The PlanetaryPy Project

Michael Aye, Ross Beyer
Andrew Annex, Chase Million

What is it?

"A **community effort** to develop a **core Python package** for **planetary science** and foster **interoperability** between **Python planetary science packages**."

- Loosely modeled after other large open-source projects (astropy, NodeJS)
- Has a governing body:
  - PlanetaryPy Technical Committee (TC)
- All of TC's work done in the open via GH repo:
  - Code of Conduct, Charter, …
  - Monthly open meetings (everybody welcome!)
  - Meeting agendas and notes
    - Updates/changes done via PR

Current status: Affiliated packages

Affiliated packages

- pvl (Trevor Olson, Ross Beyer)
  - reading /writing PDS3 labels
- SpiceyPy (Andrew Annex)
  - widely used Python SPICE wrapper
- Kalasiris (Ross Beyer)
  - Compared to PySIS:
  - Newer, more pythonic
  - Python 3 fully supported
  - Maintained
- Craterstats (GG Michael) in prep

kalasiris

Website Repository

The kalasiris library wraps functions and functionality for ISIS.

Maintainer(s): rbeyer@seti.org

Functionality	Integration	Documentation	Tests	Development	Python Version
General Package	Good	Good	Good	Good	3.6

pvl

Website Repository

Python implementation of a PVL (Parameter Value Language) library.

Maintainer(s): pvl Developers

Functionality	Integration	Documentation	Tests	Development	Python Version
General Package	Good	Good	Good	Good	3.6

SpiceyPy

Website Repository

SpiceyPy is a Python wrapper for the NAIF C SPICE Toolkit.

Maintainer(s): annex@jhu.edu

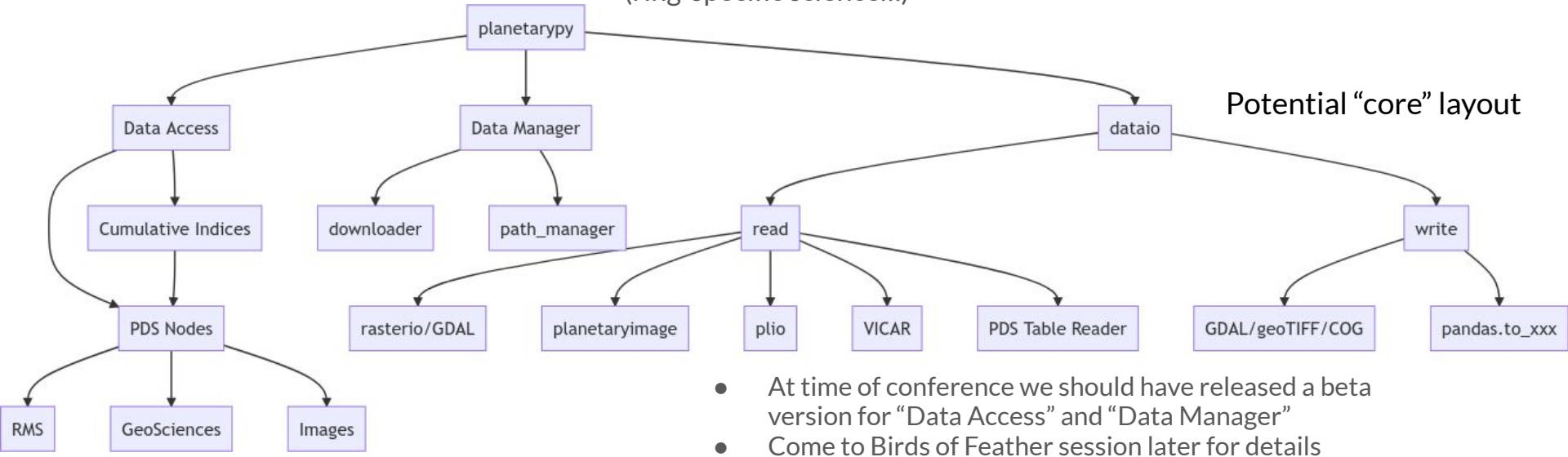
Functionality	Integration	Documentation	Tests	Development	Python Version
General Package	Good	Good	Good	Good	3.6

Current status: Core Package

- The TC members have collected several useful modules that we want to combine into “core”
- Some tools are general (see below)
- But we want to merge instrument specific modules as well
- E.g. HiRISE, Cassini ISS (ring-specific science…)

- Our TC members have applied for PDART funds
- Once for a general core pkg (declined)
- Once specifically for
  - THE planetary data reader (PDR)
  - Selected! (--> PDR poster)
  - Congrats Chase Million!!

Potential “core” layout



- At time of conference we should have released a beta version for “Data Access” and “Data Manager”
- Come to Birds of Feather session later for details

USGS
science for a changing world

The Planetary Software Organization and Open Source Software Governance

Jesse Mapel

Software Developer

USGS Astrogeology Science Center

Open Source Software Governance

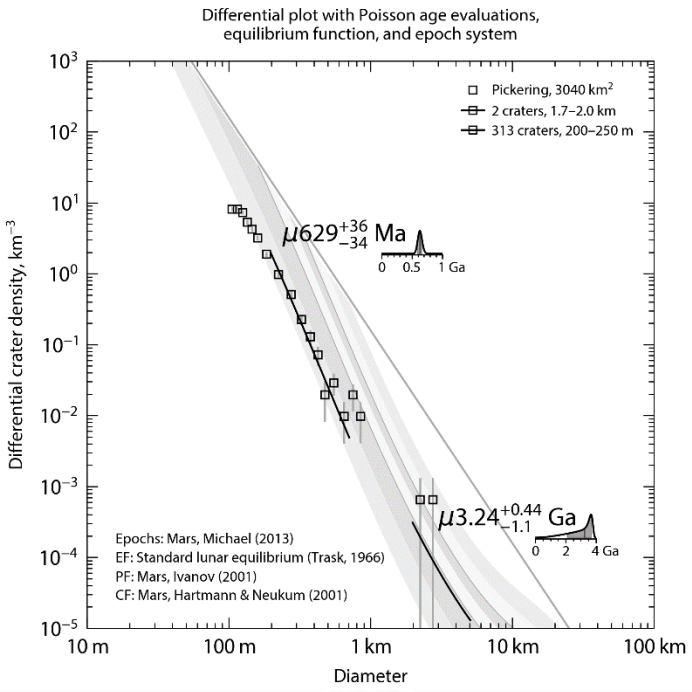
- How do you make decisions and resolve conflicts in an open source software project?
- Models
  - The Walled Garden
  - The Committee
  - The Founder-Leader
  - The Do-ocracy

USGS

Planetary Software Organization

- Examining the governance of an open source software project
  - Participation
  - Process
  - Transparency
- Technical Steering Committee (TSC)
  - Discuss governance of open source software for planetary science
  - https://github.com/planetarysoftware/TSC

USGS



Planetary surface dating with Craterstats3 – a new open source implementation in Python

Greg Michael

5th Planetary Data Workshop (PDW) and 2nd Planetary Science Informatics & Data Analytics (PSIDA) meeting
June 28–July 2, 2021

Overview

- Craterstats is a tool for estimating surface age through evaluation of a chronology model
  - Brief development history (since 2008)

- New version is written in Python instead of IDL
  - Command line interface instead of GUI
  - Some typical usage examples
  - Outline of code structure

- Impressions of Python for an IDL programmer

- Side by side: recreating an IDL plot in matplotlib

Where to find it?

https://github.com/ggmichael/craterstats

Glad to respond to any usage difficulties…
gregory.michael@fu-berlin.de

Thanks for support from Trent Hare, Corey Fortezzo, Ross Beyer, USGS

README.md

Craterstats

This is a reimplementation in Python 3.8 of the CraterstatsII software, a tool to analyse and plot crater count data for planetary surface dating.

Installation

There are various ways to install Python. If you are installing it specifically to run Craterstats, the following is suggested:

1. Install Python 3.8 or higher from python.org.
2. Download Craterstats and unzip the files where you choose.
3. Open a command prompt/terminal window and `cd` to the newly-created `craterstats-main/`
4. Enter these commands to establish a Python *virtual environment:*

Windows:

```
py -m venv venv
venv\Scripts\activate.bat
py -m pip install -r requirements.txt
```

Linux:

```
python3 -m venv venv
source venv/bin/activate
python3 -m pip install -r requirements.txt
```

After you see the required python packages installed, the set-up is complete. In the remaining sections, you should substitute `py` or `python3` for `python` according to your system.

Quick demonstration

After installation, the following commands will produce a series of example output plots and data, demonstrating the main features of the software. Plot image files are placed into the subfolder `demo/` , while text output – including the full command lines as they could be typed to generate the output – goes to the terminal window.

```
cd src
python craterstats.py -demo
```

Packages

No packages published
Publish your first package

Languages

Python 99.2% Other 0.8%

PDS API (version alpha)

Thomas Loubrieu, Jordan Padams, Dan Crichton, Emily Law, Steve Hughes
Jet Propulsion Laboratory, California Institute of Technology

5th Planetary Data Workshop &
2nd PSIDA Meeting
June 2021
Abstract #7059

PDS API Working Group

The PDS API WG provides technical expertise and guidance the PDS MC and international community on the design, development, and implementation of the PDS API Specification.

Charter (Draft):

https://docs.google.com/document/d/1I5PrNF_Jv8DQiaNvo0O0ogBCF0WMAaV9yOfGx2Y_f0M/edit#heading=h.2et92p0

PDS API Spec:

Released (v0.0.1-alpha): https://github.com/NASA-PDS/pds-api/blob/master/docs/spec/pds-api-specification.md

In Development: https://docs.google.com/document/d/16d0MVh48bFLvWsa5-B_Hy-cby1rGWdnNojWOJpUcOvA/edit?usp=sharing

API Goals and Plan

Ambition:

- Build a layer which will allow users/systems to explore/use PDS's resources seamlessly,
- In other words build the **highways to drive users to data in nodes**

Design a PDS-wide web API specification

- Initial focus is on **search on collections** (build 11.1) (in progress)
- API should be applicable to other resources (products) and services (view, download, transform, …)

Implements the standard with (Build 11.1):

- Engineering Node provides **libraries, skeleton servers, clients** to help with the implementation (done)
- Engineering Node provides a **standard implementation on top of registry** (in progress)
- Discipline Nodes (1-2) provide implementation to **demonstrate the integration** of the API (to be done)

PDS API Specification

- **PDS API Specification Core Document online:**
  - Released: https://github.com/NASA-PDS/pds-api/blob/master/docs/spec/pds-api-specification.md
  - In Development: https://docs.google.com/document/d/16d0MVh48bFLvWsa5-B_Hy-cby1rGWdnNojWOJpUcOvA/edit?usp=sharing
- **Collections/bundles/products end-points**
- **Requests:**
  - livid resolver
  - or complex search query (eq, lt, le, …, not, and, or)
- **Responses: XML PDS4 labels or JSON objects**
- **Crawl an archive:**
  - bundle → collections → products

(Swagger online specification design)

products

GET	/products	search across all PDS data products, including bundles, collections, documentation, and observational products,
GET	/products/{lidvid}	URN resolver for lidvid

collections

GET	/collections	request PDS collections
GET	/collections/{lidvid}	collections URN resolver for lidvid

collection's products

GET	/collections/{lidvid}/products	get producs belonging to a given collection

bundles

GET	/bundles	request PDS bundles
GET	/bundles/{lidvid}	bundles URN resolver for lidvid

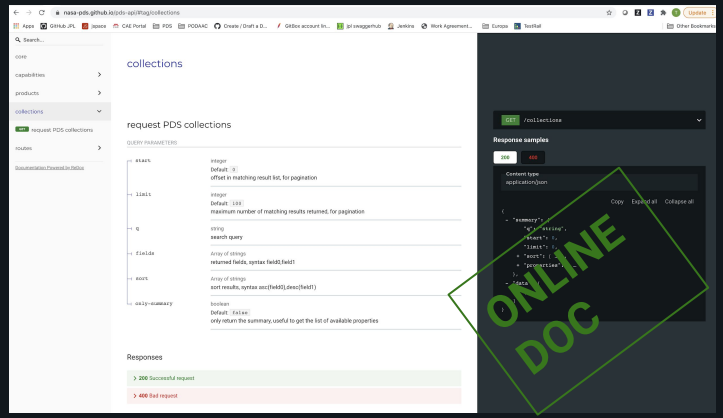
bundle's collections

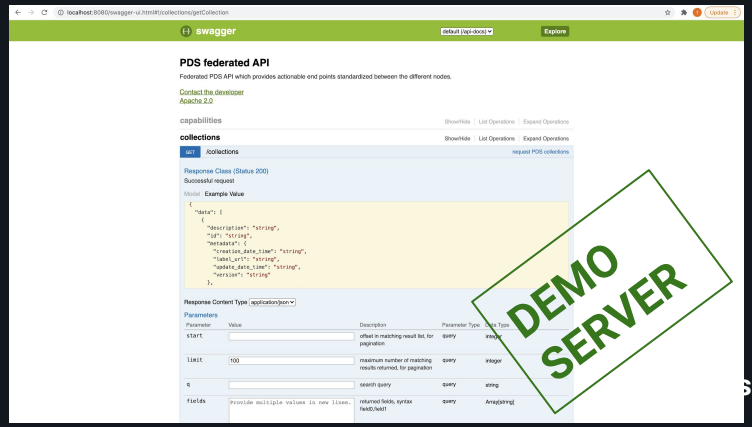
GET	/bundles/{lidvid}/collections	get collections belonging to a given bundle

API Implementation

- **An ecosystem of elementary, reusable components:**
  - Automated documentation: https://nasa-pds.github.io/pds-api/
  - Test with Postman: https://github.com/NASA-PDS/pds-api#to-generate-a-postman-test-collection
  - API search query lexer: https://github.com/NASA-PDS/api-search-query-lexer
  - API data model implementation (java): https://github.com/NASA-PDS/pds-api-javalib
  - Skeleton server (java): https://github.com/NASA-PDS/pds-api-service
  - Client (python): https://github.com/NASA-PDS/pds-api-client
  - Demonstration notebooks: https://github.com/NASA-PDS/pds-api-notebook
- **Demo server** integrated with the registry:

  https://pds-gamma.jpl.nasa.gov/api/swagger-ui.html





Demo

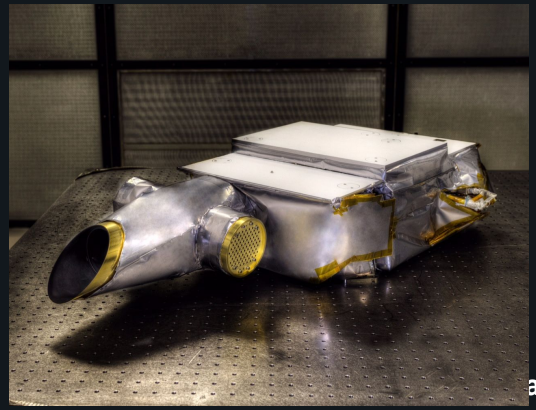
Demo Intro

Osiris-REX Visible and Infrared spectrometer (OVIR) collection:

- Part 1: explore the collection

- **Part2: find and visualize data**

https://github.com/NASA-PDS/pds-api-notebook



asa.gov

Get observations around specific spot (lat=12, lon=24) closer than 4 km

Get the result found in part1 by directly posting the search criteria to the API

The query syntax is described in the PDS API specification. It uses the following operators:

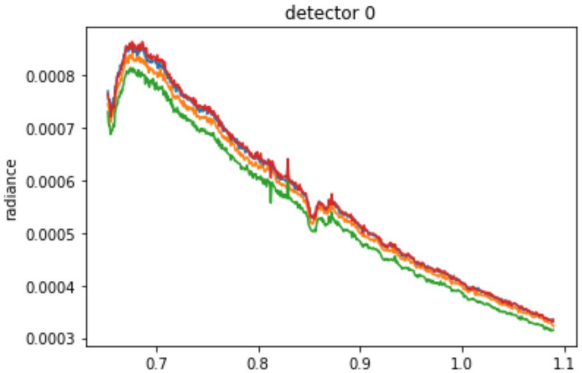
- comparison: lt, le, ...
- boolean: and, or, not
- groups: (,)

```
start_time = time.time()

products_api = pds_api.ProductsApi(api_client)

criteria = "((orex:spatial.orex:target_range lt 4.0)"
criteria += " and (orex:spatial.orex:latitude ge 9.0) and (orex:
criteria += " and (orex:spatial.orex:longitude ge 21.0) and (ore

properties_of_interest = ['orex:spatial.orex:latitude', 'orex:sp

closer_products = products_api.products(q=criteria, fields=prope

elapsed = time.time() - start_time
print(f'retrieved {len(closer_products.data)} products in {elaps
```

Plot the data (FITS files)

Plot the spectra of the 4 observations on the same figure, one figure per dimension of the instrument.

```
hduls = [fits.open(data_file) for data_file in data_files]

for i in range(20): # for each dimension of the instrument
    fig, ax = plt.subplots()
    ax.set_title(f'detector {i}')
    for hdul in hduls: # for each observation
        ax.plot(hdul[2].data[0, i, :], hdul[0].data[0, i, :])
    ax.set_xlabel('wavelength (micrometers)')
    ax.set_ylabel('radiance')
```



Questions?