Continuous Test / Integration

Michael Hirsch

CEDAR Workshop 2019

Why use an automated testing system?

- Saves much needless errors found by colleagues and users
- Ensures code lint standards are met (PEP8, code style, type hinting)
- Check that all or critical versions of compilers, interpreters, OS are supported with each “git push”

CI use is improving in the heliophysics community

- Need to update/transition away from cumbersome, outdated test systems missing critical functionality, or have excessively verbose and difficult to maintain syntax
- Encourage being flexible to use of multiple CI systems to improve coverage, decrease reliance on single system that could disappear
- Software intended for use on end-user computers should always **test on Windows as well as Linux**

Currently, two free CI providers have easy access to Linux, MacOS and Windows:

AppVeyor

Travis CI

Any language is supported by CI

- If you can build it on your computer, normally it can be done on CI
- Obvious exceptions are running large simulations—make a small test case, test the components of your model "unit test"
- Projects that need a lot of setup may be better served by using a Docker image on the CI (faster to load and run)
- "on-premise" CI requires additional setup and maintenence, but is available for free from AppVeyor and traditional systems like Jenkins

CI is essential when doing significant changes

- First write registration cases that test the whole project
- Then write unit tests, at least for the code being added / changed
- Implement the CI
- Iterate

Note: Python 2.x => 3.x upgrades should additionally add type hinting and CI checks of type hinting.

- Industry continues to make significant investment in type hinting. Any Python project will strongly benefit from use of type hinting.

Selecting test framework

- Don’t invent your own!
- Python: PyTest, C++: Google Test

For Python, PyTest is essential—it is so much simpler to achieve much better test coverage

conftest.py

```
import pytest
import random

@pytest.fixture
def floatgen():
      a = random.random()
      b = random.random()
      return a, b
```

test_adding.py

```
import mathfun as fun
from pytest import approx

def test_addints():
      assert fun.add(1, 1) == 2

def test_addfloats(floatgen):
      x, y = floatgen
      assert fun.add(x, y) == approx(x+y)
```

`pytest -v` reveals that Pytest knows *a priori* to find fixtures in conftest.py

Travis-CI examples

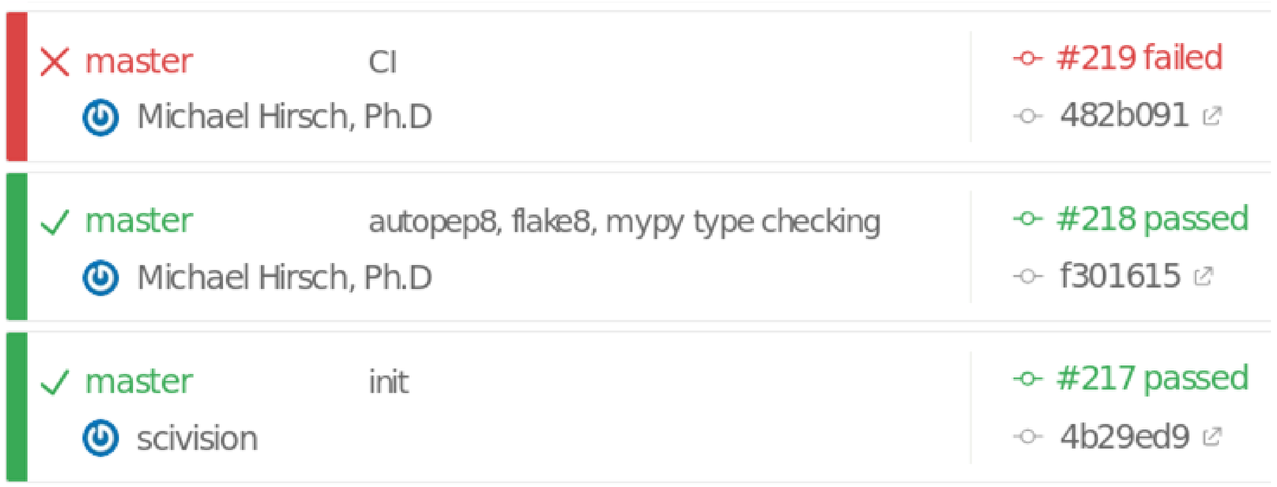
Python: .travis.yml

```yaml
language: python

python:
- 3.7
- 2.7

install: pip install -e .[tests]

script:
- pytest -v
- mypy .
- flake8
```

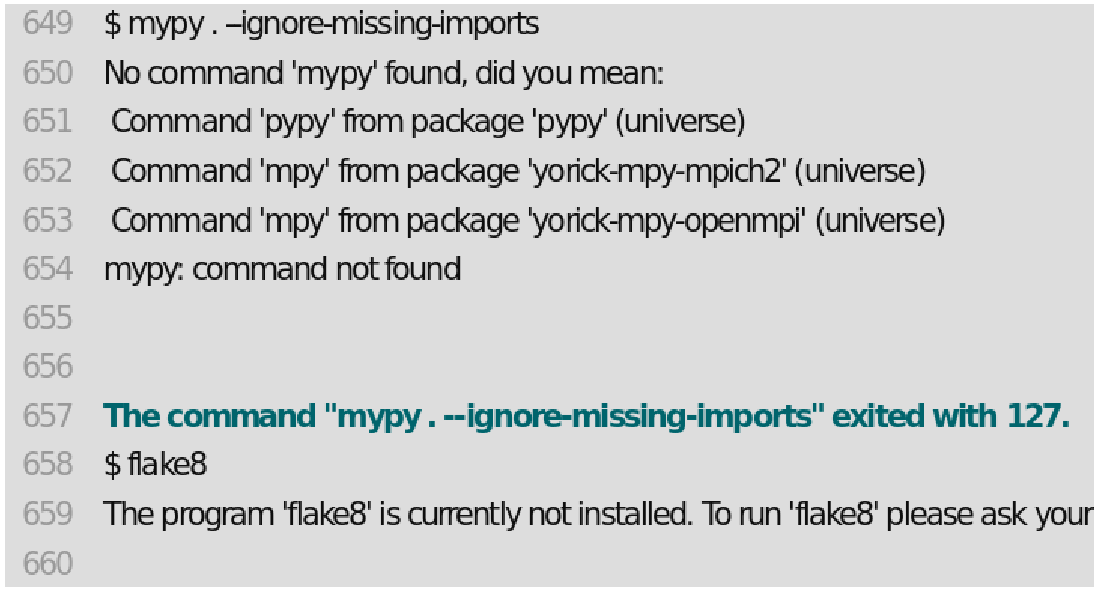
C++: .travis.yml

```yaml
language: cpp

install:
- cmake –B build
- cmake --build build -j

script:
- cd build
- ctest -V
```

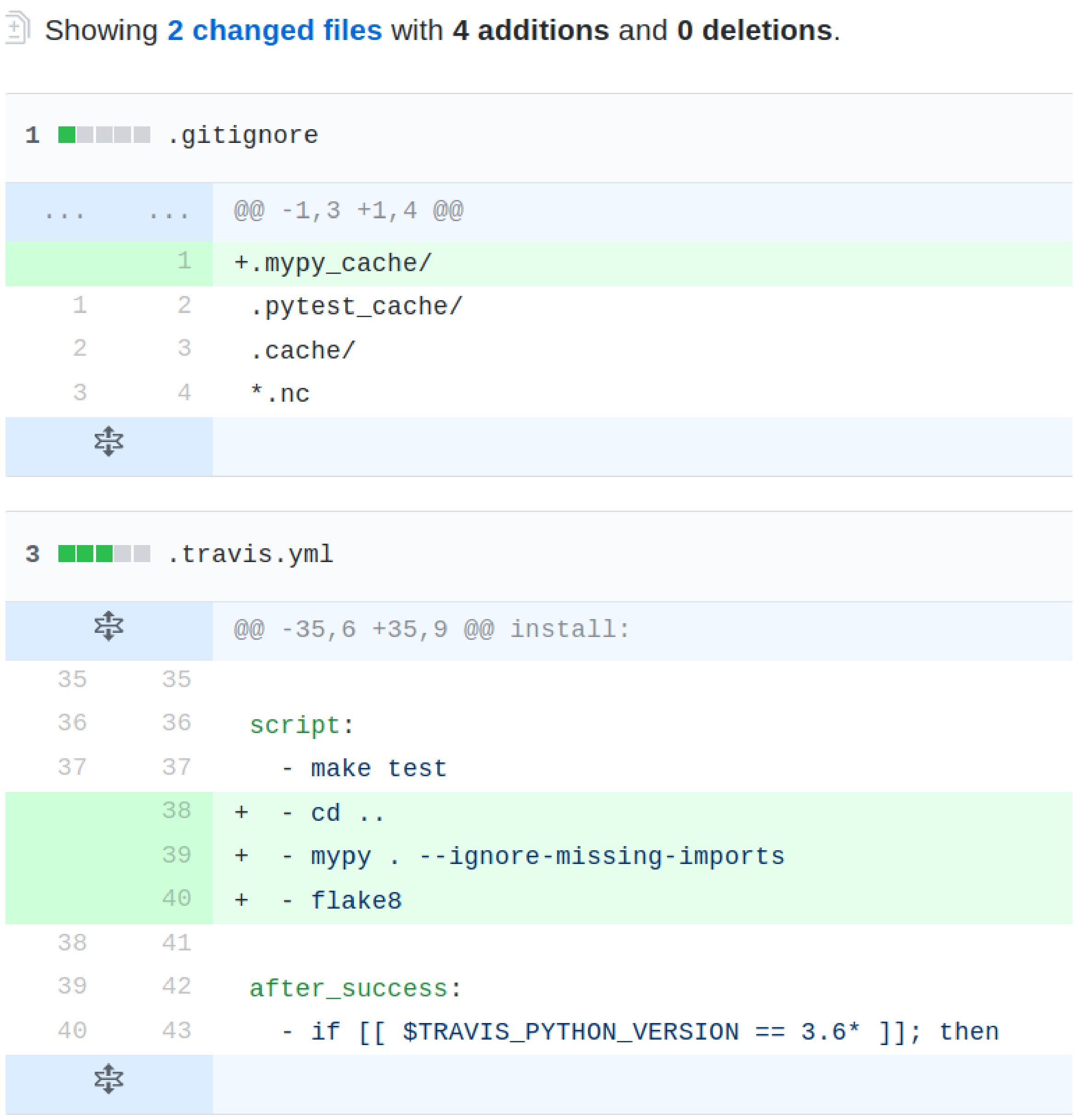
Travis-CI status dashboard



```
✗ master   CI                                        #219 failed
  Michael Hirsch, Ph.D                               482b091

✓ master   autopep8, flake8, mypy type checking      #218 passed
  Michael Hirsch, Ph.D                               f301615

✓ master   init                                      #217 passed
  scivision                                          4b29ed9
```



```
$ mypy . –ignore-missing-imports
No command 'mypy' found, did you mean:
 Command 'pypy' from package 'pypy' (universe)
 Command 'mpy' from package 'yorick-mpy-mpich2' (universe)
 Command 'mpy' from package 'yorick-mpy-openmpi' (universe)
mypy: command not found


The command "mypy . --ignore-missing-imports" exited with 127.
$ flake8
The program 'flake8' is currently not installed. To run 'flake8' please ask your

```



Showing **2 changed files** with **4 additions** and **0 deletions**.

```
1 .gitignore
@@ -1,3 +1,4 @@
+.mypy_cache/
 .pytest_cache/
 .cache/
 *.nc
```

```
3 .travis.yml
@@ -35,6 +35,9 @@ install:

 script:
   - make test
+  - cd ..
+  - mypy . --ignore-missing-imports
+  - flake8

 after_success:
   - if [[ $TRAVIS_PYTHON_VERSION == 3.6* ]]; then
```