Python in Heliophysics Community Spring 2024 Meeting

March 11th, 2024 - March 14th, 2024

Meeting materials are located at http://pyhc.org/meetings/

Table of Contents

Participants

56 people registered to participate in the virtual Spring 2024 meeting via the meeting's Google sign up form. Meeting participants' home institutions (as indicated by responses to the Google sign up form) included the following:

- pyOpenSci
- NASA Goddard Space Flight Center
- University of New Hampshire
- LASP
- Center for Astrophysics | Harvard & Smithsonian
- Lockheed Martin
- Heliophysics Digital Resource Library
- NRL
- Institute of astronomy, Bulgarian academy of sciences
- Johns Hopkins University Applied Physics Lab
- Dublin Institute for Advanced Studies
- University of Arizona
- UC Berkeley Space Sciences Lab
- Southwest Research Institute
- European Space Agency
- University of California Los Angeles (UCLA)
- Swedish Institute of Space Physics (IRF-Uppsala)
- George Mason University
- University of Turku
- NASA Space Physics Data Facility (SPDF)
- National Solar Observatory
- Space Environment Laboratory, Egypt-Japan University of Science and Technology
- American University
- National Research and Innovation Agency (BRIN)
- Observatoire de Paris
- University of Edinburgh, UK
- Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique (LESIA)
- Lockheed Martin Space and Astrophysical Laboratory
- Observatoire de Paris - PSL - LESIA
- University of Georgia
- Aperio Software
- PSI

Meeting Overview

The PyHC spring 2024 meeting spanned four days, saw average attendance based on past meetings, and was held only on Zoom to combat in-person meeting fatigue. The first day consisted of updates from PyHC “core” packages: HAPI, PlasmaPy, SunPy, pySPEDAS, Komodo, SpacePy, and pysat. On Day 2, several NASA Heliophysics Tools and Methods (HTM) presentations were given by awardees: New Magnetometer Data for Space Physics Research from Seismological Networks and Incorporation into PySPEDAS, Radiation Belts Analysis and Modeling Library in Python, Enabling Cloud Compatibility in Heliophysics Python Software, and splot: Spatiotemporal Plotting Capabilities for Heliophysics. We wrapped up with an overview on the new Science Platforms Coordination group, and an overview of the novel Heliophysics Software Search Interface effort. Day 3 was devoted to discussions surrounding the new PyHC Enhancement Proposal (PHEP) process, through reviewing PHEP 1 and completing the first vote on it (which passed), as well as further discussions on a PyHC - pyOpenSci partnership. Lastly, Day 4 focused entirely on unconferences, which have been a proven popular method for discussing trickier topics in the community.

Core Packages Updates

For future PyHC meetings, a few general comments will be kept in mind: 1) PyHC leadership will encourage presenters to avoid long package introductions as these are meant to be short presentations with purely updates since the last meeting, 2) there are a lot of documentation updates happening in the community, which is great, and PyHC leadership will encourage packages to continue to highlight these efforts, and 3) many core packages are showing some interoperability work happening, which will also continue to be highly encouraged by PyHC leadership.

SunPy

Will Barnes and Stuart Mumford gave an update on the SunPy package. v5.1 release Nov of 2023. Many changes therein came as a result of discussions at DASH 2023. For example, solar disc occultation as function of time, computation using SPICE kernels (wrapper of SpiceyPy package), coordinate frames related to Earth’s magnetic dipole, support for GONG synoptic data, new method for determining visibility of Helioprojective coords, and improved support for PSP WISPR data. SunPy also saw seven new contributors to this release effort! SunPy is consistently bringing in new developers. A great example to other packages. Look out for v6.0 to be released May 2024.

Further, SunPy has a new affiliated package, sunkit-magex, which is the successor to pfsspy. Additionally, SunPy officially has a partnership with pyOpenSci after completing their review process. This was accepted January 18th 2024 (note that this is separate from a PyHC partnership review). Completing the review required an overhaul of the affiliated package process.

Note from this presentation - Cruft[1] is used for maintaining boilerplate for package automation. i.e. if you started with a cookie cutter recipe, how do you update according to it? Cruft stops the cookie crumbling!

PlasmaPy

Nick Murphy gave a quick overview of PlasmaPy and best-developed subpackages. Updated features (all completed by students!) include: refactored ParticleTracker (J. Roberts), classes for MHD wave dispersion relations (T. Simon), classes for 1D MHD equilibria (M. Haque), and functionality to calculate thermalization ratio for a plasma in transit due to Coulomb collisions (E. Johnson). The last one could be used, for example, to look at plasma traveling in solar wind and predict temperature ratios for different ions. There were also a few infrastructure improvements: 1) added static type checking with mypy (Using mypy[2] on an existing codebase from mypy's docs) and 2) speeding up CI checks (switching from pip to uv, written primarily in Rust).

Nick mentioned that they hosted booths in 2022 and 2023 at the APS Division of Plasma Physics meeting. Their feedback was that it was an excellent way to get in touch with students and early career scientists. Furthermore, the booths had a lot of traffic. This is a good way for PyHC packages (and PyHC as a whole) to get further involved with the student and early career community.

Similar to PyHC, PlasmaPy also has an upcoming summer school/workshop over the summer, to be held from July 29 - August 1. This will be a sequel to Plasma Hack Weeks (held 2021/2022 from pandemic times). The summer school will cover how to contribute to open source Python packages (using PlasmaPy as an example). Bryn Mawr (from outside of Philadelphia) will contribute to the workshop. Further, there is an NSF award for travel support for up to approximately 20 participants. This begs the question, should PyHC have a "How to contribute to an open source package" summer school? Alternate years with regular summer school? Add on to current summer school material?

PlasmaPy, like many other software packages, faces the issue of sustained funding. The current NSF award concludes by next spring. As such, Nick has applied for NSF successor award and is looking at other DOE opportunities. Currently, no follow-up funding exists for this important, core package in PyHC.

Nick concluded their presentation with some ways towards integration with other PyHC packages, including: astropy.units, ionization states, plasma diagnostics in space and lab, plasma science data environment, changing culture across plasma community (to which they helped organize a psychological safety workshop), and onboarding new contributors.

During the presentation, Nick shared several helpful tools with the community:

---

[1] https://github.com/cruft/cruft/
[2] https://mypy.readthedocs.io/en/stable/existing_code.html

- Pre-commit hooks
    - ruff (superseding flake8)
    - ruff-format (superseding black)
    - codespell
    - rst-directive-colons
    - rst-inline-touching-normal
    - sphinx-lint
    - check-github-workflows
- Pytest
    - pytest-xdist (run tests in parallel)
    - pytest-rerunfailures (for flaky tests)
- Sphinx extensions
    - sphinxcontrib.bibtex (for a bibliography with bibtex)
    - sphinxcontrib.globalsubs
    - sphinx_issues
    - sphinx-hoverxref (tooltip popups of links)
    - sphinx_copybutton (adds copy button to code blocks)
    - sphinx_tabs.tabs
    - sphinx_collapse
    - Sphinx_reredirects

HAPI

Jon Vandegriff gave a short HAPI package overview before delving into updates and new developments, with a reminder that HAPI is a specification: a standard interface for serving time series data, with python capabilities with HAPI.

Updates on HAPI adoption and usage include the updating of HAPI's Python server[3]. HAPI has applied for NSF funding to further update this. There were also recent and useful addition to HAPI specification: additions to 3.2 (fully backward compatible with 3.1), a catalog option to show metadata all at once for everything (optional feature), and data responses contain URIs that point to resources like images and FITS image files (not CDF files to encourage access to numbers with HAPI).

As for V3.2 and what's next, the HAPI developer meeting will occur the same day as this presentation to push merge on PR to make 3.2 official! New updates in V3.2: an ability to link datasets of different cadences in standard way, provenance considerations (listing data sources involved in generating HAPI response), federation of HAPI servers, and a JSON schema development (checking catalog and info responses, etc.).

Final note: What's the major difference between HAPI and dos2 servers? They have a similar approach. However, DAS2[4] (led by Jeremy Faden) is more complicated with more features. Can do more with it, but it is harder to implement.

[3] github.com/hapi-server/server-python
[4] https://das2.org

SpacePy

Jonathan Niehof gave an update on SpacePy[5]. SpacePy is at its essence a space science data analysis and modeling software package (magnetosphere and heliosphere primarily). A hot-off-the-press release for 0.5.0 release went up approximately five hours prior to the presentation. Included in this new version release were the following: 1) more binaries—binary wheels on all 4 platforms, including NASA CDF library, migrate from distutils to setuptools—though he still needs to replace f2py with ctypes, this is at least not a user-visible issue, 2) officially declaring Python 2.x as dead, with all focus placed now on Python 3.6, 3) explicit ISTP support for all datamodel objects, 4) Pybats updates, 5) Irbem - dose calculation model, and 7) > 100 commits, done by just over four contributors (a note to the reader that lots of issues will be answered by “check out new release”). It appears that Github discussions increased usage, which hopefully means future releases get out faster.

What is next then, on the horizon for SpacePy? 0.6 or 1.0? Per Jon, SpacePy is trying for the latter. There is a little bit of NASA support for issue responsiveness (focusing on user issues and PyHC standards). SpacePy has a couple undergrads lined up with a no-cost extension (NCE) to get those done over spring and summer. The next update will also see an ability to adapt between PyHC data objects (Jon Vandegriff of HAPI is behind a lot of this), as well as cloud file support (not just purely S3 objects, but thinking about what people want to treat like files), 3D SWMF support (thanks to Dan), and finally, a documentation overhaul! That last point includes separating api and usage docs and leveraging more autodoc (nicely integrated with numpy docs).

Questions and comments from the question and answer portion of the presentation:

- Why not leverage the fact that binaries on github actions automatically on CIbuild?
    - Didn’t quite work for their purposes.
- Will SpacePy collaborate with or leverage the fsspec package?
    - This is on the radar; it came up during a conversation last spring. fsspec has a similar approach. It’s just a matter of getting their heads around the use case and ensuring it's the way to go.
- PyHC software environment should be significantly simpler with installing SpacePy now!
- SpacePy may be able to have binary wheels that are not python version dependent in the future with future updates.

Kamodo

Darren De Zeeuw gave the Kamodo core package update presentation. Updates included: 1) Komodo has more model readers: GAMERA_GM (very complicated model, own custom tools) and VERB model reader under development to accommodate its unique data coords and interpolation needs, 2) a new model driving using Kamodo (new science application; ITM models are ‘driven’ by conditions applied at the lower and upper boundaries of the model, often by single empirical solution. Other model outputs can be processed to provide an alternative

[5] https://drive.google.com/file/d/19_kzBpfeGP_H07_2f5J9Tcad9sUXHTTv/view?usp=drive_link

driver), see Figs 1 and 2 for more information, 3) a new web interface to visualize seven of the supported models in Kamodo, available through a new web interface (all CCMC RoR can be visualized with Kamodo online without Python environment, page adapts menus for variables with different dimensions, 1-3D plots available, and each page has link to Python code that created the code), and 4) a new HAPI interface to retrieve satellite extractions from models (a lot of work from Brent Smith contributed towards this). (4) allows selection of a model and specific model run in CCMC's RoR (runs on request) as well as a satellite to fly through the model output. It uses Kamodo on the backend to read passed options, executes data extractions, and provides in a HAPI-like manner. Basically, HAPI with a twist. This is nearing completion! Online soon in the next month or two. For (3), Darren gave a live demo of the new capabilities (website output and new visualization - not yet available on web), with a note that the CCMC website now has a link with "view output in Kamodo".

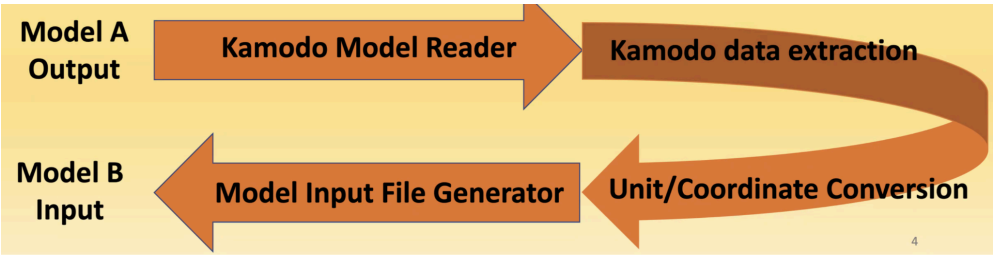

Figure 1

	GITM	WACCMX	TIE-GCM	CTIPe
SWMF	C	C	C	N
AMGeO	W	W	W	N
ADELPHI	N	N	N	N
OpenGGCM	F	F	F	F
GAMERA	F	F	F	F

Figure 2

Questions and comments from the question and answer portion of the presentation:

- Jon Vandegriff commented that it would be good to talk with Darren about the HAPI mechanism. Having a HAPI server be the output of a service request is something HAPI has thought about a little, but it would be great to learn from Darren's experience doing this.
    - Jeremy Faden noted that they are also working on a caching mechanism with HAPI, which could be useful for storing these short-lived servers' output.
- It would be great to use Kamodo as well to pull runs on demand results from CCMC. How straightforward is it to get a reader from mass output into Kamodo?
    - This is not currently on the next to-do list in Kamodo; it comes down to what the data structure is (e.g. if data output is in straightforward structure, it's much easier to implement).
- Lots of efforts doing similar things with streamviewer, tracers, etc. Is there a way to work together more effectively?

    - Hard with a data tracer that works with all objects. Lutz Rastaetter has an old tool in C that they're thinking of modifying to work with Kamodo, but if there's shared usage for tracing algorithms for more packages to use, Komodo would be interested.

pySPEDAS

Jim Lewis gave the pySPEDAS update presentation. He noted that pySPEDAS is a big group of developers; further, there's a steady stream of contributions from anonymous benefactors.

pySPEDAS has put immense work into Improving the onboarding experience. A documentation scrub is in process. pySPEDAS is all hands on deck for docstrings (with embedded examples), bringing things up-to-date, type hints, renaming things to better match IDL SPEDAS (improves AI-assisted IDL, and thus, Python porting), added Jupyter notebooks and updated existing notebooks to better work with Google colab, and added libs tool (input wildcard expression, look up matching routines). Currently, pySPEDAS is working on a new user installation and setup guides on spedas.org. The overall goal for these updates is to provide the best possible support for a potential influx of new users from upcoming PyHC summer school.

pySPEDAS made improvements to their plotting functionalities, including addressing issues, especially with spec plots (when Y-axis binning changes over time), line plot symbol shapes and colors, and pseudo variable debugging (plotting several quantities on a plot panel; this includes a magnetic field model and tracing with the geopack package). They have also created a new analysis tool, FOTE magnetic null finding; this update provides support to missions with tetrahedral spacecraft formations (e.g. MMS, Cluster). That is, pySPEDAS can estimate the local field and its 3D gradients at the center of formation and use a first order Taylor expansion to infer the location and topology of nearby magnetic null points (similar to curlometer technique). Lastly, pySPEDAS is in progress with creating access to magnetotelluric data sets via the MTH5 interface. In fact, Alexander (Sasha) Drozdov will present on that HTM effort on Tuesday of this meeting.

Future development plans for pySPEDAS (i.e. by the 2024 summer school) include plotting improvements (orbit plots especially), refactoring code to make navigation easier/more intuitive, and resolving some packaging issues ("pip install pytplot" gives wrong package; Cdflib 1.0.0 introduced some non-backward-compatible changes and pySPEDAS and PyTplot currently pinned at cdflib < 1.0.0). Future development plans (post 2024 summer school and on) include making pySPEDAS available via conda, supporting HelioCloud through allowing the opening of CDFs directly from S3 buckets, including field modeling, creating more efficient generation of 2D particles slices from 3D data at multiple time values, porting IDL SPEDAS tools to Python (especially mission-specific calibration tools), and finally, implementing a pySPEDAS GUI.

Questions and comments from the question and answer portion of the presentation:

- Many were thankful for the quick addition of the null point finding functionality.

pysat

Jeff Klenzing gave an update on pysat activities. This included a scrub of pysat documentation for all packages, updating core packages to build with pyproject.toml, leveraging standard GitHub actions workflows for testing and code coverage, and including a core testing suite for all instrument libraries. A general reminder was included that the pysat package interfaces to multiple data libraries (see Fig 3).

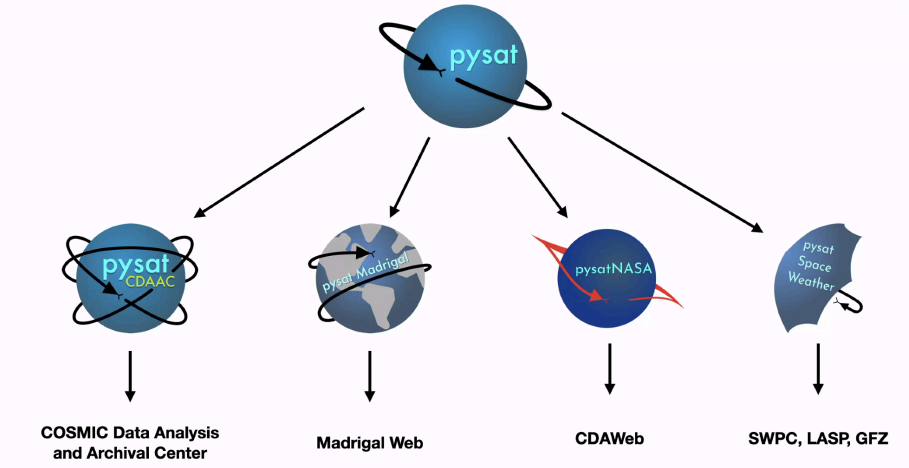

Figure 3

Specifically for the core package, the last version, 3.1.0, was released in May 2023. This improved support for xarray and improved support for exporting datasets to netCDF4. A new Release Candidate (RC) 3.2 is currently under review. Other updates for packages included within the pysat ecosystem include: 1) pysatNASA - once pysat 3.2.0 is finalized, work will begin on the 0.0.6 RC for pysataNASA in summer 2024, 2) pysatMissions - V0.3.4 was released in June 2023 (this interfaces with sgp4 propagation software), 3) pysatMadrigal has an RC for 0.2.0 currently under review, and 4) an ecosystem template package. As for operational use with pysat, an early use case would be using pysat to make end-user datasets for IVM data. This is why pysat was built this way (they had to live up to CDAAC and PSDF standards). Finally, pysat is reaching out to API folks and to see how to streamline support and coding processes.

HTM efforts

New Magnetometer Data for Space Physics Research from Seismological Networks, Incorporation into pySPEDAS - Alexander (Sasha) Drozdov

This effort has been made available via pySPEDAS, and is quickly nearing completion! The main idea of the effort is to provide easy access for this set of data. MTH5 structure is complex, but is required for those working in seismology. The PySPEDAS-MTH5 tool is available as a separate mth5 branch to be released soon.

A general question: what would it take to get a HAPI interface on top of the regularized version of the data that has been created? Essentially, they provide this access through pySPEDAS. Further, for any direct access for pySPEDAS variables through HAPI this effort is willing to make that work. MTH5 tool creates mth5 files locally, reads them in and extracts time series, etc. and puts them in the tplot variable. See Fig 4 below for how this effort simplifies MTH5 data access through incorporation into pySPEDAS.

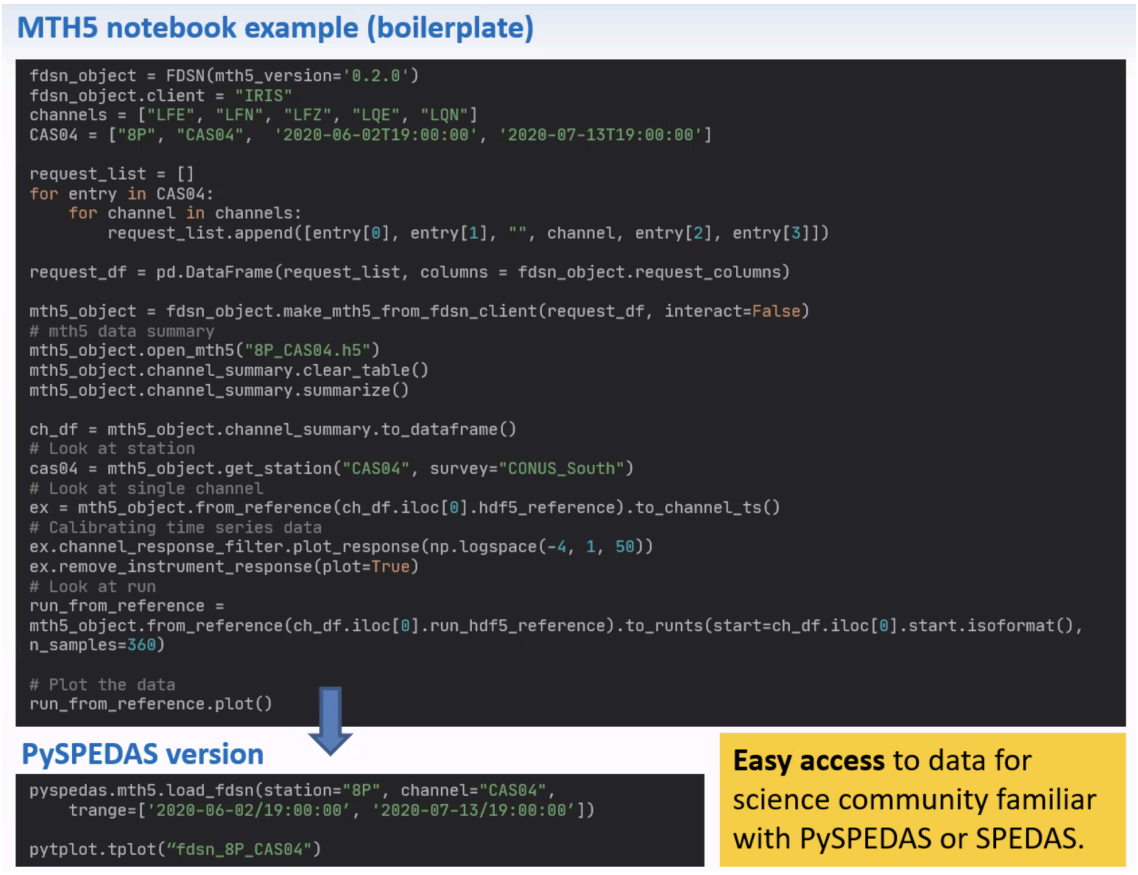

Figure 4

Radiation Belts Analysis and Modeling Library in Python - Alexander (Sasha) Drozdov

This project regarding radiation belt modeling and forecasting has just begun its efforts. Originally, this library existed at UCLA in MATLAB. It was never intended to be a public-facing software package, but rather was an internal library that grew over time. As a result, it is not well tested, nor well documented. Furthermore, it does not exist in Python. Therefore, this work aims to make the MATLAB capabilities available within Python. One can see and follow along with the project work on GitHub[6]. In the spirit of open-source development, should anyone be interested in participating in development, let Sasha know.

The library itself is lightweight with minimal dependencies (currently, only requires numpy), and is open source. As hinted at above, it is also open for contribution. Truly, this package embraces the ethos of PyHC. Note that this effort is not a direct port from MATLAB to Python, but rather aims to be a Python extension that builds on top of the MATLAB code. Documentation for this Python project can be found on readthedocs[7]. Rather than reinventing the wheel, the documentation is based off of the highly popular PlasmaPy documentation.

There are three parts to this new Python software: 1) system properties (including reusable code of basic equations), 2) modeling support (ease modeling routine tasks and facilitate an introduction to modeling for new students and students), and 3) collection of empirical models (encapsulate existing published equations for easy use in radiation belt research). The code is also tested with the unittest library, with one test class per module. The architecture of the code is built into one function in its own Python file (where the Python files are functions, and packages are modules). This creates fewer instances of merge conflicts. If any private functions are needed for the empirical models, those can be included in the top level in the file.

The following question was posed to Sash during the presentation, “is PRGEM on your radar?” PRBEM is a COSPAR body[8]. The answer is no, as PRBEM focuses on magnetic fields and has Python wrappers. This project’s idea is to extend the MATLAB library to the fullest and be lightweight. When PRBEM or SpacePy is installed, the packages are fairly heavy.

Enabling Cloud Compatibility in Heliophysics Python Software - Brent Smith

Brent Smith presented on this new effort to enable cloud compatibility in Heliophysics Python software. The PI is Wenli. Specifically, this effort addresses how most PyHC packages rely on local data storage for any type of file I/O. Currently, if one tries to run ton the cloud locally it ends up being expensive, not scalable, and not accessible to others. Hence, this HTM effort is meant for switching to use S3 object storage for this compatibility.

---

[6] https://github.com/radiation-belts
[7] https://rbamlib.readthedocs.io
[8] https://github.com/PRBEM

The effort will Initially focus on PySPEDAS, SpacePy, and SunPy due to their I/O calls. They also have data readers that are generalized and abstract for data parsing. Adding S3 capabilities will have a small impact on core developers and users. The work will be done in AWS as the limited nature of HTM timeline and funding is such that a full cloud-agnostic setup won't be possible. Work will be modeled after existing open-source community software (i.e., pandas, cdflib, astropy.io.fits).

As for S3 awareness, this work will add enhancements for PySPEDAS (access data from S3 locations), SpacePy (allow PyCDF module to load data when given S3 URI), and SunPy (extend data handlers to recognize S3 paths; ingest and save data to S3). Once implemented, these software packages will have access to mission data already on S3 (SDO and MMS), leveraging cloud computational scaling and resources.

Questions and comments after the presentation included:

1. Has h5netCDF (used by Xarray) been considered?
    a. That has S3 capability pulled in, but there are problems with netCDF3. Unsure on CDF support at this time. Further, some PyHC packages use cdflib, which is different from netCDF, so it may or may not be useful here.
2. What services are returning lists of items on S3. HelioCloud is already doing this; are there others?
    a. This effort is not aware of any other big S3 catalogues (SDO ML is on there). The nice thing is that one can specify which S3 bucket to link to and if it's accessible, then it can be read with PyHC packages.
3. Is there an S3-aware FITS reader out there?
    a. There is a tutorial that does this (possibly written by Will Barnes).
4. If you want to see things managed by NASA, have to go to their site[9]. It is unfortunate that there exists no master list of all data resources out there. Furthermore, different lists managed by different groups. This speaks to a more systemic problem.
5. Should PyHC use its indexing standard to make S3 listable by others? For example, cloudcatalog[10].
6. Is the s3 growth in SpacePy going to include pybats, or is that dependent on Dan Welling?
    a. That is going to be very naive. The bigger thing is getting SWMF output into hdf5. There, there's a big dependency on Dan being able to do that.

splot: Spatiotemporal Plotting Capabilities for Heliophysics - Brent Smith

splot is a new effort, with Brent Smith serving as PI. The work is motivated by taking a closer look at how we visualize data. splot is similar to the existing tplot, but is focused on the spatial

[9] https://registry.opendata.aws/sdoml-fdl/
[10] https://pypi.org/project/cloudcatalog/

component instead of time. Example of this is the Enlil model output, which shows planets and spacecraft positions in context of plasma density. More on this effort to come!

Science Platforms Coordination IHDEA Working Group - Shawn Polson

The Science Platforms Coordination group is a new IHDEA working group, formed in December 2023, headed by the PyHC Tech Lead, Shawn Polson. It was inspired by the PyHC environment, presenting at the IHDEA 2023 meeting at JHU/APL. The group meets monthly to discuss issues and actions, and plans moving forward. The current plan is to create Heliophysics software environments for the community at large (not purely the PyHC environment). The effort has begun with dockerized Python environments and will expand eventually to browser-based environments, publish paper(s) and DOI(s), coordinate software versioning, and improve reproducibility.

A few updates have been made since the IHDEA 2023 meeting. For one, further testing of the PyHC environment has been performed. Secondly, Shawn built a CI/CD for the environment with GitHub actions, which has thus far performed nominally (see Figs 5 and 6). No conflicts have *yet* been introduced from a PyHC package update. Note that PyHC is not yet requiring compatibility with this environment.

To look at the source code behind the PyHC environment, see the GitHub repository[11], as well as the docker container on docker's Hub[12].

[11] https://github.com/heliophysicsPy/pyhc-docker-environment
[12] https://hub.docker.com/u/spolson

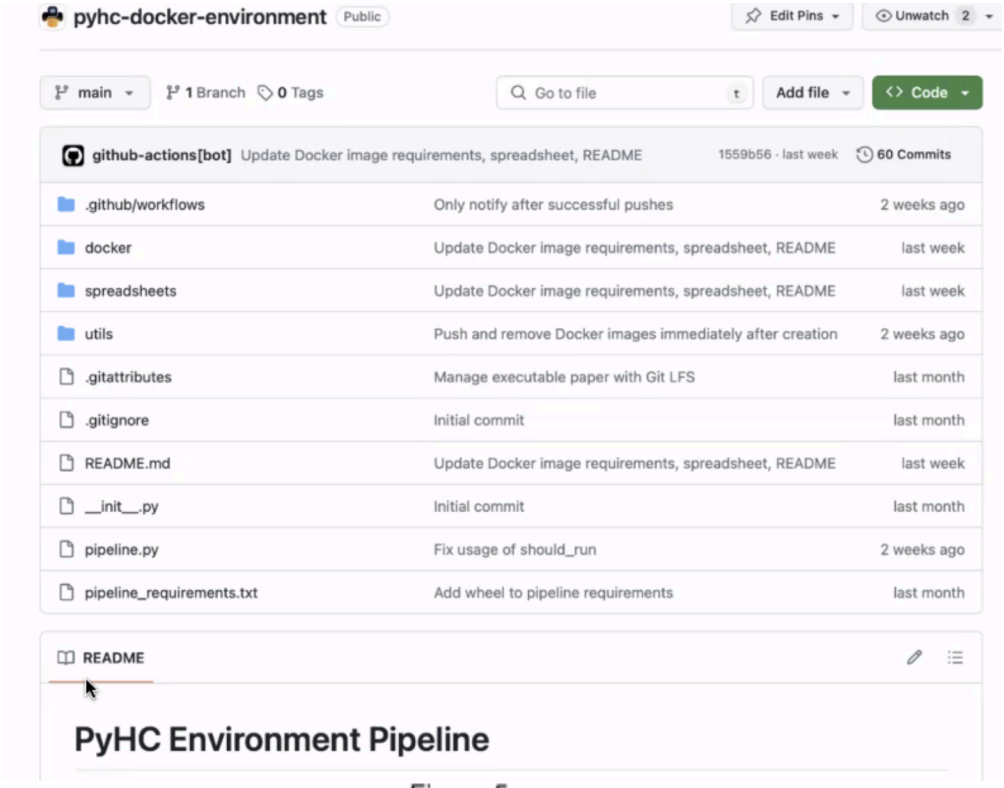

Figure 5

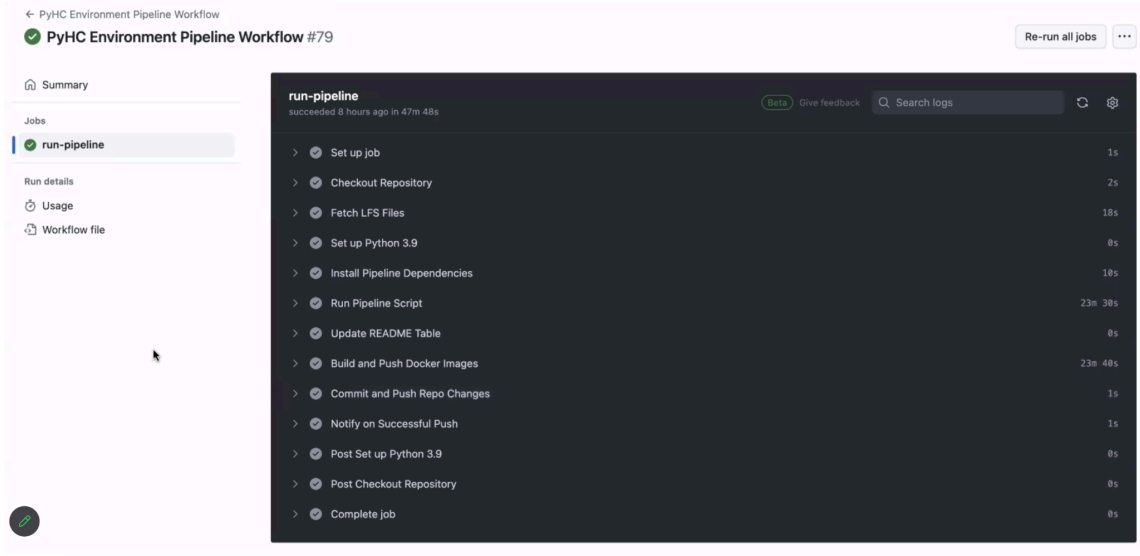

Figure 6

Questions and comments after the presentation:

- What is being used for the base Docker image? Also, will there be an imposed docker image file size so that it's minimized for use with those interactive documents?
    - The PyHC environment uses a miniconda3 base image. The compressed size is 1.33 GB. The idea is to keep the image smaller. Though, currently, more of the focus is making sure all packages are included, without as much thought to size.

- Is there a place to learn more about the working group? Or other IHDEA working groups?
    - Yes, you can go to the IHDEA web page for this[13]. Note that not all the links work, and there is currently a problem with the science platforms link. Further, there is a yearly meeting for IHDEA where working groups report and meet (i.e., the meeting at which the PyHC environment was originally presented). Last year—and this year as well—the IHDEA meeting was held adjacent to the DASH meeting (Data, Analysis and Software in Heliophysics). You can find out more information about the DASH meeting on its website[14]. Registration for the DASH meet can be found there as well[15].
- Has there been thought put to adding the tests from the package repositories?
    - That would be a much bigger task. It could be something to suggest for each package CI (e.g. a weekly build against the PyHC environment). It's technically not a hard thing to do, just a lot of busy work. PyHC is hoping to hire a student for that, but it may get pushed depending on the funding.
- There are issues with running docker under Windows. One person noted using a sandbox and installing as many packages as possible. All in all, not everyone will use docker, thus are there other options?
    - Not yet. However, eventually this will be browser-based. Currently, there are infrastructure requirements that make this quite not possible at this time.
- There is an effort to understand what packages the community uses even outside of PyHC. Please take the Science Platforms Coordination group's survey to provide an understanding of what software is used in scientific research[16].
- Would it make sense to add a CI run to test building the environment from the GitHub repositories of each package? Or perhaps just the core packages? That way, PyHC would be able to find out dependency conflicts before releases.
    - Indeed, that's a good method to do that. One thing to have eventually is to have packages do an RC before they push releases to PyPI. A requirement to accept an RC is to prove it's still compatible with PyHC environment.
    - A note from PlasmaPy that RCs are a great idea! PlasmaPy has been hoping to automate monthly beta releases for PlasmaPy. This idea has not yet been implemented due to strain on developer time and resources.
- it would be interesting for reproducibility to have a data environment. People could publish data in a certain way (e.g. on Zenodo), develop a HAPI mechanism that could read that structure and thus be permanently accessible via HAPI. There could be a data container to capture that? This is something PyHC could contribute towards (and bonus points, be citable).
    - There's a lot of support within PyHC to look towards an avenue for funding this. Further, a HAPI-Zenodo server could also help with findability for Zenodo datasets via HAPI metadata.

[13] https://ihdea.net
[14] https://dash.heliophysics.net
[15] https://www.cosmos.esa.int/web/ihdea/ihdea-dash-2024
[16] https://forms.gle/CNrFBard9KGXGZNAA

Heliophysics Software Search Interface - Eric Lopez

To promote software reuse and reduce duplication of efforts, the Heliophysics Software Search Interface (HSSI) is predicated on already-existing software. Specifically, the HSSI will leverage the Exoplanet Modeling and Analysis Center's (EMAC) interface[17].

Eric Lopez, an exoplanet scientist at NASA GSFC, and Joe Renaud, a planetary scientist and geophysicist at NASA GSFC, head the EMAC interface. EMAC's goal is to try to host and provide web access to a repository of models. With this, users can then find, store, and compare different models, how they load those, how they recruit them, etc. EMAC is looking to inspire each other and share lessons learned. Further, EMAC encompasses all things exoplanet modeling and analysis tools; there for exoplanets science codes and analysis tools. EMAC is based on the Django package (written in Python) and docker. Lastly, since EMAC is at NASA Center its resources are on the Discovery HPC cluster.

Users can publish their code to EMAC, but the EMAC team also goes out and searches for new codes as well. Over 200 codes have been submitted, almost all of the codes are Python codes! But there are also other file types. EMAC has an automated email system to reach out to potential software tool maintainers, and is bringing on recruiters to help further recruit software tools. EMAC had a virtual workshop in spring 2023 for people to present their tools.

Questions and comments after the presentation:

- Does EMAC have any overlap with the ASCL[18]?
    - Yes, it does. EMAC started talking with ASCL a lot in the last year. However, it's different in that the ASCL wants to find all open-source astrophysics codes and provide a listing. EMAC isn't limited to codes. Moreover, EMAC is chalk full of early career professionals with open source codes; these users tend to be more motivated to submit resources, but also host web applications, observatory exposure time calculators, and databases of pre-computed models (providing a citable location for those). EMAC does make efforts to list a tool if it's findable on ASCL, and vice versa.
- What is the reasoning to include stellar catalogs in a software search interface rather than on a data search interface? Would EMAC change that if starting from scratch?
    - First, EMAC sorts by science and by broad classes of tools that might be relevant to exoplanets. That's how that got included, and it's relevant! The worry is that there becomes confusion between responsibility of data archival versus software archival. Data and software must be linked to be useful to each other. But, although the subject matter experts that understand the archivals of software versus those that understand the archival of data have complimentary skills, they're still different. EMAC doesn't endeavor to make one group learn both. Additionally, EMAC does not store *everything*. They are not the place to store catalogs of host stars for example. Generally, that kind of data is handled via

[17] https://emac.gsfc.nasa.gov/
[18] http://ascl.net/

linking to a data file at a site that's archiving it. That being said, if an EMAC user knows the sort of data they need to find, but doesn't know where to look, EMAC can point them in the right direction.

- Can EMAC provide a link to its "add a resource" form to see what metadata you ask for?
    - Certainly[19].
- Regarding metadata for software cataloging, is there a schema for metadata cataloging? Some things are domain specific, but what lessons are there about that?
    - Basically, try and start from the beginning to build flexible infrastructure so things can be added as development continues. Before launching the website, EMAC had a back-end engineer make sure everything worked on servers and that all information could be parsed. Also, make sure to have a development environment and always test there first. Lastly, talk to professional programmers and follow their best practices.
- Are the metadata records based on DataCite (https://datacite.org/) in the background to some extent?
    - With DataCite, generally the metadata registration is how you get a DOI (which is what Zenodo uses). However, EMAC doesn't generate DOI. The permanent link *is* the EMAC ID. Thus, for anything published, EMAC tries to ensure there's a link of an ADS record[20].
- A suggestion was made that EMAC could move any useful information out of its Slack and into a wiki. This would greatly help new EMAC team members find information instead of having to search Slack.

EMAC and the broader HSSI session was concluded with a general discussion amongst PyHC members. PyHC members were asked what kinds of things do PyHC packages want potential users to be able to search by (e.g. short package descriptions)? This was decided to be discussed in a telecon (multiple, possibly), plus at other meetings. Several examples were given for community members to consider (HSOConnect[21], ASCL[22], EMAC[17], PyHC[23], and PyHC's current taxonomy[24]. Members were encouraged to think about their target audience and what those users would want and need (e.g. devs developing codes, as well as the audience of users trying to find codes). A final piece of advice was then presented: think about the logic of how to relate different parts of taxonomy to each other (e.g., show only "AND" logic, or include "OR").

[19] https://emac.gsfc.nasa.gov/submissions/
[20] https://ui.adsabs.harvard.edu/
[21] https://hsoconnect.hpde.gsfc.nasa.gov/
[22] https://ascl.net/
[23] www.pyhc.org
[24] https://github.com/heliophysicsPy/heliophysicsPy.github.io/blob/main/_data/taxonomy.yml

PHEP-1: discussion and voting

To begin this discussion, Jon Niehof presented a slide deck with a basic PHEP 1 overview[25,26], which is in its essence the PHEP that defines the definition of all future PHEPs. PHEP 1 was first proposed last August 2023, and discussed again at later telecons. The idea behind PHEP 1—and hence, other PHEPs—is that there is no “accepted” state; a PHEP goes straight to the final state once voted in. That is, a PHEP is immutable once final *except* things that don’t change its substance (e.g. link rot, typos, header updates from normal processes). Revisions are committed at the discretion of PHEP editors. Further, a PHEP has to be unanimously agreed upon to pass (abstained votes are allowable). The author of a PHEP should be present at votes.

Since PHEP 1 was solely focused on defining the PHEP process, there is a PHEP 2 currently on GitHub that creates a template for future PHEPs to follow for ease of creation[27].

No objections were noted to the PHEP process or PHEP 1. However, there were several questions noted:

- Someone without investment in the community could hold up votes with objections. How should PyHC handle that?
    - Code of conduct and governance needs to come into play. Objections must be concrete, actionable, and of good faith. Leadership may need to come into play as well (the PHEP editor should avoid involving themselves in this).
- Do we not need a defined governance first?
    - How do we agree on governance if we don’t have a way to agree to make those decisions? At some point, we’ll need a steering committee. ‘This PHEP refers to "PyHC leadership": individuals or groups with statutory authority over the PyHC project. This may be the Principal Investigator or other individuals or groups appropriately designated in the future, such as a steering committee.'
- Mutability versus immutability has been addressed in PHEP 1, which is great. That being said, the process is a little arduous. The trade off of well definedness is having the ability to change quickly. PyHC coud try the latter out and if the related concerns play out, the community can go through this process to revise it. For reference, the PLEPs for PlasmaPy have a section on amending PLEPs[28]. PlasmaPy has used this for the PLEP that defines PlasmaPy's top-level subpackages[29].

---

25 https://docs.google.com/presentation/d/1Fwwa6kP4Alscs8bY-Cp-agFhX0-pK4H5TgG3qFWwuoo/edit?usp=sharing

26 https://github.com/heliophysicsPy/standards/blob/a4b3f558b9ffa712324e63ff6a83325cc69e367f/pheps/phep-0001.md

27 https://github.com/heliophysicsPy/standards/pull/25

28 https://github.com/PlasmaPy/PlasmaPy-PLEPs/blob/main/PLEP-0001.rst#amending-or-superseding-a-plep

29 https://github.com/PlasmaPy/PlasmaPy-PLEPs/blob/main/PLEP-0007.rst

    - Indeed, to this point, community members can submit a PHEP about the PHEP process itself.
- Should PHEPs be reviewed yearly at the in-person PyHC meeting?
    - This could consist of an open session of looking at all of them, governance, etc.? Further, one member added a suggested write-in for PHEP 1: "The PHEP process will be reviewed informally yearly at the in-person PyHC meeting for that year, with more formal reviews occurring every 3-5 years as determined by the PyHC community attending those meetings."
    - This conversation was resolved with the final decision that a note will be written into PHEP-1 that PyHC will discuss at one PyHC meeting per year about the PHEP process and PHEPs in general. Intermediate discussion is saved for GitHub issues and discussions.
- What is the main advantage of having a heavy formalized process versus a discussion?
    - The main benefit is future proofing. This nicely sets up PyHC to continue to grow, sustainably. Having a process for decision making is good to define early. Also considering some bigger updates to PyHC, the community needs a new way to formalize these things. For example, at the last DASH meeting, Rebecca Ringuette led a session that teased out coordinate transformations and how to handle that for interoperability. But, PyHC can't have rules on how to approach this concept, etc., if there's no way to draft it as a community and require it of packages at certain tiers. There's not enough of a hammer at PyHC; PyHC can't have formal enforcement of something without a structure and community input and buy-in. Further proof of this point is that Nick Murphy put up standards change *four years ago*, but with no existing formal process to implement it has remained an unanswered issue on GitHub[30].
    - Original standards were imposed by people who happened to be in-person; this is kind of a scary situation. Additionally, at request, the PyHC standards have gone into NASA funding calls. This further underlines the need for PyHC processes and standards to be formalized. Another thing to consider in this conversation is not just the packages joining but what happens to a PyHC package over time. This PHEP process can help implement standards that make a package easier to maintain, make it easier to find a new maintainer, and/or make it easier for others to contribute to a package.

The initial vote for PHEP 1 passed successfully. 21 total people were present—15 of these people gave a "Yea" vote, there were no "Nay" votes, two abstained from voting, and four did not vote at all (which resulted in a default abstained vote). The second, and final, vote for PHEP 1 will take place on April 15, 2024 at a regular, bi-weekly PyHC telecon**.** A list of the individuals and their votes for PHEP 1 from vote 1 are as follows:

- Yea: 15
    - Rebecca Ringuette, Nick Murphy, Mike Shumko, Ashley Smith, Sandy Antunes, Julie Barnum, Darren De Zeeuw, Jon Vandegriff, Jim Lewis, Marcus Hughes, Shawn Polson, Nick Hatzigeorgiu, Stuart Mumford, Laura Hayes, Will Barnes

[30] https://github.com/heliophysicsPy/standards/pull/16

- Nay: 0
- Abstain: 2
    - Jonathan Niehof, Alexander Drozdov
- No Vote (default): 4
    - Angeline Burrell, Joe Plowman, Leah Wasser, Nabil Freij

PyHC - pyOpenSci Partnership - Leah Wasser

Lasser Wasser, the Executive Director and Founder of pyOpenSci[31] presented once more at a PyHC meeting on pyOpenSci, given the increasing desire to adopt a PyHC - pyOpenSci partnership. Some of the PyHC packages were on board to at least go through a pyOpenSci review (e.g., PlasmaPy and SunPy), however there were some concerns and reservations from others. Leah reiterated that the pyOpenSic peer review isn't meant to be a scary thing. It's meant to be a constructive conversation, where quality, usability, and maintainability of a package is the end goal. A JOSS publication is just a final credit for the work.

Notably, SunPy has already undergone the pyOpenSci review process (not as an affiliated package for PyHC, but as a stand-alone package). SunPy commented that the experience was very similar to the JOSS review process and in the end was productive with some interesting results. There was never a "you must do this!", but rather "hey, you could improve this part of your code." It was as light a weight of a review process as it could have been. Moreover, SunPy found that it was quite productive to have someone "from the outside" look at code base and processes and find blind spots. The experience was overall comprehensive and positive. The review can be seen by all and is located on GitHub[32].

Of note to the SunPy and astropy examples is that those communities are pretty cohesive with one main package and several affiliated packages. PyHC does not follow this same process. Thus, it is not the same in the sense of being "affiliated" with PyHC. We would need to define what it is to be affiliated with PyHC, which could be pulled in part from the astropy - pyOpenSci affiliation requirements[33]. A comment was made during discussion that a simple requirement could be "is this well documented, useful for our field, and reviewed." Additionally, another community member commented that the ability to install things in the same environment (i.e., within the PyHC environment spearheaded by Shawn Polson) is *not* the same thing as interoperability between packages. Being able to be installed side-by-side does not ensure compatibility. This must be kept in mind as PyHC determines affiliation requirements.

One result of the above noted discussion and concerns was that this was a great impetus to discuss what the overall goal of PyHC is. Is PyHC to remain a collection of packages that we advertise for people to use, or, is the community cultivating an interoperable ecosystem of packages? This decision, and the PyHC - pyOpenSci partnership, should be the highest priority

[31] https://www.pyopensci.org/
[32] https://github.com/pyOpenSci/software-submission/issues/147
[33] https://www.pyopensci.org/software-peer-review/partners/astropy.html

for the community to define following the conclusion of the PyHC summer school. Based on this presentation and ensuing discussion, it was suggested that PyHC chat with the astropy community about their experience with pyOpenSci and why they decided to go with pyOpenSci's review process.

Unconference Discussion Summary

Although there were several important and valid unconference topics suggested, based on the previous day's discussion around a PyHC - pyOpenSci partnership and the various considerations therein, the entirety of the two-hour unconference session was focused on that topic. A short summary of the session is given below. Full notes and comments can be found in the Spring 2024 meeting notes Google document[34].

pyOpenSci and PyHC: How do we define affiliation with PyHC? Carrots for going through the pyOpenSci review process?

What would affiliation with PyHC impose on packages? In reality, most pyOpenSci standards are already in-line with PyHC standard, and partnering shows more carrots than sticks. pyOpenSci wants to help users learn how to create packages, test them, apply CI/CD, and so on. If anything, pyOpenSci can assist packages and new community members in learning how to meet our standards. They will also assist PyHC in finding new maintainers of the package if the current developers abandon it, or gracefully sunset the project if it makes sense to do so.

To have a truly interoperable system of software, the standard listing of packages we do now isn't sufficient. Criteria has to be defined for what it means to be a part of PyHC. This also comes back to the need to define what is the main goal of PyHC. This begs the question, shall the community finally settle on common standards for things like coordinates, time, and units?

There lies an opportunity though to have a "marriage" of the two ideas: a listing of packages and a standards-based, interoperable PyHC. A new PyHC package tiering system would allow a repository for old, once useful and potentially still needed codes to reside within the community, while also advocating for and highlighting packages that are doing the work to reach the higher interoperability goal. The further "up the rung" a package goes on the tiers, the more carrots there are to a package (e.g., inclusion with the PyHC-Chat AI bot, funding for conference travel, or inclusion within PyHC summer school). The basic levels could be as such (with more details needed):

- Tier 0 - everything (dead code)
- Tier 1 - self-reviewed packages
- Tier 2 - meets PyHC standards

---

34 https://docs.google.com/document/d/1htbibknl1Npv1hQ1O4L22RKO_s8uc9dZT1iTJ49nMJg/edit?usp=sharing

    - If packages are no longer meeting a certain criteria, then they are bumped down a tier, informed, and given ways to go back up again.

The community agreed overall that tiering would be a good way to move forward. Once that is initiated by the community, a discussion for the PyHC - pyOpenSci can resume. This will all be handled within PHEPs. See Figs 7 and 8 for captures of a Miro board used to discuss what it is to be a PyHC package (this heavily guided discussion during the unconference session).

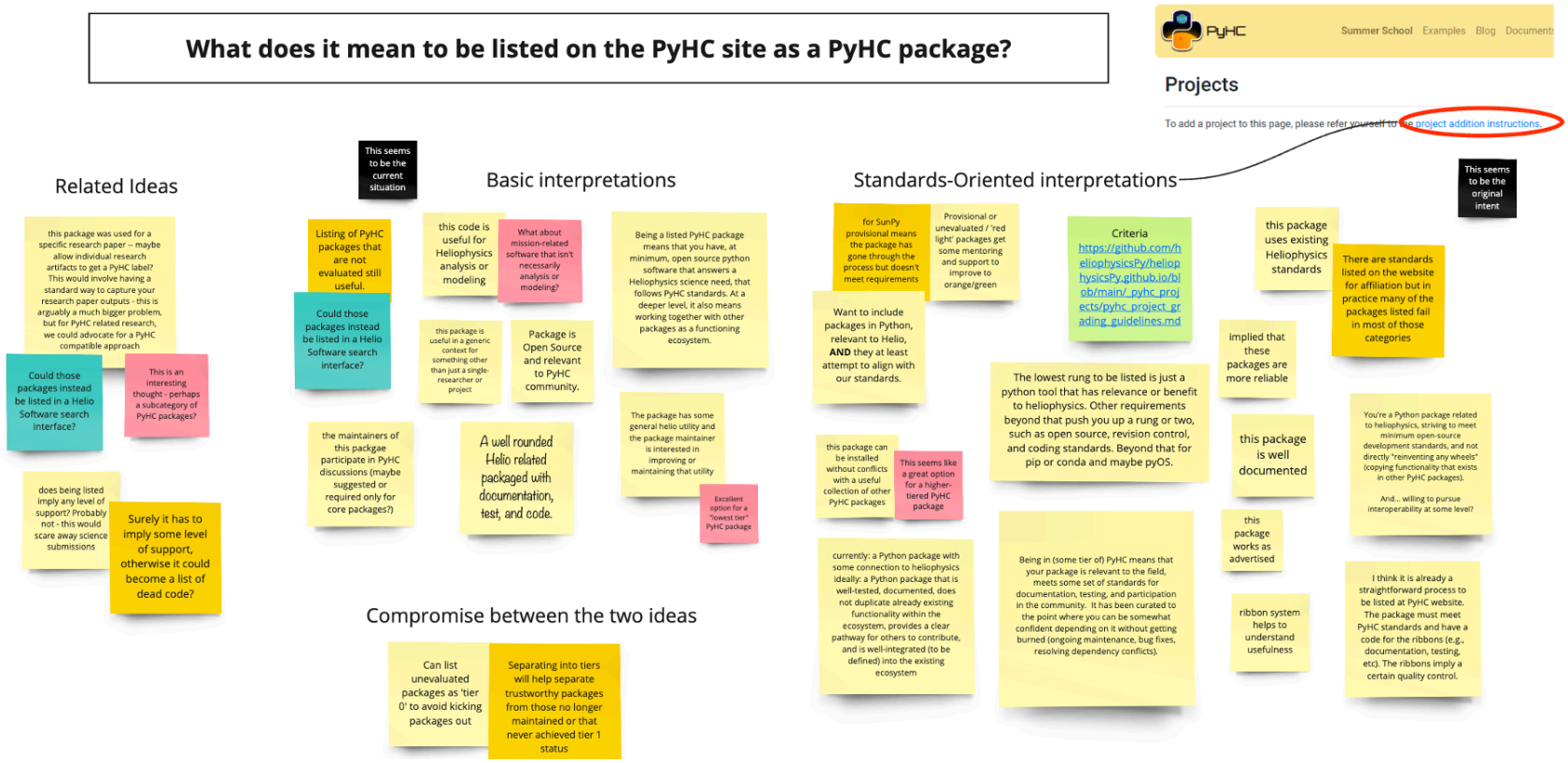
What does it mean to be listed on the PyHC site as a PyHC package?
PyHC
Summer School
Examples
Blog
Projects
To add a project to this page, please refer yourself to the project addition instructions.
Related Ideas
this package was used for a specific research paper -- maybe allow individual research artifacts to get a PyHC label? This would involve having a standard way to capture your research paper outputs - this is arguably a much bigger problem, but for PyHC related research, we could advocate for a PyHC compatible approach
Could those packages instead be listed in a Helio Software search interface?
This is an interesting thought - perhaps a subcategory of PyHC packages?
does being listed imply any level of support? Probably not - this would scare away science submissions
Surely it has to imply some level of support, otherwise it could become a list of dead code?
This seems to be the current situation
Basic interpretations
Listing of PyHC packages that are not evaluated still useful.
Could those packages instead be listed in a Helio Software search interface?
this code is useful for Heliophysics analysis or modeling
What about mission-related software that isn't necessarily analysis or modeling?
this package is useful in a generic context for something other than just a single-researcher or project
Package is Open Source and relevant to PyHC community.
Being a listed PyHC package means that you have, at minimum, open source python software that answers a Heliophysics science need, that follows PyHC standards. At a deeper level, it also means working together with other packages as a functioning ecosystem.
the maintainers of this packgae participate in PyHC discussions (maybe suggested or required only for core packages?)
A well rounded Helio related packaged with documentation, test, and code.
The package has some general helio utility and the package maintainer is interested in improving or maintaining that utility
Excellent option for a "lowest tier" PyHC package
Standards-Oriented interpretations
This seems to be the original intent
for SunPy provisional means the package has gone through the process but doesn't meet requirements
Provisional or unevaluated / 'red light' packages get some mentoring and support to improve to orange/green
Criteria https://github.com/heliophysicsPy/heliophysicsPy.github.io/blob/main/_pyhc_projects/pyhc_project_grading_guidelines.md
this package uses existing Heliophysics standards
There are standards listed on the website for affiliation but in practice many of the packages listed fail in most of those categories
Want to include packages in Python, relevant to Helio, AND they at least attempt to align with our standards.
implied that these packages are more reliable
The lowest rung to be listed is just a python tool that has relevance or benefit to heliophysics. Other requirements beyond that push you up a rung or two, such as open source, revision control, and coding standards. Beyond that for pip or conda and maybe pyOS.
this package is well documented
You're a Python package related to heliophysics, striving to meet minimum open-source development standards, and not directly "reinventing any wheels" (copying functionality that exists in other PyHC packages).
And... willing to pursue interoperability at some level?
this package can be installed without conflicts with a useful collection of other PyHC packages
This seems like a great option for a higher-tiered PyHC package
this package works as advertised
currently: a Python package with some connection to heliophysics ideally: a Python package that is well-tested, documented, does not duplicate already existing functionality within the ecosystem, provides a clear pathway for others to contribute, and is well-integrated (to be defined) into the existing ecosystem
Being in (some tier of) PyHC means that your package is relevant to the field, meets some set of standards for documentation, testing, and participation in the community. It has been curated to the point where you can be somewhat confident depending on it without getting burned (ongoing maintenance, bug fixes, resolving dependency conflicts).
ribbon system helps to understand usefulness
I think it is already a straightforward process to be listed at PyHC website. The package must meet PyHC standards and have a code for the ribbons (e.g., documentation, testing, etc). The ribbons imply a certain quality control.
Compromise between the two ideas
Can list unevaluated packages as 'tier 0' to avoid kicking packages out
Separating into tiers will help separate trustworthy packages from those no longer maintained or that never achieved tier 1 status

Figure 7

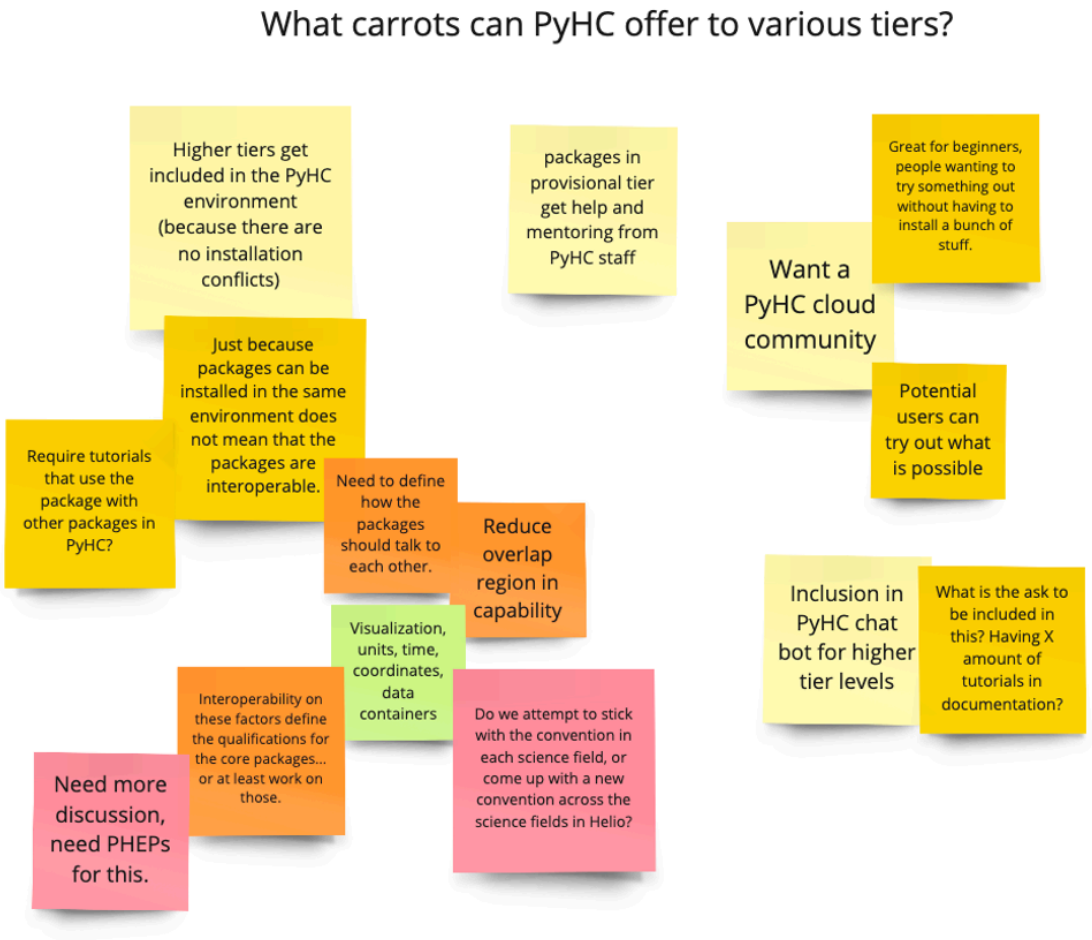

Figure 8

Next Steps

The meeting concluded with the action plans to continue conversations and development of the PyHC environment, focus on preparing for the upcoming PyHC 2024 summer school, as well as to submit a PHEP on the new PyHC package tiering system. For now, discussions regarding the PyHC - pyOpenSci partnership are tabled until the fundamental PyHC considerations are ironed out. Unlike the past year, PyHC will still have a hybrid fall 2024 meeting, considering the important topics to come (PHEPs, PyHC governance, etc.). Planning for that will begin in earnest a few months prior. Lastly, as usual, unconference topics not touched upon will be considered for future PyHC telecons:

- How to handle the contacts section for packages? Names listed at the moment, but no way of how to contact them.
- PySPEDAS, SunPY, and SpacePy seem to be working on similar tasks in field line tracing. Perhaps they can discuss how to design the efforts to be interoperable, maybe similar/identical syntaxes?
- Associating PyHC software packages with datasets (working session).

- Adding security checks to PyHC standards.
- What categories do PyHC contributors want people to use to search for their packages?
- Discussion on using all-contributors package as a PyHC recommendation.
- Discussion on creating categories of standards in PyHC (looking ahead to more complex codes that cannot be pip installed)?
- Will PyHC serve as a software search interface that includes restricted codes (e.g. person can find the code and contact info but needs access to get to it)?
- What does the community want from PyHC leadership, if Julie and Shawn are not enough for these requests then should a leadership team be created, how should others be elected onto such a PyHC leadership team?
- What happens if a project maintainer is unable/unwilling to continue? Should PyHC projects have a defined succession plan (not just github, but pypi, readthedocs, conda and other relevant assets)? Should we define a process for retiring or merging packages?

Final agenda

	Start Time (MDT)	End Time (MDT)	Duration	Title	Presenter/Discussion Lead
Mon, March 11, 2024	9:00	9:08	0:08	Welcome, Introduction, etc.	Julie Barnum
	PyHC Core Project Updates				
Mon, March 11, 2024	9:08	9:24	0:16	SunPy	Will and Stuart
Mon, March 11, 2024	9:24	9:40	0:16	PlasmaPy	Nick Murphy
Mon, March 11, 2024	9:40	9:56	0:16	HAPI	Jon Vandegriff
Mon, March 11, 2024	9:56	10:12	0:16	SpacePy	Jon Niehof
Mon, March 11, 2024	10:12	10:28	0:16	Kamodo	Darren De Zeeuw
Mon, March 11, 2024	10:28	10:44	0:16	pySPEDAS	Jim Lewis
Mon, March 11, 2024	10:44	11:00	0:16	pysat	Jeff Klenzing
	NASA HTM overviews, Science Platforms Coordination overview, and a new Heliophysics Software Search Interface				
Tue, March 12, 2024	9:00	9:30	0:30	**NASA HTM grant activities**	
				New Magnetometer Data for Space Physics Research from Seismological Networks, Incorporation into PySPEDAS	Alexander Drozdov
				Radiation Belts Analysis and Modeling Library in Python	Alexander Drozdov
				Enabling Cloud Compatibility in Heliophysics Python Software	Brent Smith
				splot: Spatiotemporal Plotting Capabilities for Heliophysics	Brent Smith
Tue, March 12, 2024	9:30	10:00	0:30	**Science Platforms Coordination: a new IHDEA wo**	
				What it is, who it is, why it is, and activities	Shawn Polson
				demo of current PyHC environment	Shawn Polson
Tue, March 12, 2024	10:00	11:00	1:00	**Heliophysics Software Search Interface**	
			0:05	Discussion motivation	Julie Barnum
			0:15	EMAC overview	Eric Lopez
			0:40	Q&A + discussion	Eric Lopez, Julie Barnum, Rebecca Ringuette, Shawn Polson
	PHEP-1 Business and pyOpenSci Parnership				
Wed, March 13, 2024	9:00	10:00	1:00	**PHEP 1: discussion, first vote**	Jon Niehof
Wed, March 13, 2024	10:00	11:00	1:00	**pyOpenSci - PyHC partnership**	
			0:10	*Implementation of a pyOpenSci - PyHC partnership*	Julie Barnum
			0:15	*pyOpenSci Review Details*	Leah Wasser
Wed, March 13, 2024			0:35	*Q&A and discussion*	Leah Wasser/Julie Barnum
	Unconferences and Wrap-up				
Thu, March 14, 2024	9:00	9:05	0:05	Unconference organizing	Julie Barnum
Thu, March 14, 2024	9:05	10:50	1:45	**Unconference session**	Various
				PyOS and PyHC: How do we define affiliation with PyHC? Carrots for going through the pyOpenSci review process?	
Thu, March 14, 2024	10:50	11:00	0:10	**Wrap-up**	Julie Barnum