INTRODUCTION TO PYSAT: A SPECTRAL DATA ANALYSIS TOOL FOR PLANETARY SCIENCE.
L.R. Gaddis[1], J. Laura[1], R.B. Anderson[1], T. Hare[1], R. Klima[2], F. Morgan[2], C. Viviano-Beck[2], N. Finch[3]. [1] U.S. Geological Survey, Astrogeology Science Center, Flagstaff, AZ; [2]Applied Physics Laboratory, Johns Hopkins University, Laurel, MD; [3]Northern Arizona University. (lgaddis@usgs.gov).

Introduction: The new Python Spectral Analysis Tool (PySAT) is a software library to enable visualization, thematic image derivation, and spectral analysis of planetary spectral data in a cross-platform, open-source environment. PySAT is accessible via an Application Program Interface (API) and two accompanying Graphical User Interfaces (GUIs). One GUI [1], addressed further here, is designed to support the use of hyperspectral data from NASA Moon Mineralogy Mapper (M^3) on the ISRO Chandrayaan-1 mission and the NASA Compact Reconnaissance Imaging Spectrometer for Mars (CRISM) from NASA's Mars Reconnaissance Orbiter. The second GUI is underway for the analysis of point spectrometer data with an emphasis on preprocessing and multivariate analysis of Laser-Induced Breakdown Spectroscopy (LIBS) spectra [2, 3]. The M^3 thematic product algorithms library consists of ~50 algorithms [4] and the CRISM thematic product library has ~60 algorithms [5]; there is redundancy between these algorithms, but most are keyed to different wavelengths. ***PySAT delivers complex processing algorithms for creating high-level thematic image products from archived PDS data, and frees planetary scientists from the need to use expensive or closed-source software.***

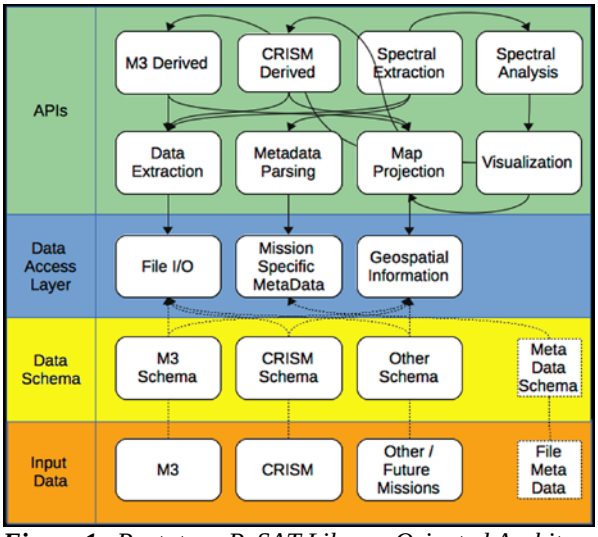

Figure 1. *Prototype PySAT Library Oriented Architecture consisting of several extensible software modules.*

Background: Spectral data ingestion, spectral analysis and product generation are supported by PySAT through the implementation of functions such as continuum correction, noise removal, band depth, position and shape analysis and derivation of related thematic products. PySAT leverages pre- and post-processing functionality of tools such as the planetary cartography tools in ISIS3 (see http://isis.astrogeology.usgs.gov/, [6]) and extends the basic band math capabilities to support more sophisticated spectral analysis and derived product generation from spectral planetary datasets. The PySAT library is not coupled to a GUI and can be readily integrated into data processing workflows.

Python: Python (https://www.python.org/) exemplifies rapid code development, an iterative workflow, and source readability. PySAT code is both Python 2.x and 3.x compliant. Use of Python provides a rapid, interactive development environment that does not require code compiling. This environment fosters a simple workflow (*e.g.*, implement, test, refactor, utilize) that facilitates efficient code prototyping and delivery. For example, code can be rapidly developed and delivered with loose requirement specifications, then iteratively refactored with input from science users as needs are refined.

Several Python libraries are used in PySAT: (1) ***NumPy*** (Numerical Python) and (2) ***SciPy*** (Scientific Python), which provide a suite of numerical analysis tools, access to the robust LaPack and Blas libraries for linear algebra operations, as well as a range of additional mathematical functions, (3) ***Pandas*** for R style dataframe representation and exceptional processing performance, (4) ***SymPy*** for symbolic notation and computation of mathematical operations, (5) ***GDAL*** (Geospatial Data Abstraction Library) which provides access to over 120 image data formats, including PDS3, ISIS2/3, FITS, ENVI, JPEG 2000, PNG and TIFF [5], (6) ***Matplotlib*** for data visualization, (7) ***PySide*** for access to the Qt Libraries, (8) ***Cython*** for automated Python to C compilation for code segments requiring the highest performance, and (9) ***scikit-learn*** for machine learning and multivariate analysis algorithms. Together these libraries form a highly effective processing suite for scientific data exploration, visualization, and analysis. These libraries, except for GDAL, PySide, and scikit-learn, are core components of the Python Scientific computing stack and widely available on all platforms. The freely available ***Anaconda Python*** software distribution mechanism simplifies installation of GDAL, PySide and scikit-learn dependencies.

Development Paths: PySAT uses a code base with high extensibility and a GUI to support visualization and analysis of multi- and hyperspectral data. Implementation of PySAT follows 6 development paths: (1)

implementation of a library of spectral analysis methods, analysis tools, derived product generation and supporting functionality (*e.g.*, file input and output), (2) creation of an automated test environment to support validation and continuous integration testing, (3) tool and derived product validation provided by M^3 and CRISM science team members, (4) user interface development and testing for a QGIS plug-in, and (5) delivery of the source code, and precompiled binaries via the USGS Astrogeology Github Page (and mirrored at NASA's Github archive), documentation via ReadTheDocs, and all deliverables linked to the PDS Imaging Node "Tools and Tutorials" site (see http://pds-imaging.jpl.nasa.gov/software/).

PySAT Library: The PySAT library is a set of classes, methods, and functions used to perform data visualization and analysis, create processing workflows, and support the creation of stand-alone applications. This library provides analytical functionality to perform basic processing tasks across multi- and hyperspectral instruments as well as specific functionality designed for the M3 and CRISM instruments. All other associated products will leverage the PySAT library. Implementation uses a Library Oriented Architecture (LOA), whereby families of similar code are defined by a single ontology and subset into a stand-alone open source package defined and managed by a unique software lifecycle plan. The M^3 and CRISM thematic product algorithms are distinct, and exist within separate ontologies (***Figure 1***). File I/O will be developed using the Geospatial Data Abstraction Library (GDAL; [7]) and provides native capabilities for common data formats noted previously. PySAT supports extraction of metadata from both the image data supplied and user-defined data (*e.g.*, from a tailored, text-based metadata file) so the addition of future datasets is straightforward.

Graphical User Interface (GUI): The graphical interface for point spectra is implemented using PyQT4 and is being developed in parallel to the interface for orbital spectral data (***Figure 2***). The prototype PySAT orbital data GUI is implemented using PySide (http://qt-project.org/wiki/PySide), a Python wrapper to the popular, cross-platform Qt library. This prototype has been significantly expanded and embedded into the popular Quantum GIS (QGIS) software package using their plug-in architecture. QGIS is an industry-standard, open-source GIS with a wide user base, planetary data and projection support, a robust development community, and works across all modern platforms (Macintosh, Windows, Linux, etc.). QGIS uses multi-threaded image rendering, allowing us to leverage high performance GUI infrastructure for large planetary data products. For example, we leverage the built-in QGIS support for Open Geospatial Consortium (OGC) existing live mapping layers (see Astrogeology WMS Map Layers here: https://astrowebmaps.wr.usgs.gov/webmapatlas/Layers/maps.html) for use as a base image or spatial context for spectral data analysis and visualization [8]. Future extensibility is supported for use of spectral modeling tools such as the Modified Gaussian Method (MGM; [9]), the functionality to integrate and compare spectra with spectral libraries such as those from the NASA/Brown University Reflectance Laboratory (RELAB, [10, 11]; see http://www.planetary.brown.edu/relab/), or any other spectral data requiring visualization and analysis.

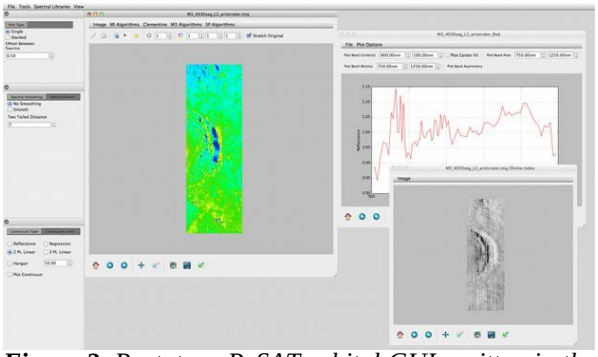

Figure 2. *Prototype PySAT orbital GUI, written in the Qt package, showing M^3 data for a lunar crater (frame M3G20090205T074030_V01_RFL.IMG). Two additional views are shown: a continuum-removed spectrum highlighting the 1- and 2-micron absorption bands and a color view of the band depth centered at 1 micron.*

Prototype: Prototype versions of both PySAT GUIs are available on GitHub: The orbital data version and the library for spectral extraction, manipulation, and visualization is online (http://github.com/USGS-Astrogeology/pysat) and the version for visualization and analysis of LIBS and other point spectral data also is available (see https://github.com/USGS-Astrogeology/PySAT_Point_Spectra_GUI). Development and implementation of orbital derived product creation is underway and will be tested and validated by this research team before release (expected in late 2018).

References: [1] Gaddis, L. et al. (2017), 48th LPSC, abs. #2548. [2] Anderson, R. et al. (2015), 2nd Planetary Data Workshop, abs. #7053. [3] Anderson, R. et al., this volume. [4] Moriarty, D. et al. (2013) *JGR-P* 118, 2310-2322. [5] Viviano-Beck, C. et al. (2014) *JGR-P* 119(6), 1403-1431. [6] Sides, S. et al. (2017) *LPS XLVIII*, 48th LPSC, abs. #2739. [7] Mitchell, T. et al. (2013), *Geospatial Power Tools*, 338 p. [8] Hare, T. et al. (2007) *LPS XXXVII*, abs. #2364. [9] Sunshine, J. et al. (1990) *JGR* 95, 6955-6966. [10] Pieters, C. and Hiroi, T. (2004) *LPS XXXV*, abs. #1720. [11] Milliken, R. et al. (2016) *LPS XLVII*, abs. #2058.