

Unit testing with Python

An introduction on how to unit test in Python. This talk is for Python developers who want to make sure their apps are tested correctly. It steps you through the libraries to use, how to do testing, mocking of external libraries, an overview of testing web apps and how to integrate with continuous integration.

Andy McKay

@andymckay

<http://github.com/andymckay/presentations>

Unit test

A method by which individual units of source are tested to determine if they are fit for use. A unit is the smallest testable part of an application.*

*** Wikipedia: definitive source of everything**

Sample Code

```
def sorted_ci(words):  
    return sorted(words, key=lambda x: x.lower)
```

```
>>> sorted(['apple', 'Orange'])  
['Orange', 'apple']  
>>> from unittest_example import sorted_ci  
>>> sorted_ci(['apple', 'Orange'])  
['apple', 'Orange']
```

Unit tests

```
import unittest

class Test(unittest.TestCase):

    def test_basic(self):
        assert (sorted_ci(['apple', 'Orange'])
                == ['apple', 'Orange'])

if __name__ == '__main__':
    unittest.main()
```

Output

```
$ python example.py
.F..E
=====
ERROR: test_zero (__main__.Test)
-----
..
    1/0
ZeroDivisionError: integer division or modulo by zero

=====
FAIL: test_failure (__main__.Test)
-----
..
    assert not 1 == 0
AssertionError

-----
Ran 5 tests in 0.001s

FAILED (errors=1, failures=1)
```

error vs failure

Testing for failure

```
>>> sorted_ci([4,5])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "unittest_example.py", line 6, in sorted_ci
    return sorted(words, key=lambda x: x.lower)
  File "unittest_example.py", line 6, in <lambda>
    return sorted(words, key=lambda x: x.lower)
AttributeError: 'int' object has no attribute 'lower'
```

Testing for failure

```
class Test(unittest.TestCase):  
  
    def test_basic(self):  
        assert (sorted_ci(['apple', 'Orange'])  
                == ['apple', 'Orange'])  
  
    def test_not_strings(self):  
        self.assertRaises(AttributeError,  
                           sorted_ci, [4, 5])
```

unittest

<http://docs.python.org/library/unittest.html>

Basic starting library and should be your first stop for learning about unit tests.

Organizing tests

```
class Test(unittest.TestCase):  
    def test_some_test(self):  
        ... # Do a test.
```

Organize similar tests into classes. All methods starting with test will be run.

Organizing tests

```
class Test(unittest.TestCase):  
  
    def setUp(self):  
        ... # Before every test.  
        super(Test, self).setUp()  
  
    def tearDown(self):  
        ... # Always runs after every test.  
        super(Test, self).tearDown()
```

Runs before and after every test.

Example setUp

```
class TestPersonalLookup(unittest.TestCase):  
  
    def setUp(self):  
        self.today = date.today()  
        self.addon = Addon(type=amo.ADDON_EXTENSION,  
                           slug='foo')
```

Nose

<http://readthedocs.org/docs/nose/en/latest/>

**...extends unittest to make testing easier.
Adds in lots of plugins.**

Plugins

xunit: outputs CI specific results

progressive: tells you errors right away

coverage: outputs the amount of coverage

And the all important...

pdb

```
$ nosetests example.py --pdb  
.F..> /Users/andy/sandboxes/presentations/confoo-2012/  
unittest_example/example.py(16)test_zero()  
-> 1/0  
(Pdb)
```

Drops you into pdb on a failure.

-S

**“Don’t capture stdout”.
For when your **print** statements go
missing.**

Gotchas

Some things that always* go wrong.

* **Actual results may vary.**

Dates

```
class Test(unittest.TestCase):  
  
    def test_yesterday(self):  
        yesterday = invoice.date()  
        assert datetime.date(2012, 2, 10) == \  
            yesterday
```

Dates

```
class Test(unittest.TestCase):  
  
    def test_yesterday(self):  
        yesterday = invoice.date()  
        assert datetime.date() - timedelta(days=1) == \  
            yesterday
```

Use timedelta to make your dates relative.

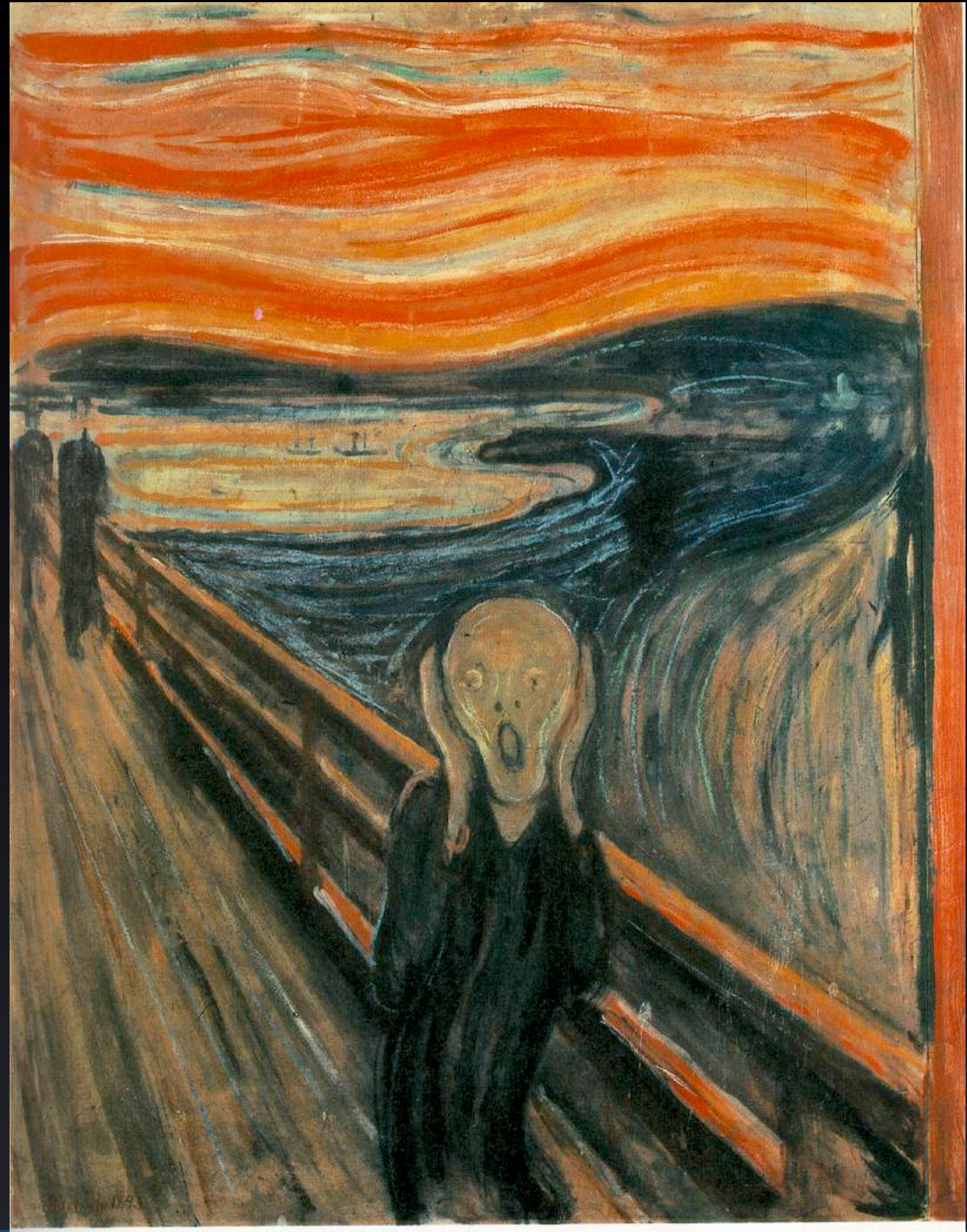
Unicode

```
class Test(unittest.TestCase):  
  
    ...  
  
    def test_unicode(self):  
        assert (sorted_ci([u'বাংলা', 'Orange']))  
                == ['Orange', u'বাংলা'])
```


Python 2

**Unicode is still a pain
and you need to check
to prevent:**

UnicodeDecodeError



Order

```
x = {'a': 'Apple', 'b': 'Banana'}
```

```
class TestSomething(...):  
    def test_order(self):  
        assert x.keys()[0] == 'a'
```

Order

```
x = {'a': 'Apple', 'b': 'Banana'}
```

```
class TestSomething(...):  
    def test_order(self):  
        assert sorted(x.keys())[0] == 'a'
```

mysql, nosql, redis

intermittent and
annoying

Not just for dictionaries. Databases?

Leaking

```
from constants.base import some_dictionary
```

```
class TestSomething(...):
```

```
    def setUp(self):
```

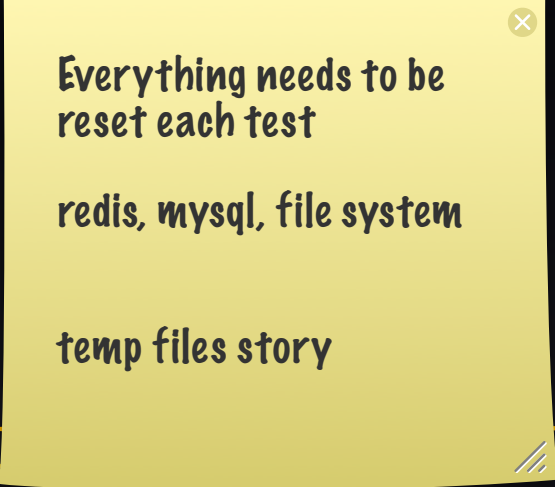
```
        del some_dictionary['some_key']
```

Leaking

```
from constants.base import some_dictionary

class TestSomething():

    def setUp(self):
        our_dictionary = some_dictionary.copy()
        del our_dictionary['some_key']
```



Everything needs to be reset each test

redis, mysql, file system

temp files story

Make sure to copy mutable data.

External

Unit tests should not rely on external sources. *

*** A foolish consistency is the hobgoblin of little minds.**

Examples

APIs from other servers

File system stuff

Anything over HTTP

Payment providers

Examples

APIs from other servers

File system stuff

Anything over HTTP

Payment providers

Databases?

Internal APIs?

External

1. Anything I can't control or fix, within reason.

2. Anything that slows the tests down.

Specific mock

<https://github.com/mozilla/nuggets/>

```
class MockRedis(object):  
    """A fake redis we can use for testing."""  
    ...  
    def incr(self, key):  
        bump = (self.get(key) or 0) + 1  
        self.set(key, bump)  
        return bump
```

Sadly means writing all
the methods etc...

Mock to the rescue

<http://www.voidspace.org.uk/python/mock/>

<http://pypi.python.org/pypi/mock>

...allows you to replace parts of your system under test with mock objects and make assertions about how they have been used.

Mock object

```
>>> mock = Mock()  
>>> mock.method(1, test='wow')  
<Mock name='mock.method()' id='... '>
```

Patch

Alternative to setUp and tearDown.

Altering a method to use the Mock instead of the actual method.

@patch

```
good_response = ( 'responseEnvelope.timestamp='  
                  '2011-01-28T06%3A16...' )
```

```
@mock.patch( 'urllib2.OpenerDirector.open' )  
def test_get_key(self, opener):  
    opener.return_value = StringIO(good_response)  
    assert paypal.get_paykey(self.data) == (  
        'AP-9GD...', 'CREATED' )
```

@patch

```
auth_error = ('error(0).errorId=520003'  
              '&error(0).message=Authentication'  
              '+failed...')  
  
@mock.patch('urllib2.OpenerDirector.open')  
def test_auth_fails(self, opener):  
    opener.return_value = StringIO(auth_error)  
    self.assertRaises paypal.AuthError,  
                      paypal.get_paykey, self.data)
```

@patch

```
@mock.patch('paypal._call')
class TestRefundPermissions(amo.tests.TestCase):

    def test_get_permissions_url(self, _call):
        _call.return_value = {'token': 'foo'}
        assert 'foo' in paypal.get_permission_url\
            (self.addon, '', [])
```

Every method is called with the mock.

Testing calling

```
@mock.patch('paypal._call')
class TestRefundPermissions(amo.tests.TestCase):

    def test_get_permissions_url_scope(self, _call):
        _call.return_value = {'token': '...'}
        paypal.get_permission_url(self.addon, '',
                                   ['REFUND', 'FOO'])
        assert _call.call_args[0][1]['scope'], \
            ['REFUND', 'FOO'])
```

Multiple different call test functions.

@patch gotcha

Sometimes the patch “doesn’t work” as you’d expect.

Fudge

<http://farmdev.com/projects/fudge/index.html>

Fudge is a Python module for using fake objects (mocks and stubs) to test real ones.

Complimentary to mock.

Fudge

```
def sorted_ci(words):  
    return sorted(words, key=lambda x: x.lower)
```

```
def sorted_numbers(numbers):  
    return sorted_ci([str(num) for num in numbers])
```

Fudge

```
class Test(unittest.TestCase):  
  
    @fudge.patch('example.sorted_ci')  
    def test_with_fudge(self, sorted_ci):  
        (sorted_ci.expects_call()  
         .with_args(['5', '4'])  
         .returns(('4', '5')))  
        assert sorted_numbers((5, 4)) == ('4', '5')
```


doc tests

<http://docs.python.org/library/doctest.html>

Documentation is comments.
Personally don't like them.
Hard to refactor.
Code is the documentation.



can also be in rst files.

doc tests

```
def minus(numbers):  
    return reduce(lambda x, y: x - y, numbers)
```

```
>>> from doctest_example import minus  
>>> minus([3,1])  
2
```

doc tests

```
def minus(numbers):  
    """  
    Subtract a list of numbers.  
    >>> minus([3, 1])  
    2  
    >>> minus([3, 2, 4])  
    -3  
    """  
    return reduce(lambda x, y: x - y, numbers)  
  
if __name__ == '__main__':  
    import doctest  
    doctest.testmod()
```

Continuous Integration

Jenkins - use xunit with nose

Buildbot

TeamCity

etc...

Jenkins

[Jenkins » amo-master](#)ENABLE AUTO REFRESH

[Back to Dashboard](#)
[Status](#)
[Changes](#)
[Build Now](#)
[GitHub](#)
[Git Polling Log](#)

Project amo-master

This build is tracking the master branch (trunk). It has the most up to date code and can be seen on [addons-dev.allizom.org](#).

Test Result Trend

The chart displays a series of green bars representing test counts. The y-axis is labeled 'count' and ranges from 0 to 4000 in increments of 500. The x-axis represents time, with labels for each minute from 10:00 to 10:59. The bars show a steady increase in test counts, starting around 2000 and reaching nearly 4000 by the end of the period. There are several thin red vertical lines interspersed among the green bars, indicating test failures.

Build Number	Timestamp
#5531	Feb 21, 2012 7:00:56 PM
#5530	Feb 21, 2012 5:00:55 PM
#5529	Feb 21, 2012 4:30:55 PM
#5528	Feb 21, 2012 3:30:56 PM
#5527	Feb 21, 2012 2:30:55 PM

[\(just show failures\)](#) [enlarge](#)

Questions?

@andymckay

andym@mozilla.com

andym on irc.freenode.net, irc.mozilla.org

<http://github.com/andymckay/presentations>