The testing framework you've been dreaming of

Eli Gur, PyCon IL

May 2nd, 2016
Agenda

• About unit testing
• Ready, Set, Go
• The power of assert
• Catching exceptions
• Fixtures and other features
• Advanced stuff
• Even more advanced stuff
About Unit Testing

- Kent Beck’s sUnit
- Junit
- Python’s unittest
- Py.test
- Hypothesis
Ready, Set, Go: No Boilerplate

• Installation
  `pip install -U pytest`

• Writing a test (in test_something.py):
  `def test_something():
      assert twice(3) == 6
      with raises(ValueError):
          math.sqrt(-2)`

• Running tests:
  `py.test`
Example Run: Success

# py.test *.py --doctest-module --cov --cov-report=term-missing --pep8 --durations=6
================================= test session starts =================================
platform linux -- Python 3.4.3, pytest-2.9.1, py-1.4.31, pluggy-0.3.1
rootdir: /home/pytest-examples, inifile:
plugins: pep8-1.0.6, cov-2.2.1
collected 121 items

eexample1.py ..
test_example1.py ......................................................

-------------------
coverage: platform linux, python 3.4.3-final-0

Name     Stmts     Miss Branch BrPart Cover  Missing
----------------------------------------------------------------
exexample1.py 104      0     62      0   100.0%
test_example1.py 212      0      4      0   100.0%
----------------------------------------------------------------
TOTAL       316      0     66      0   100.0%

================================= slowest 6 test durations ===============================
4.57s setup  test_example1.py::test_smtp
1.15s call   test_example1.py
0.47s call   example1.py
0.24s call   test_example1.py::test_smtp2
0.23s call   test_example1.py::test_smtp1
0.11s call   example1.py::example1.to_datetime

121 passed in 8.33 seconds
Example Run: Failures

```python
def test_upper():
    assert 'foo'.upper() == 'foo'
E    assert 'FOO' == 'foo'
E       assert 'FOO' == 'foo'
E         + foo

test_example1.py:71: AssertionError
```

```python
def test_split():
    s = 'hello world'
>     assert s.split() == ['hello', 'world', 'with', 'error']
E    assert ['hello', 'world'] == ['hello', 'world', 'with', 'error']
E     Right contains more items, first extra item: 'with'
E     Use -v to get the full diff

test_example1.py:87: AssertionError
```

======================================
FAILURES =======================================
_____________________________________
test_upper _______________________________________
```

---
```
def test_upper():
    assert 'foo'.upper() == 'foo'
E    assert 'FOO' == 'foo'
E       assert 'FOO' == 'foo'
E         + foo

```

```
---
```
def test_split():
    s = 'hello world'
>     assert s.split() == ['hello', 'world', 'with', 'error']
E    assert ['hello', 'world'] == ['hello', 'world', 'with', 'error']
E     Right contains more items, first extra item: 'with'
E     Use -v to get the full diff

```

---
```
def test_split():
    s = 'hello world'
>     assert s.split() == ['hello', 'world', 'with', 'error']
E    assert ['hello', 'world'] == ['hello', 'world', 'with', 'error']
E     Right contains more items, first extra item: 'with'
E     Use -v to get the full diff

```

---
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
def test_upper():
    assert 'foo'.upper() == 'foo'
E assert 'FOO' == 'foo'
E - FOO
E + foo

def test_split():
    s = 'hello world'
>    assert s.split() == ['hello', 'world', 'err']
E assert ['hello', 'world'] == ['hello', 'world', 'err']
E Right contains more items, first extra item: 'err'
E Use -v to get the full diff
The Power of Assert (1)

```python
# self.assertEquals(dbl(3), 6)
assert twice(3) == 6

# self.assertTrue('FOO'.isupper())
assert 'FOO'.isupper()

# self assertFalse('foo'.isupper())
assert not 'foo'.isupper()
```
The Power of Assert (2)

# self.assertIs(type(dbl(3)), int)
assert type(twice(3)) is int

# self.assertIsNot(type(dbl(3)), str)
assert type(twice(3)) is not str

# self.assertIsNone(foo(3))
assert foo(3) is None

# self.assertIsNotNone(dbl(3))
assert twice(3) is not None
The Power of Assert (3)

# self.assertIn(dbl(3), [2, 4, 6])
assert twice(3) in [2, 4, 6]

# self.assertNotIn(dbl(3), [1, 2, 3])
assert twice(3) not in [1, 2, 3]

# self.assertIsInstance(foo(3), int)
assert isinstance(foo(3), int)

# self.assertNotIsInstance(foo(3), str)
assert not isinstance(foo(3), str)
The Power of Assert (4)

```python
# self.assertAlmostEqual(dbl(3.00000001), 6)
assert round(twice(3.00000001) - 6) == 0

# self.assertNotAlmostEqual(dbl(3.01), 6)
assert round(twice(3.01) - 6) != 0

# self.assertGreater(foo(3), 5)
assert foo(3) > 5

# assertGreaterEqual, assertLess, assertLessEqual
```
Verifying exceptions

def test_raising_exceptions():
    with raises(ValueError):
        raise_exceptions_example(1)
    ...

Advanced Tests Invocations

```
py.test --failed-first --durations=8 -x
py.test -k something
py.test --pdb --cov
py.test --tb=long -showlocals
py.test -n 8
```
Fixtures: Definition

- Scope: module, class, method, function, session

```python
@fixture(scope="module")
def smtp(request):
    _smtp = smtplib.SMTP("mail.eligur.com")
    def fin():
        _smtp.close()
    request.addfinalizer(fin)
    return _smtp
```
Fixtures: Usage

def test_smtp(smtp):
    response, msg = smtp.noop()
    assert response == 250
    assert msg.lower() == 'ok'

def test_smtp2(smtp):
    response, msg = smtp.ehlo()
    assert response == 250
    assert msg.lower()[0] == 'server208.com'
Parametrize Tests

@pytest.mark.parametrize('obj', [
    '2008-01-31',
    '2008_1_31',
    '2008/01/31',
    '2008.1.31',
    '31/Jan/2008',
    ])
def test_to_date(obj):  # ymd
    assert to_date(obj) == date(2008, 1, 31)
Skipping Tests

```python
slow = pytest.mark.skipif(  
    not pytest.config.getoption("--runslow"),  
    reason="need --runslow option to run"
)

@sslow  
def test_func_slow():  
    ...  
• Can also be simply skipped...
```
Expected Failures

@pytest.mark.xfail
def test_new_feature():
    ...

• Allow conditional
• When test will pass it will be marked **XPASS**
Plugins (a taste of)

- Coverage
- PEP8
- Django
- Twisted
- Flask
- Hypothesis
- Xdist
- Flakes
- Catchlog
- Mock

http://pypi.python.org/pypi?%3Aaction=search&term=pytest-&submit=search
Legacy and CI support

• Support CI, including post-process of test reports:
  `py.test --junitxml=path` # Jenkins, Hudson
  `py.test --resultlog=path` # PyPy-test
  `py.test --pastebin=all` # bpaste.net
• Genscript
• cx_freeze
• Support Tox and virtualenv
• Support unittest automatically (collect + run)
• Support doctest, nose and PEP8
  `py.test --doctest-modules --pep8`
More...

• monkeypatch, tmpdir
• Capturing stdout / stderr / logs
• Incremental testing (test steps)
• Integration with non-Python tests
• Configuration files
  conftest.py
  pytest.ini
  .converagerc
Thank You!