Mid-Term Presentation for the Course Collabrative Software Development

Xianghang Zhang

Team C: Tong Liu, Hakan Yeniavci, Xianghang Zhang

2021

Outline

- Test Driven Development
- Continuous Integration Deployment
- An Example

Test Driven Development

What functions should our program implement?

- Basic: General, multiple random test data
- More: Some specific trajectory
- Exception handling

Basic Tests

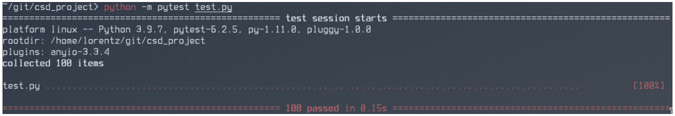
- Run multiple times
- Random data: using fixtures from pytest

```python
@pytest.fixture
def trajectoryData():
    traj = gen.Generator(2.5e-7)
    traj.generate()
    return traj

@pytest.fixture
def resultData(trajectoryData):
    x_n_max, x_n_min, y_n_max, y_n_min = trajectoryData.observe()
    cal = f.fit(x_n_max, x_n_min, y_n_max, y_n_min)
    cal.compute_observed()
    cal.linear_regression()
    return cal

@pytest.mark.parametrize('execution_number', range(100))
def test_fitting(trajectoryData, resultData, execution_number):
    assert m.isOk(trajectoryData, resultData, 1e-1)
```

Basic Tests



```
~/git/csd_project> python -m pytest test.py
==================================================== test session starts ====================================================
platform linux -- Python 3.9.7, pytest-6.2.5, py-1.11.0, pluggy-1.0.0
rootdir: /home/lorentz/git/csd_project
plugins: anyio-3.3.4
collected 100 items

test.py ....................................................................................................           [100%]

==================================================== 100 passed in 0.15s ====================================================
```

Some More...

- Lines with 0 slope

```
@pytest.fixture
def traj_zero_slope():
    traj = gen.Generator(2.5e-7)
    traj.kx = 0
    traj.ky = 0
```

- Lines specific to some bugs

```
@pytest.fixture
def traj_err_slope():
    traj = gen.Generator(2.5e-7)
    traj.kx = 0.1415785793306974
    traj.ky = 0.4965192218296801
```

Continuous Integration Deployment

Workflow: Decoupled into main program and tests as two git branches

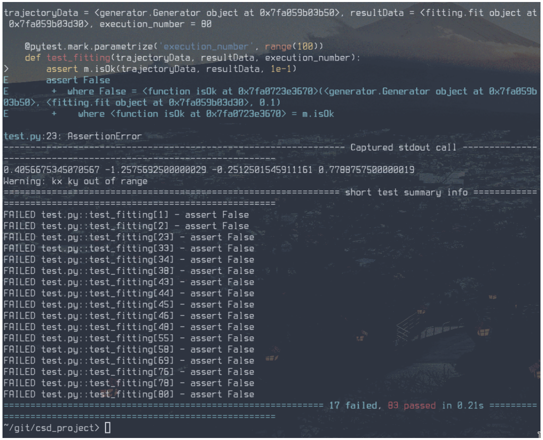
- New features added in main branch
- Run tests
- Fails: Debugging, Pass: Merge into test branch
- New tests deployed
- Run tests
- Pass: Merge into main, Fails: Find out whether problem lies in main or test

Developing in different branches while merging as frequently as possible

Automated Testing

An Debugging Example

One day. . .



```
trajectoryData = <generator.Generator object at 0x7fa059b03b50>, resultData = <fitting.fit object at
 0x7fa059b03d30>, execution_number = 80

    @pytest.mark.parametrize('execution_number', range(100))
    def test_fitting(trajectoryData, resultData, execution_number):
>       assert m.isOk(trajectoryData, resultData, 1e-1)
E       assert False
E        +  where False = <function isOk at 0x7fa0723e3670>(<generator.Generator object at 0x7fa059b
03b50>, <fitting.fit object at 0x7fa059b03d30>, 0.1)
E        +    where <function isOk at 0x7fa0723e3670> = m.isOk

test.py:23: AssertionError
---------------------------------------------------------------- Captured stdout call -------------
----------------------------------------------------
0.4056675345070567 -1.2575692500000029 -0.2512501545911161 0.7780757500000019
Warning: kx ky out of range
================================================================ short test summary info ===========
====================================================
FAILED test.py::test_fitting[1] - assert False
FAILED test.py::test_fitting[2] - assert False
FAILED test.py::test_fitting[23] - assert False
FAILED test.py::test_fitting[33] - assert False
FAILED test.py::test_fitting[34] - assert False
FAILED test.py::test_fitting[38] - assert False
FAILED test.py::test_fitting[43] - assert False
FAILED test.py::test_fitting[44] - assert False
FAILED test.py::test_fitting[45] - assert False
FAILED test.py::test_fitting[46] - assert False
FAILED test.py::test_fitting[48] - assert False
FAILED test.py::test_fitting[55] - assert False
FAILED test.py::test_fitting[58] - assert False
FAILED test.py::test_fitting[69] - assert False
FAILED test.py::test_fitting[76] - assert False
FAILED test.py::test_fitting[78] - assert False
FAILED test.py::test_fitting[80] - assert False
============================================================ 17 failed, 83 passed in 0.21s ========
====================================================
~/git/csd_project> 
```

An Debugging Example

- Collect failing data
- Report to the main developer
- Main developer locate and fix it
- Merge back

An Debugging Example

Reason: Incorrect handling of lines that do not hit all sensors