with them. Binder links provide "an environment that runs and interactively serves ... Jupyter notebooks", which allows end users to view and interact with the notebooks in the PyHC Gallery (Project Jupyter et al., 2018). Additionally, users can download these Jupyter notebooks (or the Python source code) to their local workstation for further investigation and/or sharing with others. Effort is ongoing to create more examples (particularly as relates to inter-project integration).

The website displays information regarding the community itself including.

- Home page composed of the PyHC mission statement, strategic goals, and recent blog posts from the community.
- Documents page linking to PyHC bi-annual reports, PyHC standards, etc.
- Meetings page with links for the PyHC bi-weekly telecons, the PyHC Google calendar, links to past biannual meeting information, and related meetings (e.g. links to an upcoming PyHC poster session at AGU).
- PyHC People page (with links to members' GitHub pages).
- Contact page.

2.3.2. Package accessibility

In Burrell et al. (2018), package accessibility consists of utilizing Free and Open Source Software (FOSS) and licenses that work with FOSS. According to the PyHC standards, all code must be made available and developed publicly (Annex et al., 2018). Projects are not required to use a particular code hosting platform. However, the majority of PyHC packages code is uploaded and developed within GitHub (https://github.com/); other projects have chosen to use Gitlab (https://about.gitlab.com/). With respect to PyHC project licenses, the PyHC standards dictate that projects must provide a license. Projects should use permissive licenses for open- source scientific software (Annex et al., 2018).

2.3.3. Software attribution

Fear of plagiarism has historically been a sticking point for convincing scientists to switch to develop their projects with FOSS like Python. Burrell et al. (2018) indicated that software citation and encouraging project collaboration would help alleviate those concerns. These sentiments were echoed in the August 2021 PyHC Integration Strategy Workshop Report: the importance to PyHC of [software citation] is fourfold: 1) those interested in a career in science need scientific currency to further that career (i.e., citable work that is also cited in peer reviewed publications, not just potentially citable), 2) this gives PyHC packages an opportunity to advertise their tools, 3) it advertises PyHC, driving traffic to our community (which would ideally then increase PyHC package usage), and 4) this would demonstrate to scientists, with data, that software has a role in the scientific record (Thomas et al., 2021).

Currently, the PyHC standards do not directly state that software must be cited (Annex et al., 2018). However, various communications within the community (e.g., at telecons, at bi-annual meetings, and PyHC-related workshops) have encouraged packages to utilize Digital Object Identifiers (DOIs) to cite their software as a whole, as well as their software versions. This hopefully will encourage more scientists to move their work to FOSS, as DOIs provide a way for software development efforts to be cited in papers. The most commonly suggested and utilized service for DOI generation is Zenodo (Nielsen & Smith, 2014). Many PyHC projects already utilize Zenodo (European Organization For Nuclear Research, & OpenAIRE, 2013) to generate DOIs including HAPI (Weigel, et al., 2021a), PlasmaPy (PlasmaPy Community et al., 2018), pysat (Klenzing et al., 2021), SpacePy (Morley et al., 2021), SunPy (Mumford et al., 2021), ndcube (Ryan et al., 2021), PyTplot (MAVEN SDC et al., 2021), and OMMBV (Stoneback et al., 2021). It should be noted that most of the core packages (and a few others) are also findable through the Astrophysics Data System or ADS (Kurtz et al., 2000), where they are listed therein with their Zenodo DOI. Furthermore, those packages are also listed in the Astrophysics Source Code Library or ASCL (DuPrie et al., 2013). ASCL defines itself as a free online registry for source codes of interest to astronomers and astrophysicists, including solar system astronomers, and lists codes that have been used in

research that has appeared in, or been submitted to, peer-reviewed publications. A simple next step to better advertise these DOIs, would be the creation of a "References" page on the PyHC website.

Burrell et al. (2018) state that collaborations deal with the ethics of building upon another person's work. That is, projects need to ensure that they properly work with and cite other software development efforts. The PyHC standards state that contributions to packages must be encouraged (Annex et al., 2018). To help make collaboration opportunities easier to discover, the PyHC website has a listing of all PyHC projects under its projects page. This allows other Heliophysics developers to find existing projects to which they can contribute, leverage, and cite. PyHC projects are also given many opportunities to present on software development efforts at telecons and bi-annual meetings. One method that is increasingly popular for furthering collaborative efforts is publishing papers that cite PyHC software DOIs and other related publications. This of course is only made possible when PyHC projects create software DOIs (such as those indicated above) or publish in journals (e.g. method papers such as for HAPI (Weigel et al., 2021b) or pysat (Stoneback et al., 2018)). An even better solution is for these papers to be collaborative software tool papers. An example of this would be an executable paper. Polson (2021) is in the process of writing such a paper; this will hopefully be the beginning of many publications of that nature. Such publications should also be included in the future PyHC "References" page.

2.3.4. Best practices and rigor

Many scientists do not have a pure software engineering background, but rather learn a programming language on their own in order to carry out research. To provide guidance to scientists, Burrell et al. (2018) outlined several best practices' that scientists should consider in developing software. Many of these are mandated and/or recommended via the PyHC Standards, including specifications for packaging, OS support, version control, coding style (e.g. adopting PEP8 (van Rossum et al., 2013)), documentation (as relates to docstrings and high-level documentation), and Python version usage. As a reminder, a Python docstring is a string used to document a Python module, class, function or method, so programmers can understand what it does without having to read the details of the implementation. Also, it is a common practice to generate online (HTML) documentation automatically. The standards also give suggestions for best practices on dependency usage, avoiding code duplication (to be avoided as much as possible), and software releases (Annex et al., 2018).

Project "rigor" pertains to peer reviews of code and implementation of several forms of testing within a project's software (Burrell et al., 2018). Via the open development requirement in the PyHC standards, all code is publicly available. Any pull request made is therein open to all in the community to review and make comments. Peer reviewing is not required to be affiliated with PyHC, but it is encouraged. With respect to testing, the PyHC standards outline that packages must implement unit and integration testing, while encouraging system and acceptance testing, as well as automated testing, when possible (Annex et al., 2018). Although a mention of code coverage is included, a specific percentage is not included.

In order to inform users on how a PyHC package measures up the PyHC standards, PyHC implemented a package "grading system". To become a PyHC-affiliated package, projects have to grade themselves against the PyHC standards and show that their software either meets all the standards, or are actively working towards meeting them. The self-assigned package "grades" for PyHC standards are displayed on the PyHC website's "Projects" page as a traffic light system for six groupings of the PyHC standards (Community, Documentation, Testing, Software Maturity, Python 3 Usage, and implementation of a FOSS license; see https://heliopython.org/projects/ for a graphical representation of these groupings). Green indicates that a package meets requirements for a standard grouping, yellow means a package meets some of the requirements, and red means that a package meets little or none of the requirements. Currently, packages which do not meet all the PyHC standards are not removed from PyHC.

3. Advancing PyHC efforts

The initial work of identifying commonly-used Heliophysics Python packages, gathering key members in the Heliophysics and Space Weather community, organizing community activities to help keep the community abreast of important changes, updates, etc., as well as identifying potential areas for collaboration, integration, and potential new projects is well under way. The community created a centralized location (the PyHC website) under which information about all of the above can be found. In order to continue growing as a community, however, there are still areas of improvement to consider.

As suggested during the Mini-ISWAT Virtual Meeting Series: O2 Information Architecture meeting held in March 2021 (https://iswat-cospar.org/virtual-meeting-series_O2), a PyHC Integration Strategy (IS) Workshop was held in August 2021. This NASA-organized ISWAT workshop brought together key members of the PyHC community, as well as key figures in closely related communities, by invitation only. The workshop allowed organizers and attendees to deep dive into how to collectively advance PyHC efforts in the future with respect to technical work, prioritizing project integration in particular, where key challenges and corresponding potential solutions were ideated. This workshop produced a public report (Thomas et al., 2021), which details the outcome of the workshop. Many of the potential solutions expressed therein are excellent starting points for resolving the above-mentioned sticking points, as well as other sticking points not previously considered by the organizing committee. Moving past these issues will help improve the util-

ity of the PyHC, as well as increasing PyHC packages usage in a wider community.

Below are ideas on how to improve the PyHC information architecture, its packages' usage, sustainability, and interoperability, as well as how PyHC can play a role in integrating and retaining utility of important non-Python code. These different facets of the problem are intrinsically linked and a breakdown in any of these directly impacts the usefulness, participation in and vitality of the PyHC. These ideas are partially sourced from the PyHC IS workshop, discussions at PyHC-organized activities, and knowledge of other packages in the community.

3.1. Improving information architecture

Information architecture (IA) focuses on organizing, structuring, and labeling content in an effective and sustainable way (https://www.usability.gov/what-and-why/information-architecture.html). Well-thought-out IA helps users to easily and quickly discover information and use it to achieve a goal. For PyHC, users need to be able to locate information regarding the community, the packages involved, and information about each project including general functionality, subroutines, docstrings, metadata, taxonomy, and documentation. Gathering Heliophysics Python packages under the PyHC website was the first step to creating the PyHC IA. However, to fully flesh out this IA, additional changes with respect to project documentation, discoverability of the PyHC website and utilization of related community resources should be considered.

3.1.1. PyHC project documentation

One straightforward way to strengthen the PyHC project documentation would be to enhance project descriptions on the Projects page of the PyHC website. This would enhance discoverability and utility. The end goal is to make the PyHC website the "one stop shop" for a user to discover information on Heliophysics Python packages.

Currently, each project has a corresponding short overall project description, links to code, documentation, and websites, a contact name, and project PyHC standards grades. Additional useful information to add to the PyHC projects tables shall include.

- More in-depth description of project functionality, and what kinds of science questions could be tackled using said package.
- Information on project routines and functions (including pulling docstrings describing each subroutine's functionality). This could be pulled out into a separate table from the main listing of projects.
- An option to "request" a new functionality or analysis method. This would be an easy way to allow users to identify future work for a project, while strengthening PyHC IA through making packages more complete in their functionality.
- Example use cases for a project; at present, only a few examples are uploaded to the PyHC Gallery. The community should work to increase the number of examples therein, especially examples where multiple PyHC packages are used to achieve a goal and consider linking to them in the projects' tables. These examples would not only help cross-document current functionalities of packages in a coordinated way, but also give a direct demonstration of the synergies between these packages.
- A link to a project's README.
- Ensure all projects have official documentation on a documentation hosting platform (e.g., documentation on readthedocs; https://readthedocs.org/)
- List the PyHC package taxonomy on the projects page - this taxonomy should be agreed upon by the community and utilized in every package. Implementing these suggestions will improve a user's ease in locating package information.

Implementing the above changes would ensure that PyHC project information is easily findable and accessible. However, it would involve a consistent effort to ensure that information is kept up-to-date, potentially time consuming though.

3.1.2. PyHC website discoverability

The PyHC IA considerations do not end with updating the contents and organization of the PyHC Projects page. This information will only be useful to users if it is easily findable by major search engines like Google search, i.e., a need for Search Engine Optimization (SEO). SEO is defined as the process of improving the visibility of a website or webpage on a Search Engine Results Page (SERP) so as to make a company's website more discoverable (i.e., on the first pages, see https://www.interaction-design.org/literature/topics/search-engine-optimization).

There are various approaches that could be taken to improve the SEO for the PyHC website. One approach relates to the Python Project Documentation section, by pulling out docstring information from package functions and routines and displaying them on the PyHC website. Hence, PyHC project information would be better "exposed" to search engines, through this website. This could be accomplished through the use of Intersphinx, which links to other project documentation (https://www.sphinx-doc.org/en/master/usage/extensions/intersphinx.html). SEO keywords could also be leveraged in PyHC blogs (in addition to improving blog usage overall), within the PyHC project taxonomy, within the Gallery examples, and so forth. Projects could create links on their webpages to the PyHC website, as well as within conference and meeting presentations and reports. This would drive more traffic to the PyHC website via users of individual project pages, but would also improve the chance of the PyHC website showing in the first couple of links on a search engine result. Moreover, PyHC could increase outreach

efforts to their communities and share a link to the website at each instance. Specifically, the following examples of outreach are given within the PyHC IS Workshop report including advertising PyHC telecons, new releases, package improvements and capabilities, etc. in community newsletters, and advertise PyHC at meetings (Thomas et al., 2021). Finally, including a references page, that would mention DOIs of the PyHC software packages, as well as links to the PyHC project publications, would also potentially increase the traffic to the PyHC website.

3.1.3. Community resources

Reorganization and restructuring of the PyHC website would improve PyHC's internal IA. However, other organizations and individuals have also created tools, which, when connected to PyHC, could help improve the overall Heliophysics and Space Weather community's IA (including initial discovery of the PyHC website and all its resources, which naturally then increases their utilization by the community). Below we present a few of possible resources which could connect to PyHC: the LIKED and DIARieS resources, Helio-KNOW, and HSO Connect.

Oftentimes, the learning curve can be quite steep regarding what tools exist and how they can be used when entering a new scientific field, picking up a new programming language, or both. What is the best way to organize all information, such that it would be useful to someone in such a position? During the PyHC Fall 2021 meeting, Rebecca Ringuette presented two different solutions that would help to answer that question (Ringuette, 2021). The first was "A LIbrary KnowledgE and Discovery (LIKED) online resource for discovering and implementing knowledge, data, and infrastructure resources" (Ringuette et al., 2023a). The LIKED resource would serve a similar purpose as PyHC, as it would be a centralized location for all information in which someone new to heliophysics may be interested in discovering. This would also help scientists and developers, within the heliophysics space, to discover one another and better understand the range of available resources. The LIKED resource serves as the support for an ecosystem to simplify Discovery, Implementation, Analysis, Reproducibility, and Sharing of scientific results and environments or DIARieS (Ringuette et al., 2023b). Thus, within this ecosystem, scientists could collaborate from the discovery phase (the LIKED resource) until the sharing phase, where scientists could share their research with the community for instance via interactive research tours, dashboards, and environments. Within these two resources' connected framework, PyHC initially fits in via the "Library Software Discovery Resource" category in the LIKED resource. The DIARieS resource would then allow users to take the discovered software and carry out efficient and collaborative research.

In order for PyHC to connect seamlessly with the above resources, Ringuette suggested various minimum and preferred standards to be applied to PyHC software. Many of these standards are already covered by the PyHC standards (Annex et al., 2018). However, Ringuette proposed applying a certain code coverage percentage (minimum 80 %, preferably 90 % or higher), whereas our standards only state that our testing (code) coverage should be measured (Annex et al., 2018). Additionally, while the PyHC standards touch on collaboration, no specific mention is given to interoperability within PyHC software. The latter is a topic that the community has begun addressing (Thomas et al., 2021), while the former is something that the community should prioritize in PyHC software development, along with considering adding more clarifications for both in the PyHC standards.

Another resource which aims to improve the Heliophysics and Space Weather community's IA is the Space Data Knowledge Commons described by McGranaghan et al. (2021a). In general, a knowledge commons refers to all resources (e.g., software, data, groups) that are openly available to and managed by a community. A truly open and accessible knowledge commons is composed of three components: knowledge graphs or KG (e.g., graphical depictions of relationships between entities), which when combined lead to knowledge networks, which are then made available to the knowledge community. One such effort to create knowledge graphs that make up a knowledge network in the Heliophysics community, is the Heliophysics KNOWledge Network (Helio-KNOW) project (McGranaghan, 2021b; McGranaghan, 2022). As described on the project's GitHub repository, Helio-KNOW is the collection of software and systems for improved information representation in Heliophysics, and the commons for the community to use and collaborate through them (McGranaghan, 2022).

In a heliophysics KG, PyHC would be considered a point of software discovery, and the tools therein are the connections to studying heliophysics phenomena. PyHC would benefit from continuing to support this effort where possible via participating in Helio-KNOW led workshops and offline efforts to create KGs of Heliophysics tools; KGs created through Helio-KNOW could be linked to and/or displayed on the PyHC website. Additionally, PyHC could further leverage the use of KGs to link PyHC software to science use cases, general Heliophysics and Space Weather domains, which serves to enhance PyHC's internal IA.

A final example of a community-based resource that connects with PyHC to improve the community's IA is the NASA Heliophysics System Observatory (HSO Connect) effort (Kirk et al., 2020; Thompson, 2021). As Thompson (2021) stated, though originally created to work with the Parker Solar Probe mission, this resource now serves to create an integrated observatory out of instrument observations and model data. HSO Connect provides users with Heliophysics-related data and products through keyword filters, which connect the user with a list of results, each of which include information such as About, Demos, and Discuss (e.g. links to discussion pages like GitHub issues) (Thompson, 2021). These products include observa-

tional data, related data products, and basic tools to analyze the observations as well as providing the same access to models and simulations which explain the data (Kirk et al., 2020). So that all PyHC resources are searchable and findable via HSO Connect, the community should work together to submit all PyHC packages to this project.

3.2. Growing the community

Improving PyHC package usage, sustainability, and community engagement actually strengthens PyHC's IA, by ensuring that resources made available to the community's "knowledge commons" persist and continue to be maintained. PyHC project usage and sustainability go hand in hand; increasing a package's usage increases the likelihood of its long-term vitality, and providing a package the tools needed for sustainability will likely attract more users. Then, focusing on establishing an active, connected, and growing community inherently improves package sustainability and usage.

3.2.1. Sustainability

Development and maintenance of Python packages in the heliophysics community is often supported by Python experts funded via competitive grants. These grants are usually obtained thanks to new ideas of development. Another source of manpower comes from the scientific community, in particular from PhDs and postdocs who develop their own package or contribute to known packages, thanks to the need of their own research. Hence, the development of a python package is a bit similar to an operational archive during the course of a space mission. As long as the space mission operates, new functionalities are added to the archive together with a regular upgrade of the front-end and back-end of the archive to the latest technologies and data access protocols. But when a space mission no longer operates, the operational archive is eventually not supported anymore. However, in this case, a long-term archive facility is there to preserve the data and their distribution to the community (e.g. NASA Space Physics Data Facility or SPDF, NASA Solar Data Analysis Center or SDAC, European Space Agency ESAC Science Data Centre or ESDC, CNES/Centre de Données de la Physique des Plasmas or CDPP).

In other words, long-term funding is needed to help keeping Python projects up-to-date, implement architectural fixes, guarantee compatibility of functionalities across Python packages, not just funding new ideas. Someone to address continuous integration (CI) and continuous development (CD) in PyHC projects would for instance definitely improve the sustainability of the core PyHC packages. But more funding of the PyHC core developer team is of course not enough. As we will see in the next section, this team also needs to find ways not only to attract new users but to turn users into developers.

3.2.2. Usage

Usage is key to turn active users into contributing developers. To this end, the PyHC core team intends to explore three different ways. The first one is to improve the documentation and website. Basically, by addressing the PyHC Project Documentation and Website Discoverability issues (see Section 3.1), this will help improve user's ability to leverage PyHC and its resources. Moreover, this documentation needs to address the needs of both experts and newcomers. A second aspect to improve PyHC python packages' usage is to develop and make available more tutorials showing how packages can be used, for what science questions are they handy, including Jupyter notebooks, method papers and executable papers (e.g. Polson, 2021). For more on this, consult Section 3.2.3. A third way to improve usage is to find ways to connect to the community on a regular basis. This can be achieved through different channels such as webinars, social media, newsletters, hackathons, hands on workshops, summer school and conferences. Something only partially supported by PyHC staff due to its limited manpower. Finally, new heliophysics missions shall be encouraged by their funding agency/agencies to make use of PyHC python packages from day one. More on these last two points to be seen in the following Section 3.2.3.

3.2.3. Community outreach and engagement

The social aspect of PyHC, although more ambiguous in nature, is still an important piece in growing the PyHC. The community is more than just heliophysics Python software, it is also the people who come together to work on all the aspects of sustaining and advancing PyHC efforts. There are several ways to tackle community engagement, many of which were ideated upon during the PyHC IS workshop.

Ameliorating PyHC outreach efforts will go a long way to bettering community engagement. There are various social media tools which can help with this. For example, tutorials from the PyHC Gallery page could be shared to a PyHC twitter account. The added benefit therein is that this easily reaches the wider Heliophysics and Space Weather community. Twitch is also a great place for developers to meet up and co-work on hacking projects, or keep each other "accountable" to taking time to work on software-related papers. Additionally, there are already many other established newsletters to which PyHC could post announcements. Another low-hanging fruit to improve community engagement is reworking the PyHC Blog content. Currently it consists mainly of meeting announcements, however, other articles could easily be added, such as explanations of PyHC Gallery entries, package release/update information, new project introductions, PyHC package tutorial/demo examples, and write ups of developer answers to users' frequently-asked questions. Thirdly, the community should consider replacing, or supplementing, the current Element group chat with Helio-

nauts and Slack. The Element chat group is not well used and is not indexable by search engines. Helionauts, although currently an invite-only group, is working towards being open and searchable by search engines (e.g. Google). Finally, an easily implementable outreach option is the adoption of PyHC "Virtual Office Hours". These would be led by PyHC and core PyHC project leadership. The office hours would give the community an opportunity to meet with some of the most heavily involved PyHC members and ask questions. These office hours could be advertised through the outreach methods mentioned above. In the event of no outside attendance, the office hours would turn into co-working sessions amongst PyHC developers.

Another aspect to consider in growing the PyHC community is the PyHC Website. Many suggestions for improving the findability, ease of use, and utility of the PyHC have been given in Section 3.1.2 (PyHC Website Discoverability). Implementing those suggested changes increases the likelihood that PyHC is one of the first few search results end users see. Additionally, these changes also ensure that when searching the PyHC website, information is easily findable and accessible. These points inherently improve community engagement; a community is more likely to grow when the information proves easy to locate and useful.

PyHC tutorials on the PyHC Gallery page provide a huge potential for bringing in more users (and developers) from those outside the PyHC itself. Users are more likely to consider software that is well documented, with examples that clearly show how a package's functionality can aid in scientific research. Currently, the PyHC Gallery page only contains a few tutorials, which address few science use cases. How can this be improved? The easiest angle to attack would be generating new tutorials that show examples with multiple PyHC packages and add them to the PyHC Gallery page. The possibilities for new tutorials are limitless; these software demos need to be encouraged/-given time to be worked at PyHC meetings, etc. Doing this has the added benefit of showing a developer/scientist how to use the package to begin with, which can convert them into a package user. While generating new tutorials is a helpful first step, the long-term sustainability and maintenance of the PyHC Gallery must also be considered. Tutorials that are not reproducible or copy-able are not useful. One possible solution to this issue is containerizing code (e.g. in Docker containers). Those who submit tutorials, or another appointed individual, could also be in charge of regularly updating tutorials; this would be a more intensive approach in terms of time and money. Finally, one sticking point with submitting tutorials is that they are not straightforward to submit in the first place in the current setup. Jupyter Book (https://jupyterbook.org/intro.html) was recently suggested as a solution to this problem.

The above suggestions are excellent methods for informing the broader community on PyHC, introducing them to PyHC packages and their functionalities, and giving them examples to see how PyHC could fit into their scientific workflow. However, these members will forgo the use of PyHC if they are not able to successfully leverage PyHC software themselves. To this end, PyHC could offer 1) a PyHC Summer School (see Section 3.2.3.1), as well as 2) a buddy/partnering setup, where an expert operator in the PyHC universe would work with a new user to guide them through what is available, and walk them through PyHC software implementation in the user's setup. For example, a new user would explain their research needs, and the expert would then quickly point them at the packages that have that functionality, and walk them through using that package. This provides the potential for being less daunting or time consuming than looking through PyHC's 60 + packages for a specific functionality that may or may not yet exist. In particular, this would be incredibly helpful for early PhD students and early career scientists, especially those from developing countries who may never have heard of many of the resources and vernacular utilized in the PyHC realm.

Finally, instrument teams for new Heliophysics missions are a big source of potential PyHC community members and software users. To reach these people, attendees of the PyHC IS workshop suggested the PyHC create a permanent "PyHC Liaison for Instrument Teams" position, funded by PyHC leadership. This person would get involved with instrument teams from the beginning of a mission (or an AO itself could dictate that liaison would be a point of communication); they would provide development support, at no charge, to instrument teams, introduce them to PyHC software and its capabilities, and get them to leverage PyHC software, or at the least, ensure their software fits the PyHC ethos and works well with our existing software.

3.2.3.1. PyHC Summer School. An excellent opportunity to advertise PyHC software, while at the same time providing the chance to get demos of PyHC core and highly-used packages to put on the PyHC website, connecting with the larger Heliophysics community, and promoting the use of open-source scientific software is a PyHC Summer School. To this end, the inaugural PyHC 2022 Summer School was held 30 May – 3 June 2022, in-person in Europe, at the European Space Agency (ESA) space astronomy center near Madrid, as well as online on Zoom. The Summer School was geared towards all graduate students, early career scientists, and established scientists looking to transition to Python in the Heliophysics and Space Weather disciplines. Nearly 500 signed up to attend, with participants from 43 countries spread across North America (~50 universities across the US), South/Central America, Europe, Asia/Oceana, and Middle East/Africa. During the summer school itself there were approximately 300 people online, with 289 registered to Heliocloud, including 60 at ESA/ESAC. Some participants only followed the event through YouTube as it was live streamed. All sessions have been recorded and available online on the PyHC YouTube

channel (https://www.youtube.com/channel/UCvdq-DuLbMtecsFF7elzPG_A).

The PyHC Summer School touched a little on research software engineering and introductory Python skills (including an overview on testing in Python), but more importantly, it allowed a deep dive into PyHC core and highly used software. Each core package presented for around 3 h, providing background on the project in general, then dove into demos and tutorials, showcasing their package's functionalities, as well as how their package can be used to simply and supplement a scientist's research workflow. Demos and tutorials were provided via premade Jupyter notebooks. To the extent possible, they also took the opportunity to showcase interoperability with other PyHC software. We also had a few other non-core projects present (HAPI, Kamodo, Solar-MACH, speasy, and OMMBV), to show some of the variety of the other PyHC packages, which while valuable, are more focused in nature than the PyHC core packages.

A bonus of this Summer School was the ability to expose attendees to cloud-based research via the HelioCloud platform created and provided by NASA Goddard Space Flight Center (NASA/GSFC, 2002). HelioCloud creates a working, stable environment of all PyHC packages. This was vital to highlight the interoperability of PyHC packages. More specifically, HelioCloud provided summer school attendees with a container that implemented a default Python environment, removing the need for attendees—with potentially very little programming background, particularly in Python—to figure out resolution of dependency conflicts. HelioCloud then provided a cloud-based platform that implemented that container. Given the increasing importance of understanding how to work within a cloud environment, HelioCloud played a pivotal role in providing scientists with the opportunity to get a feel for working in the cloud, all within organized, pre-packaged Jupyter notebooks.

The feedback for the summer school was overwhelmingly positive. Many attendees stating that it was incredibly valuable to be able to hear from the developers behind the packages, see real use case examples of software, and learn from live demos and tutorials about how to leverage the software (Barnum et al., 2022). Feedback from this summer school will be used to plan future summer school efforts. Based on the success of the first rendition, PyHC is confident that continuing to hold summer schools on a regular basis will be of great benefit to the wider Heliophysics and space weather communities.

3.3. PyHC package integration

Emphasis on improving interoperability and compatibility between PyHC packages is crucial to moving forward as a community, potentially converging to an astropy-like package for heliophysics. In fact, this need is what instigated the occurrence of the PyHC IS Workshop. If PyHC packages are not able to employ common interface practices for interoperability between tools, databases, and other PyHC packages, this decreases the likelihood of package usage, funding and personnel for sustainability, and thus a breakdown in the PyHC IA, and community overall. This is also a must for their usage on online science exploitation platforms like NASA Heliocloud or ESA DataLabs.

To improve the PyHC package integration, a number of ideas were put forward at the IS workshop. First, there is a need for more tutorials making use of multiple packages—specifically, tutorials that are added to the PyHC Gallery page. Going through the process of show use cases via papers showcases not only how PyHC packages do work together but as well as identifying where sticking points/incompatibilities remain. Next, at the very least, a document highlighting where and sometimes why there are overlaps shall be produced and put forward. If possible, get the community together to come to a consensus on one method instead of several (e.g. data access to archives). Another aspect is to provide guidelines to eventually standardize/share data models, use common APIs such as the Heliophysics API where applicable. Finally, there remains a big effort to ease the comparison of numerical simulations output with observations, including data and metadata formats.

3.4. What about non-Python libraries?

Python is certainly the (current) way of the future. However, we must still contend with legacy code written in older and/or closed-source languages (i.e. proprietary like IDL or MATLAB), as well as simulation code and intensive calculations written in compiled languages. One example in the solar physics community is the SolarSoft IDL package. SolarSoft is, for instance, the only publicly available package able to calibrate some of the datasets of the ESA/NASA SOHO mission, still in operation after 25 years. Due to the longevity of the mission and the complexity of its calibration routines, an upgrade to Python was deemed unrealistic despite several attempts. In other words, these resources are still in use by the community and should be considered to ensure that they remain usable in the future. A careful consideration of these issues would strengthen PyHC by looping in software capabilities and users who may otherwise have been unintentionally excluded from the community.

Two ways are evoked here to enhance the maintenance of legacy code. One is of course to convert the code to python when feasible, or, create wrappers in Python for widely used code in outdated/less widely implemented languages. The other is to abstract away the need to understand another language. This can be done by dockerizing the code while providing documentation on how to run the container. Alternatively, the code could be stored on a cloud system like Amazon Web Services (AWS) and develop/utilize an API or Jupyter notebooks to access it.

Finally, simulation codes and computationally intensive calculations have been and are often developed in lan-

guages other than Python (e.g. C++, C or Fortran). The question is, how can the community future proof the issues arising as people move more and more towards Python? One way is to provide a Python interface, for instance through the NASA Coordinated Community Modeling Center (CCMC) Kamodo analysis suite (e.g., Pembroke et al., 2022; Ringuette et al., 2022; Ringuette et al., 2023c; see also https://github.com/nasa/Kamodo). The other is to provide a language independent option for instance by utilizing JSON WEB API type interface to abstract away the need for specific language knowledge.

4. Conclusion

The Python in Heliophysics Community (PyHC) was set up in 2018 to promote and facilitate the use and development of Python in the Heliophysics community. Since then, PyHC standards of development were set up and implemented in its five core Python libraries: SunPy, PlasmaPy, pySPEDAS, SpacePy and pysat. The PyHC website is now becoming the one-stop shop maintaining an up-to-date list of all 60 + known Python packages available in the worldwide heliophysics and space weather community. This website also provides a gallery of examples particularly useful for newcomers. PyHC has enabled to build up a community of developers working hand in hand under the PyHC umbrella.

This paper first presents the current state of PyHC, starting by evoking its community activities. In particular, a number of bi-annual meetings, hackathons and webinars are regularly organized to grow the usage of these libraries (SECTION 2.1). The PyHC standards are then reminded in Section 2.2. The following section presents why PyHC may now be regarded as a heliophysics framework thanks to its effort to better centralize Heliophysics python packages information, improve package accessibility, enhance software attribution and promote best practices and rigor. However, a number of aspects can still be improved in PyHC.

Hence, the second part of this paper focuses on PyHC future outlook. This outlook is largely inspired from the brainstorming sessions held at the ISWAT workshop on PyHC integration strategy. This workshop was organized by NASA late August 2021, by invitation only. A number of challenges were identified and the top rated four were discussed starting with PyHC project documentation. While there is already a short overall description of its project, links to code and documentation and level of compliance with PyHC standards, it was felt that additional effort shall be put on documentation including: a more in-depth description of each project functionalities, science domain application, increase of science case examples based on each but also multiple projects. Examples in the PyHC gallery shall be targeted to both experts and newcomers. Next, the PyHC website discoverability shall be improved with technological solutions identified while better advertising, particularly in the most popular newsletters in the heliophysics community. Overall, the goal is clear: increasing PyHC libraries visibility and documentation to grow not only its usage, but also its contributors, mutually benefiting from being part of an active crowd of software developers and scientists.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work is supported by NASA, under award 80NSSC22K0326, managed by the Goddard Space Flight Center.

Appendix A. The below table provides a listing of current PyHC packages (as of time of writing this paper), along with a short description of the project's overall functionality. These are directly pulled from the PyHC site's Project section (https://heliopython.org/projects/); links to the package point of contact, package code repositories, documentation, and project websites can be found on that page. The first five listed packages (PlasmaPy to SunPy) are the PyHC "core packages".

Project Name	Short Project Description
PlasmaPy	A Python package for plasma physics.
pysat	Management and analysis tool for satellite and radar data.
pySPEDAS	Tools for loading, analysis and plotting of data from various heliophysics missions and ground magnetometers.
SpacePy	Space science library for Python. Includes file I/O, time and coordinate conversions, common analysis techniques.
SunPy	Python for Solar Physics.
AACGMV2	A Python wrapper for the AACGM-v2 C library
ACEmag	Load and Plot ACE satellite magnetometer data
AFINO	A tool for finding oscillations in time series data
aiapy	A Python package for analyzing data from SDO/AIA
aidapy	A Python package to provide machine learning and statistical methods to heliophysics data
apexpy	A Python wrapper for the Apex Fortran library

AstrometryAzEl	plate scale / calibrate star imagery to use multiple auroral/airglow cameras together
Auroral Electrojet	Auroral Electrojet AE-index read and plot.
CDFlib	Read / Write NASA CDF with pure Python + NumPy, no compiling
DASCutils	Digital All Sky Camera utilities, for camera at Poker Flat Research Range and elsewhere
Digital Meridian Spectrometer	UAF Digital Meridian Spectrometer-- load and plot
enlilviz	A Python toolkit for Enlil solar wind visualizations.
fiasco	A Python interface to the CHIANTI atomic database.
fisspy	Fast Imaging Solar Spectrograph (FISS) on the New Solar Telescope.
geodata	Geophysics analysis of radar and optical systems.
geopack	Python version of geopack and Tsyganenko models
GEOrinex	Python RINEX 2/3 NAV/OBS reader with speed and simplicity, handling most RINEX formats.
GIMAmag	UAF Geophysical Institute magnetometer network data read and plot
GLOW	NCAR GLOW 0.981 aurora/ airglow model IR-VIS-UV from Python
GOESutils	Download and plot GOES satellite PNGs and high-resolution NetCDF4 by date/time
HAPI Client	Access time series data access from many sources
hissw	Easily integrate SSWIDL scripts into your Python workflow via Jinja templates
HWM-93	NASA Horizontal Wind Model HWM93 in Python and MATLAB
IGRF-13	International Geomagnetic Reference Field IGRF -- in Python and MATLAB
IRI-2016	International Reference Ionosphere 2016 from Python and MATLAB
IRI-90	IRI90-international reference ionosphere in Python
Kamodo	CCMC tool for access, interpolation, and visualization of space weather models and data
LOWTRAN	LOWTRAN atmospheric absorption extinction, scatter and irradiance model--in Python and MATLAB
MadrigalWeb	Access data from any Madrigal database.
Maidenhead	Python Maidenhead < --> WGS84 coordinate conversions, useful for crowdsourced observations
MCALF	Accurately constraining velocity information from spectral imaging observations using machine learning techniques.
MGSutils	Mars Global Surveyor radio occultation
MSISE-00	NRL MSISE-00 atmospheric model-- in Python and MATLAB
NDCube	A Python package for manipulating, inspecting and visualizing multi-dimensional contiguous and non-contiguous coordinate-aware data arrays
NEXRADutils	Download/Plot NEXRAD compositive reflectivity by date/time, for ionospheric perturbations
OCBpy	A Python module that converts between AACGM coordinates and an adjustable magnetic coordinate system based on the location of the polar cap
OMMBV	Orthogonal Multipole Magnetic Basis Vectors - Map electric fields and ion drifts, express vectors relative to a multipole magnetic field using an orthogonal basis.
POLAN	estimate true ionosphere height from ionosonde
PyDARN	A package for analyzing data from SuperDARN.
PyGemini	Python frontend for Gemini3D ionospheric kinetic + fluid dynamics models
pyglow	Upper atmosphere models and geophysical indices.
PyMap3D	Python 3D coordinate conversions for geospace ecef enu eci and more
pysatCDF	Python reader for NASA CDF, includes CDF libraries.
python-magnetosphere	Python wrapper for cxform, coordinate transformation package
PyTplot	Based on IDL tplot, plots and manipulates time series data
PyZenodo	Simple, clean pure Python 3 Zenodo API (upload, download data).
ReesAurora	Rees/Sergienko module of excitation rates, relevant to auroral optical emissions

sami2py	Run, read, and plot the SAMI2 ionospheric model.
Scanning Doppler Interferometer	Download & plot Scanning Doppler Interferometer data from PI Mark Conde's instruments.
ScienceDates	Date / time conversions used in the sciences.
SpiceyPy	Pythonic wrapper for Spice.
THEMISasi	Read & Plot THEMIS ASI 256x256 "high resolution" GBO ground-based imager data
TomograPy	Coronal tomographic reconstructions.
viresclient	Access to ESA Swarm mission products
WMM2015	World Magnetic Model 2015 from Python
WMM2020	World Magnetic Model 2020 from Python

References

Annex, A., Alterman, B. L., Azari, A., et al., 2018. Python in Heliophysics Community (PyHC) Standards (v1.0). Zenodo. https://doi.org/10.5281/zenodo.2529131.

Astropy Collaboration, Robitaille, T.P., Tollerud, E.J., Greenfield, P., et al., 2013. Astropy: A community python package for astronomy. Astronomy Astrophys. 558, A33. https://doi.org/10.1051/0004-6361/201322068.

Barnum, J., Polson, S., Thomas, B., et al., 2022. The Python in Heliophysics Community (PyHC) and Contributions towards Open-Source Software. Available at: https://www.youtube.com/watch?v=qhF1Xsd5WGQ (Accessed 7 July 2022).

Brandl, G., 2021. Sphinx documentation. http://sphinx-doc.org/sphinx.

Burrell, A.G., Halford, A., Klenzing, J., et al., 2018. Snakes on a spaceship—An overview of Python in heliophysics. J. Geophys. Res. Space Physics 123 (12), 10384–10402. https://doi.org/10.1029/2018JA025877.

DuPrie, K., Allen, A., Berriman, B., et al., 2013. Astrophysics Source Code Library: Incite to Cite! ArXiv. https://doi.org/10.48550/arXiv.1312.6693.

European Organization For Nuclear Research, & OpenAIRE, 2013. Zenodo. https://doi.org/10.25495/7GXK-RD71.

Goodger, D., 2022. An Introduction to reStructuredText. Available at: https://docutils.sourceforge.io/rst.html.

Kirk, M.S., Fung, S.F., Ireland, J., et al., 2020. HSO Connect: Creating User-driven Infrastructure for Space Science, Abstract (SH018-0008). Online presentation at the 2020 AGU Fall Meeting, 1-17 Dec. Available at: https://agu.confex.com/agu/fm20/meetingapp.cgi/Paper/753518.

Klenzing, J., Stoneback, R., Spence, C., Burrell, A. G., 2021. pysat/pysatSeasons: Version 0.1.3 (v0.1.3). Zenodo. https://doi.org/10.5281/zenodo.4950172.

Kluyver, T., Ragan-Kelley, B., Perez, F., et al., 2016. Jupyter Notebooks - a publishing format for reproducible computational workflows. In: Positioning and Power in Academic Publishing: Players, Agents and Agendas. ELPUB. Göttingen, Germany, pp. 87–90. https://doi.org/10.3233/978-1-61499-649-1-87.

Kurtz, M.J., Eichhorn, G., Accomazzi, A., et al., 2000. The NASA Astrophysics Data System: Overview. Astron. Astrophys. Suppl. Ser. 143 (1), 41–59. https://doi.org/10.1051/aas:2000170.

MAVEN SDC, Harter, B., Lucas, E., Grimes, E., Barnum, J., et al., 2021. MAVENSDC/PyTplot: Bumping version to get metadata updates into PyPI (v1.7.28). Zenodo. https://doi.org/10.5281/zenodo.5510606.

McGranaghan, R.M., Bhatt, A., Matsuo, T., et al., 2017. Ushering in a new frontier in geospace through data science. J. Geophys. Res. Space Physics 122 (12), 12586–12590. https://doi.org/10.1002/2017JA024835.

McGranaghan, R., Klein, S.J., Cameron, A., et al., 2021a. The need for a Space Data Knowledge Commons. Structuring Collective Knowledge. Available at: https://knowledgestructure.pubpub.org/pub/space-knowledge-commons.

McGranaghan, R., 2021b. Toward a Heliophysics Knowledge Commons: The Heliophysics KNOWledge Network (Helio-KNOW). Online presentation at the Laboratory for Atmospheric and Space Physics "Friends of the Magnetosphere (FOM)" Series. Available at: https://drive.google.com/file/d/1iyML4lPYyxbVso-j5LDlei5OxeM5mJJoW/view.

McGranaghan, R., 2022. Helio-KNOW [source code]. Available at: https://github.com/rmcgranaghan/Helio-KNOW.

Morley, S., Niehof, J., Welling, D., et al., 2021. spacepy/spacepy: 0.2.3 (release-0.2.3). Zenodo. https://doi.org/10.5281/zenodo.5629146.

Mumford, S. J., Freij, N., Christe, S., et al., 2021. SunPy (v3.0.3). Zenodo. https://doi.org/10.5281/zenodo.5751998.

NASA/GSFC, 2022. The HelioCloud Project [cloud environment]. Available at: https://heliocloud.org.

Nielsen, L.H., Smith, T., 2014. Zenodo overview. Zenodo. https://doi.org/10.5281/zenodo.8428.

Pembroke, A., De Zeeuw, D., Rastaetter, L., Ringuette, R., Gerland, O., et al., 2022. Kamodo: A functional api for space weather models and data. JOSS 7 (75), 4053. https://doi.org/10.21105/joss.04053.

PlasmaPy Community, Murphy, N.A., Leonard, A.J., Stańczak, D., et al., 2018. PlasmaPy: an open source community-developed Python package for plasma physics. Zenodo. https://doi.org/10.5281/zenodo.1238132.

Polson, S., 2021. Developing an Executable Paper with the Python in Heliophysics Community (PyHC). American Geophysical Union (AGU), New Orleans, LA & Online Everywhere. Zenodo. https://doi.org/10.5281/zenodo.5822784.

Project Jupyter, Bussonnier, M., Forde, J., et al., 2018. Binder 2.0 - Reproducible, Interactive, Sharable Environments for Science at Scale. Proceedings of the 17th Python in Science Conference. https://doi.org/10.25080/Majora-4af1f417-011.

Python in Heliophysics Community (PyHC), 2018. Available at: http://pyhc.org/ (Accessed 27 June 2022).

Ringuette, R., De Zeeuw, D., Rastaetter, L., Pembroke, A., Gerland, O., Garcia-Sage, K., 2022. Kamodo's Model-Agnostic Satellite Fly-through: Lowering the Utilization Barrier for Heliophysics Model Outputs. Front. Astron. 12, under review.

Ringuette, R., McGranaghan, R., Thompson, B.J., 2023a. The LIKED Resource - A LIbrary KnowledgE and Discovery online resource for discovering and implementing knowledge, data, and infrastructure resources. Adv. Space Res., under review

Ringuette, R., Engell, A., Gerland, O., McGranaghan, R.M., Thompson, B.J., 2023b. The DIARieS ecosystem – A software ecosystem to simplify discovery, implementation, analysis, reproducibility, and sharing of scientific results and environments in Heliophysics accepted. Adv. Space Res. https://doi.org/10.1016/j.asr.2022.05.012.

Ringuette, R., Rastaetter, L., DeZeeuw, D., Pembroke, A., Gerland, O., 2023c. Simplifying Model Data Access and Utilization. Submitted to. Adv. Space Res., under review

Ringuette, R., 2021. LIKED and DIARieS: Developing the NExt Generation Heliophysics Resources. Online presentation at the PyHC Fall 2021 Meeting, 26 October 2021. Available at: https://drive.google.com/drive/folders/1OEr-CoW9lAkWyHhESr1TpAaT2tlmkIMF.

Ryan, D., Mumford, S., Sharma, Y., et al., 2021. sunpy/ndcube: v2.0.1 (v2.0.1). Zenodo. https://doi.org/10.5281/zenodo.5715161.

Stoneback, R.A., Burrell, A.G., Klenzing, J., Depew, M.D., 2018. PYSAT: Python Satellite Data Analysis Toolkit. J. Geophys. Res.

Space Physics 123 (6), 5271–5283. https://doi.org/10.1029/2018JA025297.

Stoneback, R., Klenzing, J., Iyer, G., 2021. rstoneback/OMMBV: v1.0.0 (v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.5804083.

Thomas, B.A., Masson, A., Barnum, J., Roberts, A., Hans Walter Friedel, R., 2021. PyHC Integration Strategy Workshop Report. NASA. Available at: https://ntrs.nasa.gov/search?q=20210023307.

Thompson, B., 2021. HSO Connect Data and Modeling Resources Portal and Helio Hackweek 2022. Online presentation at the PyHC Fall 2021 Meeting, 26 October 2021. Available at: https://drive.google.com/drive/folders/1OEr-CoW9lAkWyHhESr1TpAaT2tlmkIMF.

van Rossum, G., Warsaw, B., & Coghlan, N., 2013. PEP 8 – Style Guide for Python Code. Python. Available at: https://www.python.org/dev/peps/pep-0008/.

van Rossum, G., 1995. Python tutorial. Dept. of Computer Science, Centrum voor Wiskunde en Informatica (CWI), Amsterdam CS-R9526. Available at: https://ir.cwi.nl/pub/5007.

Ware, A., Barnum, J., Candey, R., et al., 2019. Python in Heliophysics Community Meeting. Zenodo. https://doi.org/10.5281/zenodo.2537188.

Weigel, B., Narayana Batta, H., Faden, J., Vandegriff, J., 2021a. hapi-server/client-python: (v0.2.1). Zenodo. https://doi.org/10.5281/zenodo.5553156.

Weigel, B., Vandegriff, J., Faden, J., et al., 2021b. HAPI: An API standard for accessing Heliophysics time series data e2021JA029534. J. Geophys. Res. Space Phys. 126 (12). https://doi.org/10.1029/2021JA029534.