Fast, reproducible, and extensible: new software tools for planetary science data access and analysis

Chase Million[1] (chase@millionconcepts.com), Michael St. Clair [1], Sierra V. Kaufman[1];
[1]Million Concepts (https://millionconcepts.com/)

Summary:
We have created a number of open source software libraries with utility in planetary science research. All of these libraries are actively being worked on by our team, and exist at a variety of levels of development, support, and stability.

`pdr` – The Planetary Data Reader is a Python-language software library that will eventually support read operations for nearly all data contained in the Planetary Data System (PDS), as well as many files in common scientific interchange data formats that are not contained in the PDS. A simple command — `read('/path/to/file')`—returns a Python object that contains that file's observational data and metadata as attributes in common scientific Python types. `pdr` is under active development and the list of formally-supported data sets—meaning that they have robust test coverage—is continuously growing. Many more data sets that are not yet formally supported also work well with `pdr`. [1][2]

`pdr-tests` – This is, primarily, the test infrastructure for the `pdr` project. It is built around a highly-configurable application called "`ix`". Because `pdr` requires a test infrastructure that can make very specific selections and samples from the over 2 petabytes of observational data contained in the PDS, `ix` treats data product selection as an integrated component of end-to-end testing; it includes a Pythonic algebra that makes these "selection rules" simple to construct and understand. Because of its generality, it is potentially just as useful for applications other than its narrowly-intended testing purpose—for instance, sorting and indexing *subsets* of planetary science data like individual instrument or mission data sets. [3][4]

`multidex` – The Multispectral Data Explorer is a locally-hosted, in-browser GUI that supports rapid exploration of spectral image data and metadata across an arbitrary number of quantitative and qualitative dimensions. It is highly extensible and instrument-agnostic, and can be easily adapted to nearly any spectral imaging system. It was initially designed for strategic scientific analysis of data from Mars Science Laboratory's Mastcam spectral imager, and is now in active use for Mars 2020's Mastcam-Z and MSL's Chemcam spectral data. [5][6][7]

`marslab` – This is a library of utilities for working with observational data, with special (but not exclusive) attention to multispectral image data from rovers. Its features include time and geometry conversions, filename and metadata parsing, and scientifically-accurate image processing functionality (e.g. debayering, masking, stretching and smoothing). It implements an multispectral data interchange format ("the `marslab` format") that is a valid input and output to `multidex`. It also contains a class named `BandSet` that supports fast, easy definition and execution of scientifically-valid, reproducible band parameter calculations; this class integrates closely with `Look`, a domain-specific language for defining spectral visualizations implemented in `Composition` (see below). [8][9]

`killscreen` – This collection of utilities provides an idiomatic Python interface to cloud resources that provides more user control than high-level interfaces like `s3fs` but is less obtuse than thin API wrappers like `boto3`. (We also think it is more Pythonic than either of these libraries.) In addition to these API interface features, it also includes rich, host-agnostic functionality for managing pipelines distributed across remote servers. A note of caution: It is trivially easy with `killscreen` to make simple commands that can terminate all of the AWS instances available to a user, spin up massive amounts of costly resources, or flood your servers with requests! Use responsibly. [10]

`dustgoggles` – This is a low-level utility library that supports every other application listed in this abstract, containing a variety of fast, handy tools for parsing text, restructuring language, and manipulating data structures. These tools include `Composition`, a flexible, Pythonic framework for domain-specific functional languages (equivalently, a pipeline assembly system), and `Codex`, a collection of in-memory caching utilities. [11]

References:

[1] https://github.com/millionconcepts/pdr
[2] Million, et al. (PDW/PSIDA 2021)
[3] Kaufman, et al. (LPSC 2022)
[4] https://github.com/millionconcepts/pdr-tests
[5] Million, et al. (LPSC 2022)
[6] https://github.com/millionconcepts/multidex
[7] Rice, M. S. (2022), Mastcam multispectral database from the Curiosity rover's traverse in Gale crater, Mars (sols 0-2302). WWU Geology Faculty Publications, 104. https://doi.org/10.25710/rqr0-8x75.
[8] https://www.youtube.com/watch?v=HWOHO2OUnzI
[9] https://github.com/millionconcepts/marslab
[10] https://github.com/millionconcepts/killscreen
[11] https://github.com/millionconcepts/dustgoogles

Acknowledgments: The development of *pdr* is supported by NASA grant No. 80NSSC21K0885. We would like to thank the Planetary Data System (PDS) for their continued cooperation. The development of *multidex*, *marslab*, and *dustoggles* have been supported in part by the Mars 2020 and Mars Science Laboratory projects.