



# Advanced Topics in Python

Russell A. Stoneback

Center for Space Sciences, University of Texas at Dallas

25 June 2018

Snakes on a Spaceship: The Return of the Python

CEDAR Workshop

## SciPy 2012

- IDL to Pysat
  - Python Satellite Data Analysis Toolkit first appeared in public in April 5, 2015 (Truly in 2009)
  - High-level package designed specifically to create a common ground across all instruments in space science (think matlab/mathematica/idl but for space science)
- Pysat Stats
  - Over 1000 commits and 800 unit tests
  - Published in JGR - Space Physics <https://doi.org/10.1029/2018JA025297>
  - Pysat is running across the world (UC Berkeley, IMPEI - Brazil, NCAR/UCAR, UTD, NASA/Goddard)
  - Foundation for Ion Velocity Meter (IVM) processing for NASA ICON, NOAA COSMIC-2, NASA SORTIE CubeSat
- Features in the pipe:
  - Support for Constellations
  - pandas and xarray data format
  - Satellite simulation and IVM processing for NASA/IMPEI SPORT CubeSat

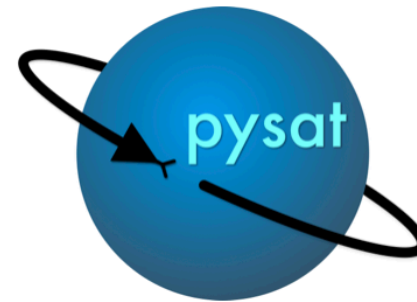
## Other Packages

- pysatCDF
  - Python interface to NASA C library, via Fortran
  - Python build system coupled to NASA build.
  - pysat not required, but includes pysat coupling features
  - Pip installable
- pysatMagVect
  - Vector system for analyzing plasma motion in the ionosphere
  - Field-Aligned, Meridional, Zonal
  - Coordinate conversions
  - Vector conversions
  - pysat not required, but includes pysat coupling features

18 June 2018

Snakes on a Spaceship: The Return of the Python - CEDAR 2018

- DaViTPy
- AACGMv2
- apexpy
- pysatCDF
- pyglow
- pyEphem
- pysgp4
- pysatMagVect
- netCDF4
- IGRF reference Fortran
- scipy
- Pandas and xarray
- OCBPY (pysat integration not public)



## pysat: Python Satellite Data Analysis Toolkit

build **passing** docs **passing** coverage **84%** DOI [10.5281/zenodo.1245182](https://doi.org/10.5281/zenodo.1245182)

- Currently using nose (command nosetests)
- New projects should use pytest
  - Built on top of python builtin unittest but don't use that directly
  - pytest can handle nosetest configurations, out of the box, according to pytest folks
  - I have not yet actually used pytest

- Its a like a unit vector, but for programs
  - Unit magnitude - test one thing
    - ▶ What is one thing anyway?
      - ~ Is going forward 10 orbits, then back 15, then forward 22, then back 17 to make sure you end up where you started one thing?
  - Known result
    - ▶ What do you mean by known?
    - ▶ What do you mean by a result?
  - Repeatable

```
class TestGeneralOrbitsMLT():
    def setup(self):
        '''Runs before every method to create a clean testing setup.'''
        info = {'index':'mlt'}
        self.testInst = pysat.Instrument('pysat','testing', '86400', clean_level='clean',
                                         orbit_info=info)

    def test_repeated_orbit_calls_symmetric_single_day_starting_with_last(self):
        self.testInst.load(2009,1)
        # start on last orbit of last day
        self.testInst.orbits[0]
        self.testInst.orbits.prev()
        control = self.testInst.copy()
        for j in range(10):
            self.testInst.orbits.next()
        for j in range(10):
            self.testInst.orbits.prev()
        assert all(control.data == self.testInst.data)
```

```
class TestGeneralOrbitsOrbitNumber(TestGeneralOrbitsMLT):

    def setup(self):
        '''Runs before every method to create a clean testing setup.'''
        info = {'index':'orbit_num', 'kind':'orbit'}
        self.testInst = pysat.Instrument('pysat','testing', '86400', clean_level='clean',
                                         orbit_info=info)

class TestGeneralOrbitsLong(TestGeneralOrbitsMLT):

    def setup(self):
        '''Runs before every method to create a clean testing setup.'''
        info = {'index':'longitude', 'kind':'longitude'}
        self.testInst = pysat.Instrument('pysat','testing', '86400', clean_level='clean',
                                         orbit_info=info)

class TestGeneralOrbitsLatitude(TestGeneralOrbitsMLT):

    def setup(self):
        '''Runs before every method to create a clean testing setup.'''
        info = {'index':'latitude', 'kind':'polar'}
        self.testInst = pysat.Instrument('pysat','testing', '86400', clean_level='clean',
                                         orbit_info=info)
```

```
class TestOrbitsGappyData2(TestGeneralOrbitsMLT):
    def setup(self):
        '''Runs before every method to create a clean testing setup.'''
        info = {'index':'mlt'}
        self.testInst = pysat.Instrument('pysat','testing', '86400',
                                         clean_level='clean',
                                         orbit_info=info)

        times = [ [pysat.datetime(2008,12,31,4), pysat.datetime(2008,12,31,5,37)],
                  [pysat.datetime(2009,1,1), pysat.datetime(2009,1,1,1,37)]
                ]

        for seconds in np.arange(38):
            day = pysat.datetime(2009,1,2)+pds.DateOffset(days=int(seconds))
            times.append([day, day+pds.DateOffset(hours=1, minutes=37, seconds=int(seconds))-pds.DateOffset(seconds=20)])

        self.testInst.custom.add(filter_data2, 'modify', times=times)

    def teardown(self):
        '''Runs after every method to clean up previous testing.'''
        del self.testInst
```



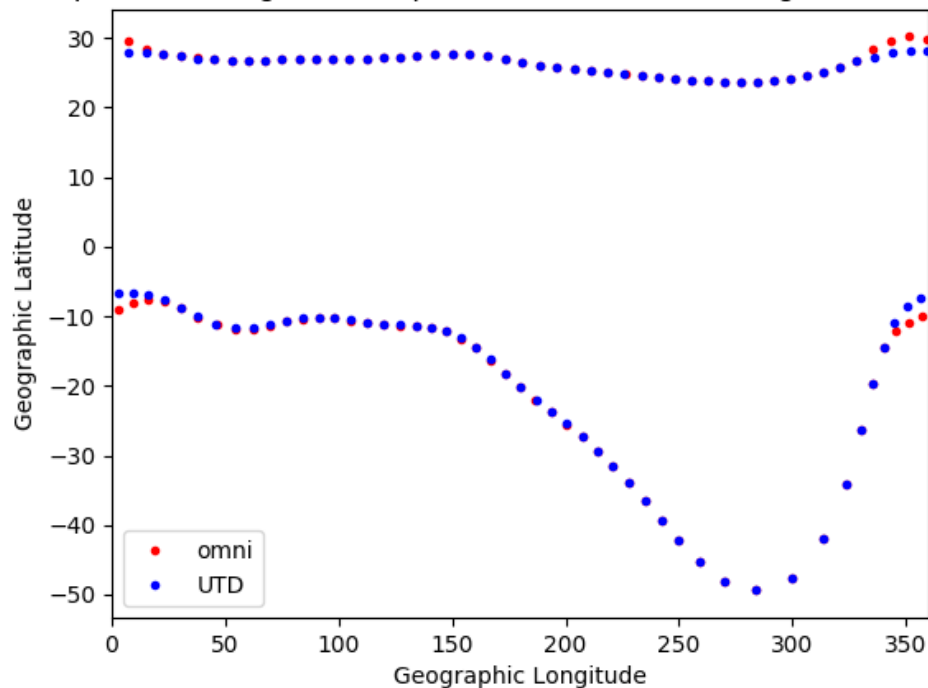
```
def test_inst_ho_data_assign_meta_different_labels(self):
    self.testInst.load(2009,1)
    frame = pds.DataFrame({'dummy_frame1':np.arange(10),
                          'dummy_frame2':np.arange(10)},
                          columns=['dummy_frame1', 'dummy_frame2'])
    meta = pysat.Meta(units_label='blah', desc_label='whoknew')
    meta['dummy_frame1'] = {'blah': 'A'}
    meta['dummy_frame2'] = {'whoknew': 'nothing'}
    self.testInst['help'] = {'data':[frame]*len(self.testInst.data.index),
                            'units': 'V',
                            'long_name': 'The Doors',
                            'meta': meta}

    assert self.testInst.meta['help', 'long_name'] == 'The Doors'
    assert 'dummy_frame1' in self.testInst.meta.ho_data['help']
    assert 'dummy_frame2' in self.testInst.meta.ho_data['help']
    assert 'dummy_frame1' in self.testInst.meta['help']['children']
    assert 'dummy_frame2' in self.testInst.meta['help']['children']
    assert self.testInst.meta['help']['children'].has_attr('units')
    assert self.testInst.meta['help']['children'].has_attr('desc')
    assert self.testInst.meta['help']['children']['dummy_frame1', 'units'] == 'A'
    assert self.testInst.meta['help']['children']['dummy_frame1', 'desc'] == ''
    assert self.testInst.meta['help']['children']['dummy_frame2', 'desc'] == 'nothing'
```

- Support for multiple metadata standards in real time
- Used to support metadata standards for IVM as part of the NASA ICON mission
- pysat functionality writes the netCDF4 files

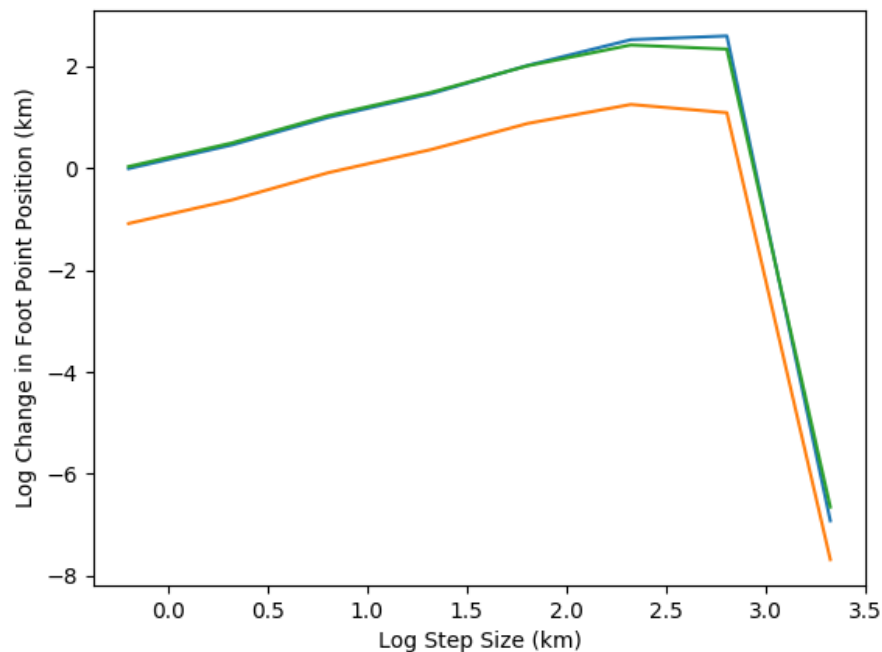
- Simple functions are easy
- Try to think like a user
  - In addition to basic tests, implement user stories
  - If user does A, then B, then C, off to D, and then E for some reason, and then G. Wait. Undo G. Do F, then do G.
- Check against public versions of models
- Check against public data
  - Lets you know when things change
- Make a plot

Comparison of Magnetic Footpoints for Field Lines through 20 Lat, 550 km



I stand by my locations though changes in comparison model settings may be appropriate

Error in Field Line Tracing Footpoint Location



Error is difference with current position and that obtained using half the step size.




# C/NOFS IVM Data

CDAWeb removed existing C/NOFS data. I found out when builds failed.

**Improved robustness when existing data sources are changed, load and ...**

...later tests not run if download fails.

🔗 master (#35) 📦 v1.0.1


 **rstoneback** committed on May 6

1 parent [7d54177](#) commit [b84999c2de0a70f01b59e5bfcd2e](#)

CDAWeb added new version of C/NOFS data. I found out when builds failed.

**Updated cleaning routine to go with new CNOFS IVM files.**

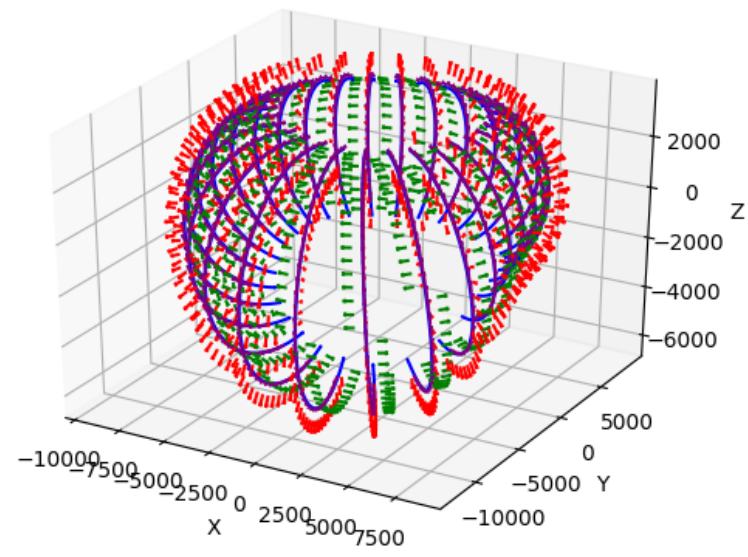
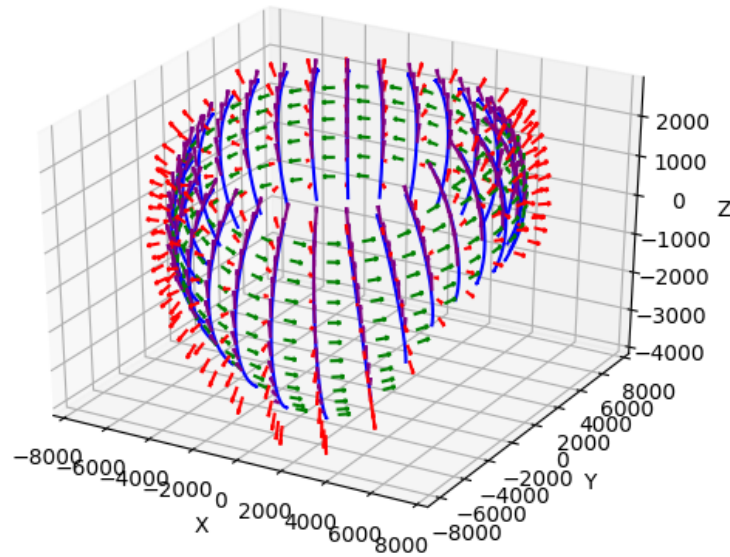
🔗 master (#36) 📦 v1.0.1

 **rstoneback** committed on May 7

1 parent [dc35e0a](#) commit [9d07416efdfd03ae30a5e6644](#)

Snakes on a Spaceship: The Return of  
the Python - CEDAR 2018

# Before and After Public pysatMagVect



I translated the code correctly. Then I made it better. Then I made it better again. Then I thought I made it better. Plots on local computer showed otherwise. Fixed now.

- Generators are functions that return control after execution but remembers state between calls
  - Yield
- I've used them with nosetests to create tests that write tests
  - Used by pysat to ensure all the different instrument modules function and adhere to the pysat standard
- While thinking of this talk, I solved a problem with random number generation
  - Random number functions work best when a ton of random numbers are created at once. Generators are great for then handing them out, except for some user interface issues with having to carry the generator around, initialize it, etc.
  - pysat + generator = good easy to obtain random numbers
  - Upcoming pysat feature (Advanced Conferencing - see Dr. Angeline Burrell)

```
def test_modules_loadable(self):
    instrument_names = pysat.instruments.__all__
    self.instruments = []

    for name in instrument_names:
        f = partial(self.check_module_importable, name)
        f.description = ' '.join(('Checking importability for module:', name))
        yield (f,)

        try:
            module = importlib.import_module(''.join(['.', name]), package='pysat.instruments')
        except ImportError:
            pass ← Not all packages work on TravisCI
        else:
            # try and grab basic information about the module so we
            # can iterate over all of the options
            f = partial(self.check_module_info, module)
            f.description = ' '.join(('Checking module has platform, name, tags, sat_ids. Testing module:', name))
            yield (f,)
```

Iterate over a dataset by file, by day. Can also include by orbit in there.

```
# do loop to iterate over instrument season
for inst in inst:
```

Iterate over a heterogeneous constellation dataset by file, or by day

```
for inst in const:
    # do loop to iterate over instrument season
    #// probably iterates by date but that all depends on the
    #// configuration of that particular instrument.
    #// either way, it iterates over the instrument, loading successive
    #// data between start and end bounds
    for inst in inst:
```

Code above from pysat seasonal bin averaging routine. Any combo of data averaged by the same code in the same bins. Transition from one to many written by undergrad CS students. 3 lines of code in the actual routine.



Supporting both pandas and xarray at once

```
@property
def index(self):
    """Returns time index of loaded data."""

    if self.pandas_format:
        return self.data.index
    else:
        if 'time' in self.data.indexes:
            return self.data.indexes['time']
        else:
            return pds.Index([])
```

How can you test all these options?

```
class TestBasicsXarray(TestBasics):
    def setup(self):
        reload(pysat.instruments.pysat_testing)
        """Runs before every method to create a clean testing setup."""
        self.testInst = pysat.Instrument('pysat', 'testing_xarray', '10',
                                         clean_level='clean',
                                         update_files=True)
```

Controlling access to data without user knowing. Respond to data assignment as well.

```
@property
def data(self):
    return self._data

@data.setter
def data(self, new_frame):
    self._data = new_frame
```

- Lack of Funding
  - NASA lacks requisite knowledge to accurately evaluate proposals
  - I was able to get techniques NASA considers too hard into pysat in one semester using a team of undergrads
  - That feature, Constellation support, is the one of the last pieces needed to enable easy system science for all instruments and all times
    - It already works, Constellations support makes it easier for operating on groups at once
  - NSF CEDAR isn't really paying either (last grant reduced by 80%)
    - This grant also specifically requires technologies I've developed
- In effect, I'm paying NASA and NSF so I can work to solve their problems
  - This is not sustainable

**Cheers**

*-Johnny Appleseed*

Snakes on a Spaceship: The Return of  
the Python - CEDAR 2018