Software Testing

Adrian Price-Whelan

(Columbia University)

adrianprw adrn

github.com/adrn/SoftwareTesting

github.com/adrn/SoftwareTesting

Disclaimer:

This is **Python**-focused

(but applies to any language)

Why

Why

- Complex code is impossible to debug by eye

Why

- Complex code is impossible to debug by eye
- Testing is a robust way to validate software

Why

- Complex code is impossible to debug by eye

- Testing is a robust way to validate software
  - Does it work as the I expect?

Why

- Complex code is impossible to debug by eye

- Testing is a robust way to validate software
  - Does it work as the I expect?
  - Does it work as the users expect?

Why

- Complex code is impossible to debug by eye

- Testing is a robust way to validate software
  - Does it work as the I expect?
  - Does it work as the users expect?
  - Does this update break existing code?

Why

- Complex code is impossible to debug by eye

- Testing is a robust way to validate software
  - Does it work as the I expect?
  - Does it work as the users expect?
  - Does this update break existing code?

- Testing also helps refine structure / intent

Why

- Complex code is impossible to debug by eye

- Testing is a robust way to validate software
  - Does it work as the I expect?
  - Does it work as the users expect?
  - Does this update break existing code?

- Testing also helps refine structure / intent
  - Is this code modular and testable?

What

What

- Unit tests
  - Check small “units” of code independent of other code
  - Is the code doing what the programmer expects?

What

- Unit tests
  - Check small “units” of code independent of other code
  - Is the code doing what the programmer expects?
- Functional tests
  - Check that units of code interact as expected
  - Tests may use multiple classes, functions, modules…
  - Is the programmer doing what the user expects?

Hypothetical software

Hypothetical software

- Read an optical spectrum from a text file

Hypothetical software

- Read an optical spectrum from a text file

- Integrate flux over some wavelength range

Unit tests

- Read an optical spectrum from a text file
  - Does it succeed for all anticipated input types?
  - Does it fail if you have invalid data?
- Integrate flux over some wavelength range
  - Is integral positive for physical input values?

Functional tests

Functional tests

- Read an optical spectrum from a text file
- Integrate flux over some wavelength range

  - Generate valid temp. file on the fly with known integral
  - Read from temporary file, pass to integration routine
  - Check output

Functional tests

- Read an optical spectrum from a text file
- Integrate flux over some wavelength range

  - Generate valid temp. file on the fly with known integral
  - Read from temporary file, pass to integration routine
  - Check output

➡ Tests end-to-end functionality

Test-driven development

Test-driven development

1. Write tests / API

Test-driven development

1. Write tests / API

2. Write skeleton of code

Test-driven development

1. Write tests / API
2. Write skeleton of code
3. Run tests and debug

Test-driven development

1. Write tests / API
2. Write skeleton of code
3. Run tests and debug
4. Write code until tests pass

Test-driven development

1. Write tests / API
2. Write skeleton of code
3. Run tests and debug
4. Write code until tests pass

Tools: testing frameworks

Tools: testing frameworks

- `unittest`
  - Standard library
  - Requires some boiler-plate code to make simple tests

Tools: testing frameworks

- `unittest`
  - Standard library
  - Requires some boiler-plate code to make simple tests

- `py.test`
  - Third-party package
  - Easy to write simple tests
  - Plugin architecture (e.g., doctests, coverage, pep8, mpl, ...)
  - *(used by Astropy)*

Tools: testing frameworks

- `unittest`
  - Standard library
  - Requires some boiler-plate code to make simple tests

- `py.test`
  - Third-party package
  - Easy to write simple tests
  - Plugin architecture (e.g., doctests, coverage, pep8, mpl, ...)
  - *(used by Astropy)*

- `nose`
  - Similar to py.test — pick one and roll with it

Tools: testing frameworks

- Discovers & runs functions starting with `test_…`
  classes starting with `Test…`

```
% py.test demo/tests.py
========================== test session starts ==========================
platform darwin -- Python 3.5.1, pytest-2.8.1, py-1.4.30, pluggy-
0.3.1
rootdir: /Users/adrian/projects/softwaretesting/demo, inifile:
collected 3 items

demo/tests.py ...

======================= 3 passed in 2.29 seconds ========================
```

Tools: testing frameworks

```
% py.test demo/tests.py
========================== test session starts ==========================
platform darwin -- Python 3.5.1, pytest-2.8.1, py-1.4.30, pluggy-
0.3.1
rootdir: /Users/adrian/projects/softwaretesting/demo, inifile:
collected 3 items

demo/tests.py .F.

=============================== FAILURES ================================
_____________________ test_creation_from_file _____________________
```

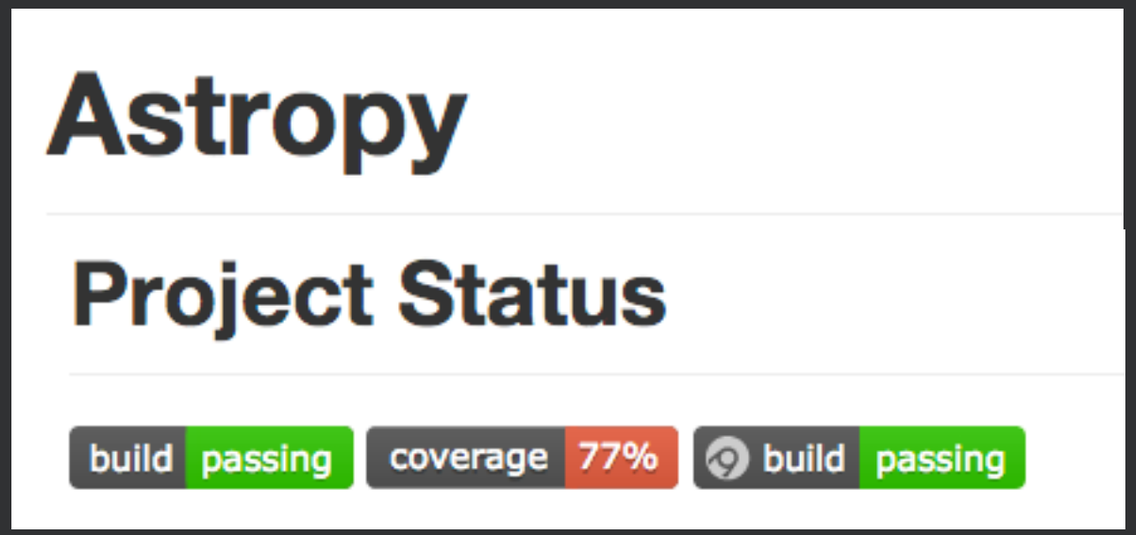
Tools: testing frameworks

```
                else:
                    fmt = [fmt, ] * ncol
                format = delimiter.join(fmt)
            elif iscomplex_X and n_fmt_chars != (2 * ncol):
                raise error
            elif ((not iscomplex_X) and n_fmt_chars != ncol):
>               raise error
E               ValueError: fmt has wrong number of % formats
:  %s %s

../../anaconda/envs/three/lib/python3.5/site-packages/numpy/lib/n
pyio.py:1137: ValueError
=============== 1 failed, 2 passed in 2.03 seconds ===============
```

Tools: continuous integration

- When new code is integrated or is proposed (e.g., pull request), verify that tests pass
    - See: Travis-CI, Jenkins



Tools: continuous integration

- Travis-CI — “matrix” builds

Build Jobs

✓ # 11169.1		no language set	PYTHON_VERSION=2.6 SETUP_CMD='egg_info'	48 sec
✓ # 11169.2		no language set	PYTHON_VERSION=2.7 SETUP_CMD='egg_info'	39 sec
✓ # 11169.3		no language set	PYTHON_VERSION=3.3 SETUP_CMD='egg_info'	1 min 5 sec
✓ # 11169.4		no language set	PYTHON_VERSION=3.4 SETUP_CMD='egg_info'	55 sec
✓ # 11169.5		no language set	PYTHON_VERSION=3.5 SETUP_CMD='egg_info'	1 min 5 sec
✓ # 11169.6		no language set	PYTHON_VERSION=2.7 SETUP_CMD='test' CONDA_DEPENDENCIES=$(	11 min 16 sec
✓ # 11169.7		no language set	PYTHON_VERSION=2.7 SETUP_CMD='build_sphinx -w' CONDA_DEPEN	8 min 57 sec
✓ # 11169.8		no language set	PYTHON_VERSION=2.6 SETUP_CMD='test --open-files'	8 min 38 sec
✓ # 11169.9		no language set	PYTHON_VERSION=2.7 SETUP_CMD='test --open-files'	12 min 15 sec
✓ # 11169.10		no language set	PYTHON_VERSION=3.3 SETUP_CMD='test --open-files'	7 min 8 sec
✓ # 11169.11		no language set	PYTHON_VERSION=3.4 SETUP_CMD='test --open-files'	7 min 10 sec
✓ # 11169.12		no language set	PYTHON_VERSION=3.5 SETUP_CMD='test --open-files'	6 min 15 sec
✓ # 11169.13		no language set	PYTHON_VERSION=2.7 SETUP_CMD='test --coverage' CONDA_DEPEN	12 min 25 sec
✓ # 11169.14		no language set	PYTHON_VERSION=3.4 SETUP_CMD='test' CONDA_DEPENDENCIES=$(	8 min 21 sec
✓ # 11169.15		no language set	PYTHON_VERSION=2.7 NUMPY_VERSION=1.8 SETUP_CMD='test'	7 min 16 sec
✓ # 11169.16		no language set	PYTHON_VERSION=2.7 NUMPY_VERSION=1.7 SETUP_CMD='test'	6 min 58 sec
✓ # 11169.17		no language set	PYTHON_VERSION=2.7 NUMPY_VERSION=1.6 SETUP_CMD='test'	8 min 43 sec
✓ # 11169.19		no language set	PYTHON_VERSION=2.7 MAIN_CMD='pep8 astropy --count' SETUP_CM	53 sec

Tools: continuous integration

- Travis-CI — “matrix” builds

Build Jobs

✓ # 11169.1		no language set	PYTHON_VERSION=2.6 SETUP_CMD='egg_info'	48 sec
✓ # 11169.2		no language set	PYTHON_VERSION=2.7 SETUP_CMD='egg_info'	39 sec
✓ # 11169.3		no language set	PYTHON_VERSION=3.3 SETUP_CMD='egg_info'	1 min 5 sec
✓ # 11169.4		no language set	PYTHON_VERSION=3.4 SETUP_CMD='egg_info'	55 sec
✓ # 11169.5		no language set	PYTHON_VERSION=3.5 SETUP_CMD='egg_info'	1 min 5 sec
✓ # 11169.6		no language set	PYTHON_VERSION=2.7 SETUP_CMD='test' CONDA_DEPENDENCIES=$(	11 min 16 sec
✓ # 11169.7		no language set	PYTHON_VERSION=2.7 SETUP_CMD='build_sphinx -w' CONDA_DEPEN	8 min 57 sec
✓ # 11169.8		no language set	PYTHON_VERSION=2.6 SETUP_CMD='test --open-files'	8 min 38 sec
✓ # 11169.9		no language set	PYTHON_VERSION=2.7 SETUP_CMD='test --open-files'	12 min 15 sec
✓ # 11169.10		no language set	PYTHON_VERSION=3.3 SETUP_CMD='test --open-files'	7 min 8 sec
✓ # 11169.11		no language set	PYTHON_VERSION=3.4 SETUP_CMD='test --open-files'	7 min 10 sec
✓ # 11169.12		no language set	PYTHON_VERSION=3.5 SETUP_CMD='test --open-files'	6 min 15 sec
✓ # 11169.13		no language set	PYTHON_VERSION=2.7 SETUP_CMD='test --coverage' CONDA_DEPEN	12 min 25 sec
✓ # 11169.14		no language set	PYTHON_VERSION=3.4 SETUP_CMD='test' CONDA_DEPENDENCIES=$(	8 min 21 sec
✓ # 11169.15		no language set	PYTHON_VERSION=2.7 NUMPY_VERSION=1.8 SETUP_CMD='test'	7 min 16 sec
✓ # 11169.16		no language set	PYTHON_VERSION=2.7 NUMPY_VERSION=1.7 SETUP_CMD='test'	6 min 58 sec
✓ # 11169.17		no language set	PYTHON_VERSION=2.7 NUMPY_VERSION=1.6 SETUP_CMD='test'	8 min 43 sec
✓ # 11169.19		no language set	PYTHON_VERSION=2.7 MAIN_CMD='pep8 astropy --count' SETUP_CM	53 sec

Tools: continuous integration

Make jsviewer table have a sortable index #4404

Open **eteq** wants to merge 12 commits into `astropy:master` from `eteq:virtual-index-col`

Conversation 20 | Commits 12 | Files changed 4

eteq commented 19 days ago — Owner

comments, commits, etc...

All checks have passed
3 successful checks — Show all checks

This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.

Conclusions

Conclusions

- Unit tests by themselves are almost useless — balance unit and functional tests

Conclusions

- Unit tests by themselves are almost useless — balance unit and functional tests

- Try out test-driven development — could save you time in the long run

Conclusions

- Unit tests by themselves are almost useless — balance unit and functional tests

- Try out test-driven development — could save you time in the long run

- There are many tools out there that make writing and running tests easy — use them!

Conclusions

- Unit tests by themselves are almost useless — **balance unit and functional tests**

- Try out **test-driven development** — could save you time in the long run

- There are many tools out there that make writing and running tests easy — use them!
  (try: **Py.test + GitHub + Travis-CI**)