

Code Testing and Continuous Integration



Brigitta Sipőcz

DIRAC Fellow, University of Washington



@bsipocz



@AstroBrigi



Most astronomers in fact DO test.

However they may do them as manual tests rather than in a formalized, reproducible way.

Why test - How to test

- Does the code run from start to end, do the examples work?
- Result oriented: running on known data, do I get the expected science out of the whole pipeline
- Want to ensure nothing breaks along the line

What testing is

- Code that runs features with known expected result or behavior
- Known expected result can be e.g. a mathematical relation, trivial case for an algorithm or previously buggy but now fixed behavior.

What testing is

- Code that runs features with known expected result or behavior
- Known expected result can be e.g. a mathematical relation, trivial case for an algorithm or previously buggy but now fixed behavior.

Caveats

- Writing tests takes requires significant human resources, but
- Indispensable especially for collaborative projects where developers are joining and leaving and no single person is able to oversee all the details

Unit tests

- Lightweight, code is broken up for the smallest sensible pieces
 - Modular code is critical
- Run them early and often
- Make sure the most basic parts work correctly, cover edge cases

Regression tests

- A way to collectively remember the development history of old bugs.

Other types of tests

- Benchmarking
 - Speed
 - Load
- Style
- Security
- Integration
- ...

Example

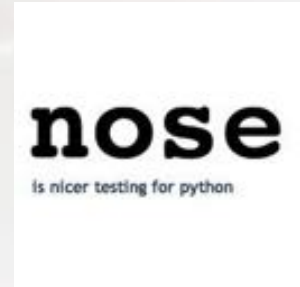
```
def test_constellations():  
  
    sc = SkyCoord(135*u.deg, 65*u.deg)  
    assert sc.get_constellation() == 'Ursa Major'  
    assert sc.get_constellation(short_name=True) == 'UMa'  
  
    scs = SkyCoord([135]*2*u.deg, [65]*2*u.deg)  
    npt.assert_equal(scs.get_constellation(), ['Ursa Major']*2)  
    npt.assert_equal(scs.get_constellation(short_name=True), ['UMa']*2)
```


Code coverage - The illusion of 100% coverage

- 100% coverage:
 - All code was run
 - it doesn't mean all code was tested, all cases may not been covered
 - **But:** Code not covered was certainly not tested
- Passing a test doesn't mean the code is correct
- Incorrect code can pass a tests suite, and correct code can fail
- Tests can be incomplete, tests can be wrong

In practice

- Pytest
Nose
unittest
mock



- Test driven development:
 - write the tests based on the API design
 - test results of algorithms based on extra knowledge
 - from analytic solutions of simple cases
 - test data design (test data are drawn from known distribution)

Continuous integration

- Running tests automatically for all proposed code changes.
- Goal: Catch problems, incompatibility with the existing codebase introduced by new modifications as soon as possible.

Continuous Integration

- Self hosted, e.g.



- Cloud hosted providers, integrated with GitHub:

- Travis CI



- Appveyor



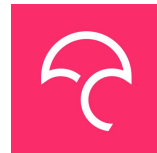
- Azure Pipeline



- CircleCI



- Codecov



Continuous Integration

astropy / astropy  

Current Branches Build History Pull Requests > Build #21243

More options 

✓ Pull Request #8077: Fix a bug that caused writing to FITS of Table objects that had index columns

#21243 passed

 Restart build

→ Commit FF35F63 

#8077: Fix a bug that caused writing to FITS of Table objects that had index columns to fail 

→ Branch master 

Thomas Robitaille





⌚ Ran for 23 min 58 sec

⌚ Total time 58 min 25 sec

📅 a day ago





Initial tests

⌚ 9 min 25 sec

✓	# 21243.1		Compiler: clang C	PYTHON_VERSION=3.7 CONDA_DEPENDENCIES=\$CONDA_ALL_DEPENDENCIES	⌚ 9 min 15 sec	
✓	# 21243.2		Compiler: gcc C	MAIN_CMD="flake8 astropy --count SFLAKE8_OPT" SETUP_CMD=""	⌚ 2 min 59 sec	

Comprehensive tests

⌚ 14 min 41 sec

✓	# 21243.3		Compiler: gcc C	PYTHON_VERSION=3.5 NUMPY_VERSION=1.13 INSTALL_CMD="python setup.py bi	⌚ 7 min 16 sec	
✓	# 21243.4		Compiler: gcc C	PYTHON_VERSION=3.5 SETUP_CMD="test --remote-data=astropy --readonly" CD	⌚ 10 min	
✓	# 21243.5		Compiler: gcc C	SETUP_CMD="test --coverage --remote-data=astropy --readonly" CONDA_DEPEN	⌚ 14 min 41 sec	
✓	# 21243.6		Compiler: gcc C	NUMPY_VERSION=prerelease EVENT_TYPE='push/pull_request cron'	⌚ 2 min 10 sec	
✓	# 21243.7		Compiler: gcc C	NUMPY_VERSION=dev SETUP_CMD="test --remote-data" CONDA_DEPENDENCIES	⌚ 12 min 4 sec	

Continuous Integration

ci-helpers

[Current build](#) [History](#) [Deployments](#) [Events](#) [Settings](#)



NEW BUILD



RE-BUILD PR

Pull request #333 - Updating astropy versions to 2.0.9 and 3.0.5

Pinning PYTHON_VERSION to make sure conda doesn't update it

1.0.1056

10 days ago by Brigitte Blocc

`17 master -- 7b453b26 -- 17 astropy_305 -- 280910fd`

Failed 10 days ago in 35 min 34 sec

Join

Job name	Tests	Duration
Environment: PYTHON_VERSION=2.7, NUMPY_VERSION=development, CONDA_DEPENDENCIES=scipy gymux, CONDA...		9 min 54 sec
Environment: PYTHON_VERSION=3.6, NUMPY_VERSION=stable, ASTROPY_VERSION=stable, CONDA_DEPENDENCE...		3 min 40 sec
Environment: PYTHON_VERSION=2.7, DEBUG=True		1 min 17 sec
Environment: PYTHON_VERSION=3.4, NUMPY_VERSION=1.9		1 min 44 sec
Environment: PYTHON_VERSION=3.5, NUMPY_VERSION=1.13, ASTROPY_VERSION=development		3 min 41 sec
Environment: PYTHON_VERSION=3.5, NUMPY_VERSION=1.13, ASTROPY_VERSION=its		2 min 14 sec
Environment: PYTHON_VERSION=3.5, CONDA_ENVIRONMENT=conda_environment.yml, TEST_CMD=python -c "import...		Cancelled
Environment: PYTHON_VERSION=3.6, ASTROPY_VERSION=3.0, SUNPY_VERSION=0.9, PIP_FALLBACK=False, CONDA...		Cancelled