Python Unittesting

About Me

- Sam Forester
- Systems Administrator for the Marriott Library, University of Utah
- Contact:
  - Email: sam.forester@utah.edu
  - MacAdmins Slack: @sam

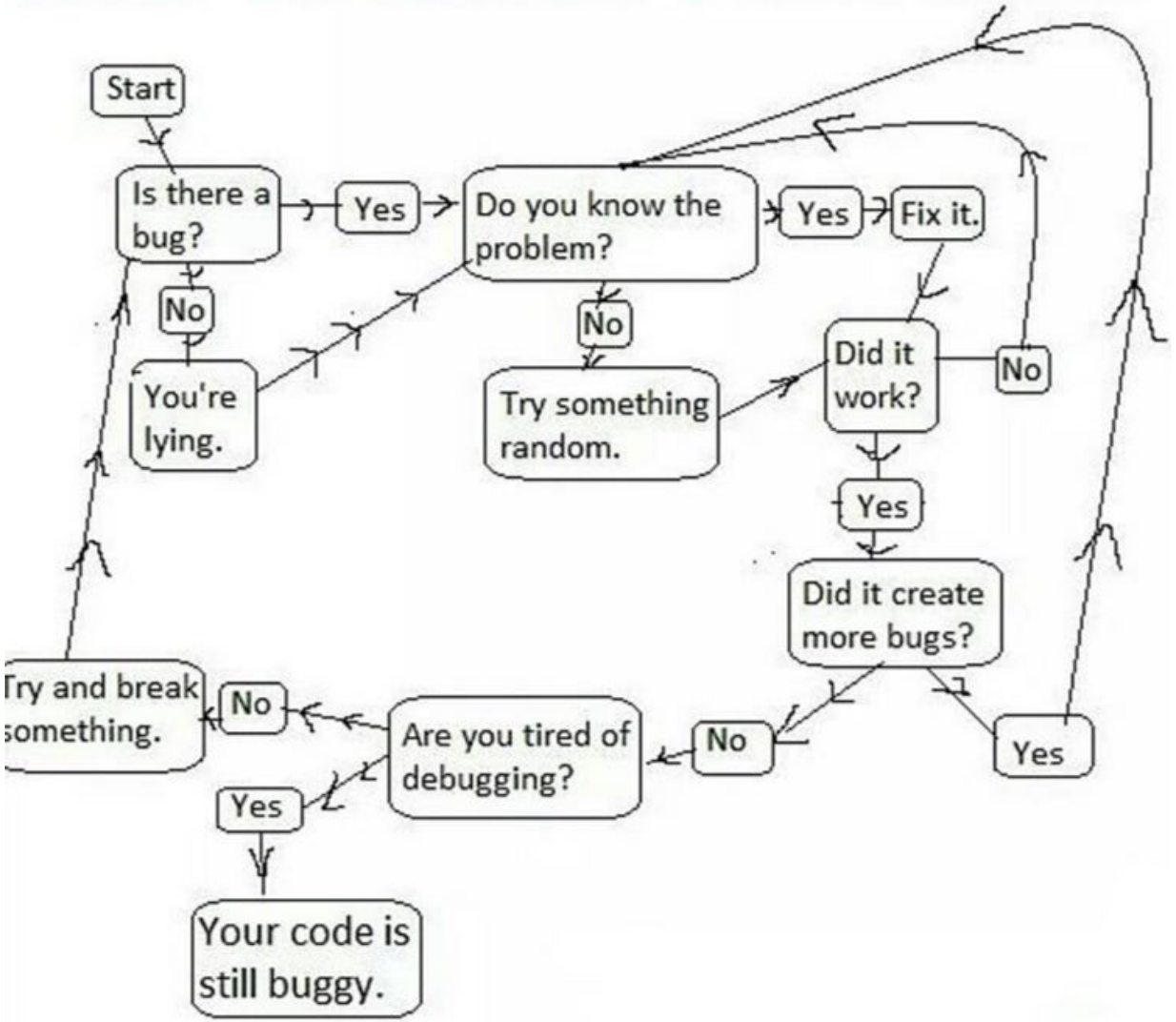
J. Willard Marriott Library
THE UNIVERSITY OF UTAH

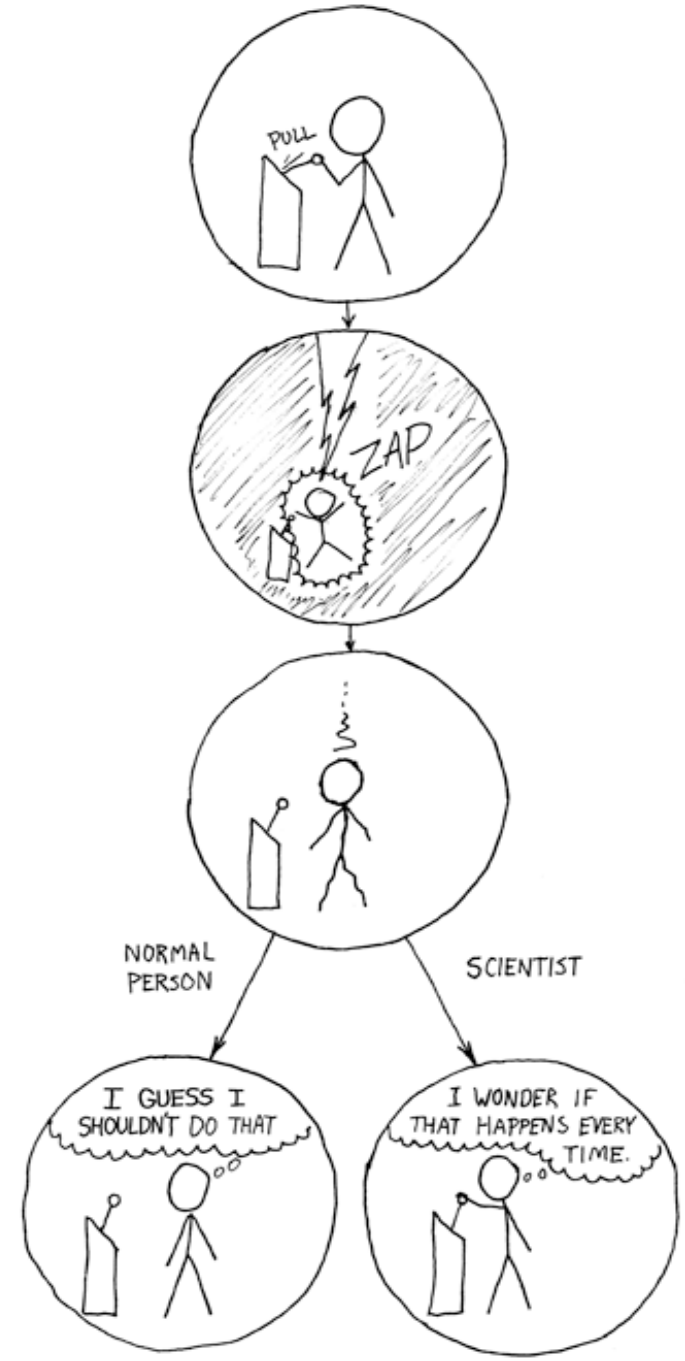
ALL U NEED

Overview

- Introductions
- Test Driven Development
- Unittest module
- Fixtures
- Assertions
- Examples
- Resources

HOW TO DEBUG YOUR CODE





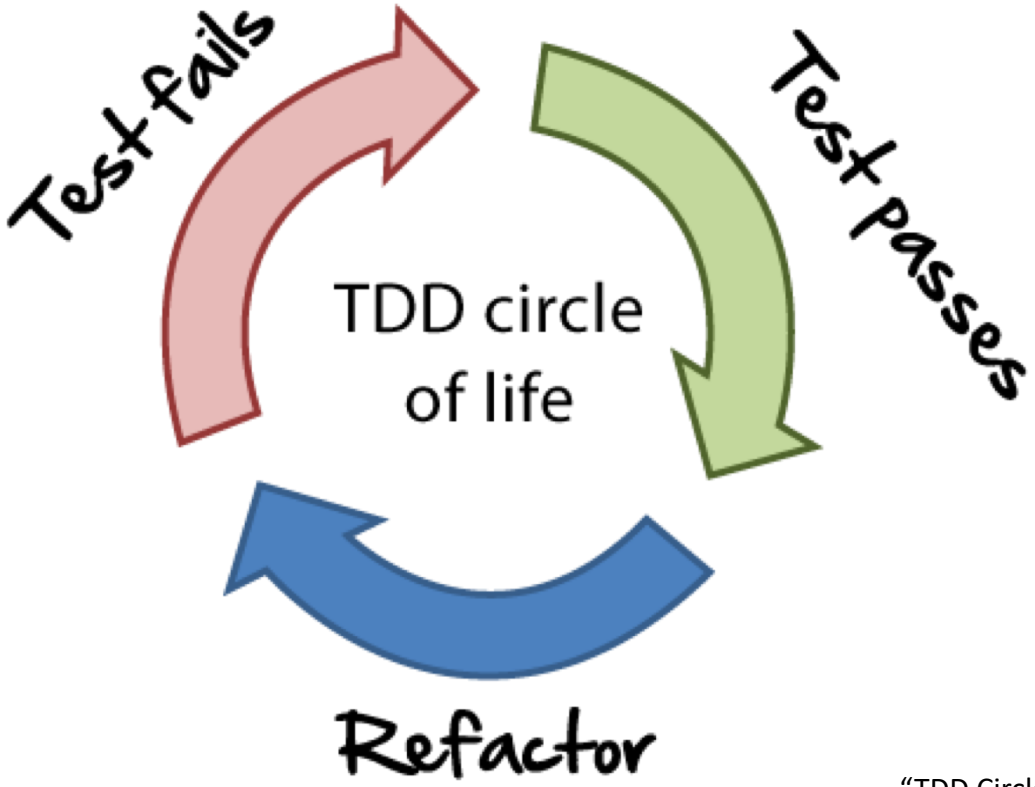
- XKCD 242, The Difference

More About Me

- Classically Trained Developer … (i.e. I wasn't)
- I'm pretty barebones:
  - Terminal, and BBEdit are my preferred tools
- Implementing my own unittests for about 3 months… I'm no expert
- I want to tell you want I've learned and illustrate MY process

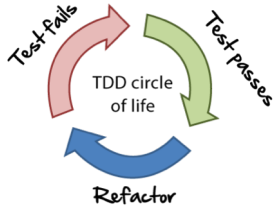
- "The more I learn… The less I know" - Unknown

Test Driven Development (TDD)



- “TDD Circle of Life”

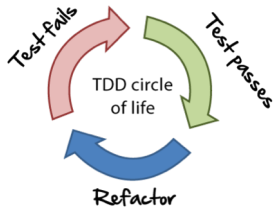
Test Driven Development (TDD)



- Software Development Process.
- Involves writing tests for “most” basic parts of a program (units).
- Tests are collected into suites.
- Suites are “separate” from production code.

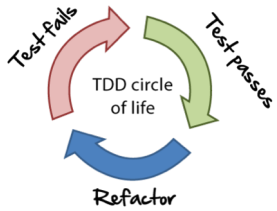
*Disclaimer

Why TDD?

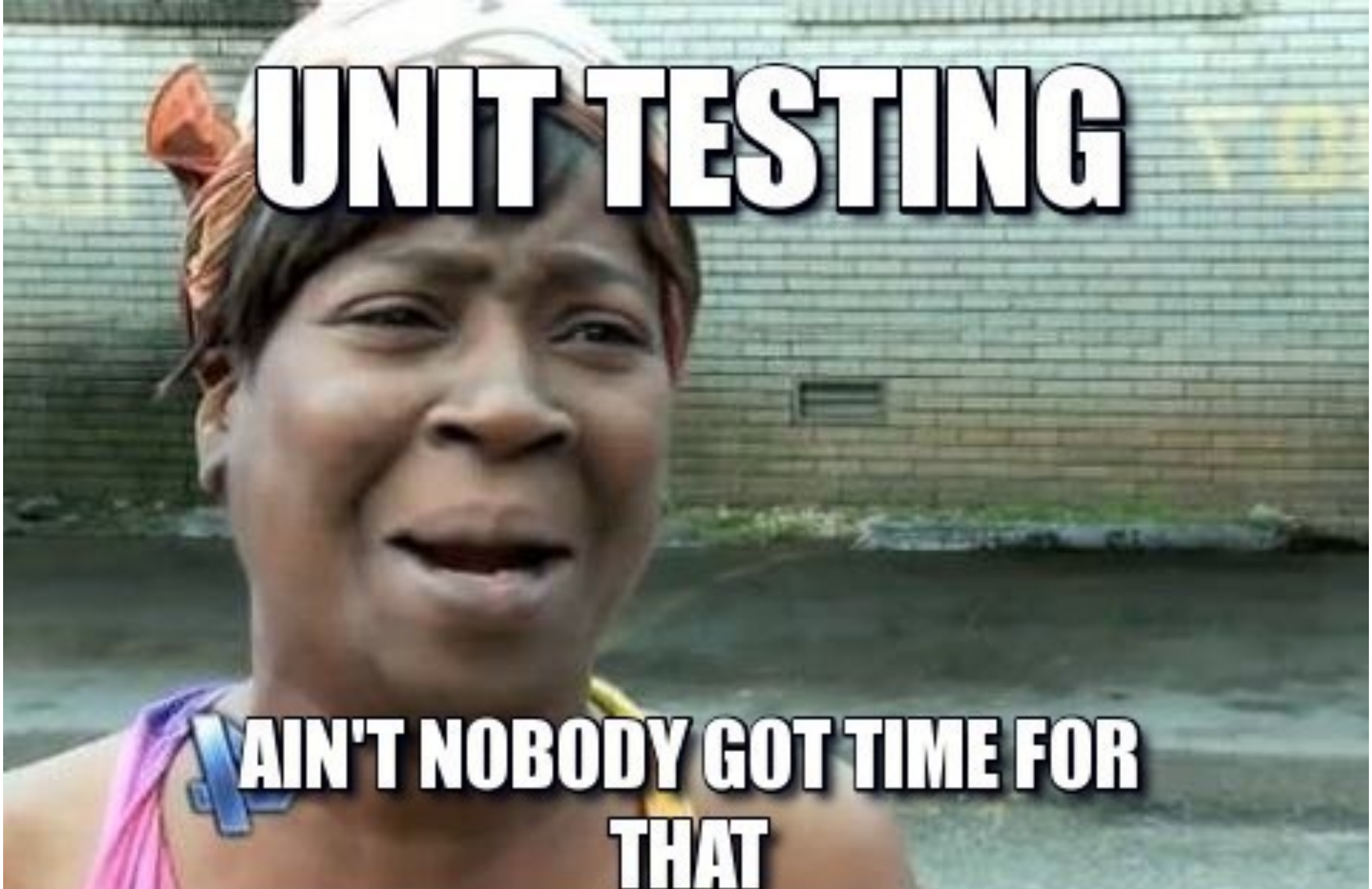


- Debugging:
  - As much help as we can get
- Structure:
  - Using a different approach
- Documentation:
  - Good is amazing! Bad is better than nothing
- Longevity:
  - Tests are forever
- Less Error Prone

TDD Challenges



- Difficult to integrate after code has been written
- Some code is naturally difficult to test:
  - Filesystem Modifications
  - Side Effects
  - Scripts
- Extra work…


UNIT TESTING
AIN'T NOBODY GOT TIME FOR
THAT

Python unittest Module

- Standard Python library since 2.1
- Command-line interface
- Test Discovery
- Lots of documentation
- Provides:
  - Many types of assert functions
  - Setup and teardown between tests
  - Conditional skipping

Typical Online Example

- Many tutorials give examples like this:

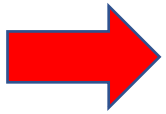
- Thanks! That clears up everything! /s

- These examples are difficult to translate to real world testing.
- However, this example still has some information to offer.

```
import unittest
import myexample

class TestExample(unittest.TestCase):

    def test_math(self):
        x = 3 * 4
        self.assertEqual(x, 12)

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

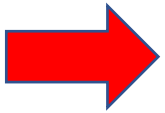
Typical Online Example

- Example of a simple TestCase:
  - TestExample
  - Can contain as many tests as needed
  - Typically at least one positive test and one negative tests



```
import unittest
import myexample

class TestExample(unittest.TestCase):

    def test_math(self):
        x = 3 * 4
        self.assertEqual(x, 12)

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

- Execution:
  - unittest.main()
  - Collects all of the TestCases and runs the tests.



Fixtures

- Setting up tests and cleaning up afterwards
- 3 main fixture pairs:
  - setUpModule() and tearDownModule():
    - Run once at the beginning and end of the entire module
    - If an exception is raised during setUpModule() no tests are run
  - setUpClass() and tearDownClass():
    - One time setup and teardown per TestCase
    - Similar process to setUpModule(), failure keeps any tests in the TestCase from running
  - setUp() and tearDown():
    - Run before and after each test

unittest Fixture Call Stack

```
class BaseTestCaseExample(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        print("setUpClass({0!r})".format(cls))

    @classmethod
    def tearDownClass(cls):
        print("tearDownClass({0!r})".format(cls))

    def setUp(self):
        print("setUp({0!r})".format(self))

    def tearDown(self):
        print("tearDown({0!r})".format(self))
```

```
def setUpModule():
    print("setUpModule({0!r})".format(__name__))

def tearDownModule():
    print("tearDownModule({0!r})".format(__name__))

class TestCaseExample(BaseTestCaseExample):
    def test_example(self):
        print("test_example({0!r})".format(self))

if __name__ == '__main__':
    unittest.main(verbosity=0)
```

Fixture Call Stack

```
[example:~] sam% python test_stack.py
setUpModule(__main__)
setUpClass(<class '__main__.TestCaseExample'>)
setUp(<__main__.TestCaseExample testMethod=test_example>)
test_example(<__main__.TestCaseExample testMethod=test_example>)
tearDown(<__main__.TestCaseExample testMethod=test_example>)
tearDownClass(<class '__main__.TestCaseExample'>)
tearDownModule(__main__)
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
[example:~] sam%
```

Fixture Call Stack

```
[example:~] sam% python test_stack.py
setUpModule(__main__)
setUpClass(<class '__main__.TestCaseExample'>)
setUp(<__main__.TestCaseExample testMethod=test_example>)
test_example(<__main__.TestCaseExample testMethod=test_example>)
tearDown(<__main__.TestCaseExample testMethod=test_example>)
tearDownClass(<class '__main__.TestCaseExample'>)
tearDownModule(__main__)
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
[example:~] sam%
```

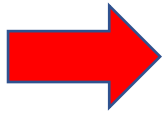
Fixture Call Stack

```
[example:~] sam% python test_stack.py
setUpModule(__main__)
    setUpClass(<class '__main__.TestCaseExample'>)
        setUp(<__main__.TestCaseExample testMethod=test_example>)
            test_example(<__main__.TestCaseExample testMethod=test_example>)
        tearDown(<__main__.TestCaseExample testMethod=test_example>)
    tearDownClass(<class '__main__.TestCaseExample'>)
tearDownModule(__main__)
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
[example:~] sam%
```

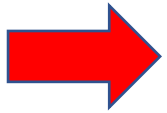
Fixture Call Stack

- Add another TestCase
- Add 2 more tests

```
class TestCaseExample(BaseTestCaseExample):

    def test_example(self):
        print("test_example({0!r})".format(self))
```



```
class TestCaseExampleTwo(BaseTestCaseExample):

    def test_example_2(self):
        print("test_example_2({0!r})".format(self))
```



```
    def test_example_3(self):
        print("test_example_3({0!r})".format(self))
```

- setUpClass(), tearDownClass(), setUp(), and tearDown() defined in BaseTestCaseExample

Fixture Call Stack

```
setUpModule(__main__)
setUpClass(<class '__main__.TestCaseExample'>)
setUp(<__main__.TestCaseExample testMethod=test_example>)
test_example(<__main__.TestCaseExample testMethod=test_example>)
tearDown(<__main__.TestCaseExample testMethod=test_example>)
tearDownClass(<class '__main__.TestCaseExample'>)
setUpClass(<class '__main__.TestCaseExampleTwo'>)
setUp(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
test_example_3<__main__.TestCaseExampleTwo testMethod=test_example_3>)
tearDown(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
setUp(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
test_example_2(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
tearDown(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
tearDownClass(<class '__main__.TestCaseExampleTwo'>)
tearDownModule(__main__)

Ran 3 tests in 0.000s

OK
[example:~] sam%
```

Fixture Call Stack

```
setUpModule(__main__)
setUpClass(<class '__main__.TestCaseExample'>)
setUp(<__main__.TestCaseExample testMethod=test_example>)
test_example(<__main__.TestCaseExample testMethod=test_example>)
tearDown(<__main__.TestCaseExample testMethod=test_example>)
tearDownClass(<class '__main__.TestCaseExample'>)
setUpClass(<class '__main__.TestCaseExampleTwo'>)
setUp(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
test_example_3<__main__.TestCaseExampleTwo testMethod=test_example_3>)
tearDown(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
setUp(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
test_example_2(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
tearDown(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
tearDownClass(<class '__main__.TestCaseExampleTwo'>)
tearDownModule(__main__)
```

Fixture Call Stack

```
setUpModule(__main__)
setUpClass(<class '__main__.TestCaseExample'>)
setUp(<__main__.TestCaseExample testMethod=test_example>)
test_example(<__main__.TestCaseExample testMethod=test_example>)
tearDown(<__main__.TestCaseExample testMethod=test_example>)
tearDownClass(<class '__main__.TestCaseExample'>)
setUpClass(<class '__main__.TestCaseExampleTwo'>)
setUp(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
test_example_3<__main__.TestCaseExampleTwo testMethod=test_example_3>)
tearDown(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
setUp(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
test_example_2(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
tearDown(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
tearDownClass(<class '__main__.TestCaseExampleTwo '>)
tearDownModule(__main__)
```

Fixture Call Stack

```
setUpModule(__main__)
    setUpClass(<class '__main__.TestCaseExample'>)
        setUp(<__main__.TestCaseExample testMethod=test_example>)
            test_example(<__main__.TestCaseExample testMethod=test_example>)
        tearDown(<__main__.TestCaseExample testMethod=test_example>)
    tearDownClass(<class '__main__.TestCaseExample'>)
    setUpClass(<class '__main__.TestCaseExampleTwo'>)
        setUp(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
            test_example_3(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
        tearDown(<__main__.TestCaseExampleTwo testMethod=test_example_3>)
        setUp(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
            test_example_2(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
        tearDown(<__main__.TestCaseExampleTwo testMethod=test_example_2>)
    tearDownClass(<class '__main__.TestCaseExampleTwo '>)
tearDownModule(__main__)
```

Fixture Call Stack

```
setUpModule()
    setUpClass()
        setUp()
            test_example()
        tearDown()
    tearDownClass()
    setUpClass()
        setUp()
            test_example_3()
        tearDown()
        setUp()
            test_example_2()
        tearDown()
    tearDownClass()
tearDownModule()
```

Fixture Call Stack

```
setUpModule()
    setUpClass()
        setUp()
            test_example()
        tearDown()
    tearDownClass()
    setUpClass()
        setUp()
            test_example_3()
        tearDown()
        setUp()
            test_example_2()
        tearDown()
    tearDownClass()
tearDownModule()
```

Fixture Call Stack

```
setUpModule()
    setUpClass()
        setUp()
            test_example()
        tearDown()
    tearDownClass()
    setUpClass()
        setUp()
            test_example_3()
        tearDown()
        setUp()
            test_example_2()
        tearDown()
    tearDownClass()
tearDownModule()
```

Fixture Call Stack

```
setUpModule()
    setUpClass()
        setUp()
            test_example()
        tearDown()
    tearDownClass()
    setUpClass()
        setUp()
            test_example_3()
        tearDown()
        setUp()
            test_example_2()
        tearDown()
    tearDownClass()
tearDownModule()
```

Fixture Call Stack

```
setUpModule()
    setUpClass()
        setUp()
            test_example()
        tearDown()
    tearDownClass()
    setUpClass()
        setUp()
            test_example_3()
        tearDown()
        setUp()
            test_example_2()
        tearDown()
    tearDownClass()
tearDownModule()
```

Fixtures

- setUpModule() and tearDownModule():
  - Defined per module
  - Only run once
- setUpClass() and tearDownClass():
  - Defined per TestCase
  - Only run once per TestCase
- setUp() and tearDown():
  - Also defined per TestCase
  - Run before and after each test
- Unless specifically set with unittest.TestLoader, tests and TestCases can be run in random order.

Fixture Suggestions

- Create a template:
  - Easy to copy a template when making a new testing module
  - Pre-define these fixtures with docstrings, and comments
  - Use pass

- Saves time looking up documentation while trying to discover unittest functionality and integrating tests for yourself

Fixture Template

```
def setUpModule():
    '''One time setup for entire module.
    If exception is raised, no tests in entire module are run.
    '''
    # OPTIONAL
    pass

def tearDownModule():
    '''One time cleanup for entire module.
    '''
    # OPTIONAL
    pass
```

Fixture Template

```
class BasicTestCase(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        '''One time setup for this TestCase. If Exception is raised, no tests are run.
        '''
        # OPTIONAL
        # Useful for expensive actions that need to be completed before
        # tests can be run (e.g. establishing a connection to a database,
        # or creating temporary directories)
        # Also useful for setting constant class variables
        pass
```

Fixture Template

```
def setUp(self):
    '''Runs before each test.
    '''
    # OPTIONAL
    # Useful for setting variables to be used in each test or initializing objects
    pass

def tearDown(self):
    '''Runs after each test.
    '''
    # OPTIONAL
    # put things back the way they were
    pass
```

Assertions

- Unittest comes with a few handy assertions we can use in our tests:
  - assertRaises(Exception):
    - Traps specified Exception, fails if other Exception or no Exception is raised
  - assertTrue(a) and assertFalse(a):
    - a is True, a is False
  - assertEqual(x, y) and assertNotEqual(x, y):
    - x == y, x != y
- 2.7 assertions:
  - assertIn(x, y):
    - Useful for comparing items in data structures
  - assertItemsEqual(x, y):
    - test all elements of x are present in y, regardless of order

Assertion Examples

- assertIn(x, y):
  - Make sure x in y

- Seems basic, but can be very useful with functions that take multiple items, concatenate into a string, and add the string to a list.

```
def test_item_appended(self):
    '''test item was apended
    '''
    t = [1, 2, 3]
    t.append(5)
    self.assertIn(5, t)
```

Assertion Examples

- assertItemsEqual(x, y):
  - Make sure x and y have the same values, regardless of order
- Incredibly useful for testing command building for subprocess

```
def test_items_similar(self):
    '''test lists have all items
    '''
    x = [1, 'two', 3]
    y = [3, 'two', 1]
    self. assertItemsEqual(x, y)
```

Assertion Examples

- assertRaises(Exception):
  - Traps specified Exception
  - Succeeds if specified Exception is raised
  - Fails:
    - Different Exception is raised
    - No Exception is raised
- Great for negative testing and expected failures.

```
def test_remove_missing_file(self):
    '''test os.remove on missing file
    '''
    with self.assertRaises(OSError):
        os.remove('/no/such/file')
```

Assertion Not Necessary

- Not every test needs an assertion
- Sometimes you just want to make sure no exceptions are raised

```
def test_no_exception(self):
    '''test exception not raised on assignment
    '''
    x = True
```

Real World Tests How Do I Test This?

- Delete some file if it exists when script begins
- Things we might want to test:
  - Was file present?
  - Was file actually deleted?
- Challenges:
  - Dependent on external state
  - Side Effects

```
import os

def main():
    file = '/path/to/some/file'
    if os.path.exists(file):
        os.remove(file)

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

Real World Tests How Do I Test This?

- My Previous Method:
  - “When in doubt, print it out!”
  - Extra logic, but at least we know what’s happening.
  - Logging statements for debugging
  - Check log if something fails
  - Run a few times in the real world, change if necessary, and trust it will work like it did when it was written (fingers crossed!)
- There is a better way! But we’ll have to give it some thought.

```
def main():
    logger = logging.getLogger(__name__)
    file = '/path/to/some/file’
    if os.path.exists(file):
        logger.debug("file exists: {0}".format(file))
        try:
            logger.debug("removing: {0}".format(file))
            os.remove(file)
        except:
            err = "failed to remove: {0}".format(file)
            logger.error(err)
            raise SystemExit(err)
    else:
        logger.debug("file doesn't exist: {0}".format(file))
```

Building Real World Tests

- Rethinking our approach
- What are we really trying to accomplish?
  - Delete a triggerfile at the beginning of a script.
  - Do we care if it's missing?
  - What happens on failure?
  - How can it fail?

```
def remove_triggerfile(file):
    '''remove specified file, unless it is already missing
    '''
    try:
        os.remove(file)
    except OSError as e:
        # re-raise any errors except "no such file"
        if e.errno != 17:
            raise
```

Building Real World Tests

- Create a TestCase
- Add setUp() and tearDown():
  - Set the file we are going to test
  - Before the test runs, create the file
  - Remove the file after each test
- If either setUp() or tearDown() fails, the test fails…
- Write a test:
  - test_remove_triggerfile()

```
class TestRemoveTriggerfile(unittest.TestCase):

    def setUp(self):
        self.file = '/tmp/triggerfile_test'
        open(self.file, 'a').close()

    def tearDown(self):
        if os.path.exists(self.file):
            err = "file still exists: {0}".format(self.file)
            raise RuntimeError(err)

    def test_remove_triggerfile(self):
        '''test the file is actually removed
        '''
        myexample.remove_triggerfile(self.file)
```

Building Real World Tests

- Add more tests:
- test_remove_missing_triggerfile()
  - Remove the file created in setUp() before running the tested function
  - No errors should be raised

```
class TestRemoveTriggerfile(unittest.TestCase):
    …

    def test_remove_missing_triggerfile(self):
        '''test no error is raised when file is missing
        '''
        os.remove(self.file) # file created in setUp()
        myexample.remove_triggerfile(self.file)
```

Building Real World Tests

- Keep adding tests!
- test_remove_unremovable()
  - Set the triggerfile to 'uchg'
  - Trap the error that should be raised (test will fail if OSError is not raised)

```
class TestRemoveTriggerfile(unittest.TestCase):
    …

    def test_remove_unremovable(self):
        '''test Exception is raised when file cannot be removed
        '''
        subprocess.call(['chflags', 'uchg', self.file])
        with self.assertRaises(OSError):
            myexample.remove_triggerfile(self.file)
```

Running Tests

```
[example:~] sam% python test_myexample.py
```

J. Willard Marriott Library
THE UNIVERSITY OF UTAH

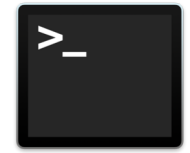
ALL U NEED

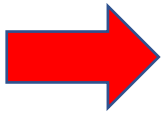
```
[example:~] sam% python test_myexample.py
E.E
======================================================================
ERROR: test_remove_missing_triggerfile (__main__.TestRemoveTriggerfile)
test no error is raised when file is missing
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 30, in test_remove_missing_triggerfile
    myexample.remove_triggerfile(self.file)
  File “/Users/example/myexample.py", line 7, in remove_triggerfile
    os.remove(file)
OSError: [Errno 2] No such file or directory: '/tmp/triggerfile_test'

======================================================================
ERROR: test_remove_unremovable (__main__.TestRemoveTriggerfile)
test OSError is raised when file cannot be removed
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 18, in tearDown
    raise RuntimeError(err)
RuntimeError: file still exists: /tmp/triggerfile_test

----------------------------------------------------------------------
Ran 3 tests in 0.008s

FAILED (errors=2)
[example:~] sam%
```

Running Tests - Errors

```
E.E
======================================================================
ERROR: test_remove_missing_triggerfile (__main__.TestRemoveTriggerfile)
test no error is raised when file is missing
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 30, in test_remove_missing_triggerfile
    myexample.remove_triggerfile(self.file)
  File "/Users/example/myexample.py", line 7, in remove_triggerfile
    os.remove(file)
OSError: [Errno 2] No such file or directory: '/tmp/triggerfile_test'
```

Running Tests - Errors

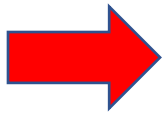




```
E.E
======================================================================
ERROR: test_remove_missing_triggerfile (__main__.TestRemoveTriggerfile)
test no error is raised when file is missing
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 30, in test_remove_missing_triggerfile
    myexample.remove_triggerfile(self.file)
  File "/Users/example/myexample.py", line 7, in remove_triggerfile
    os.remove(file)
OSError: [Errno 2] No such file or directory: '/tmp/triggerfile_test'
```

Error Docstrings

- Test docstrings are automatically printed if an error occurs.
- Twice the documentation for half the effort
- Caveats:
  - Only prints one line
- If a test requires too much documentation, maybe it's too complicated



```
class TestRemoveTriggerfile(unittest.TestCase):
    …

    def test_remove_missing_triggerfile(self):
        '''test no error is raised when file is missing
        '''
        os.remove(self.file) # file created in setUp()
        myexample.remove_triggerfile(self.file)
```

So, What Happened?

- Apparently it choked trying to remove a missing file…
- I thought I had coded for that

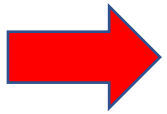
```
def remove_triggerfile(file):
    '''remove specified file, unless it is already missing
    '''
    try:
        os.remove(file)
    except OSError as e:
        # re-raise any errors except "no such file"
        if e.errno != 17:
            raise
```

```
OSError: [Errno 2] No such file or directory: '/tmp/triggerfile_test'
```

- Oops… Wrong error…

Ok, Let's Fix It!

- We can adjust our code without modifying any of the tests
- Now… let's re-run the tests!

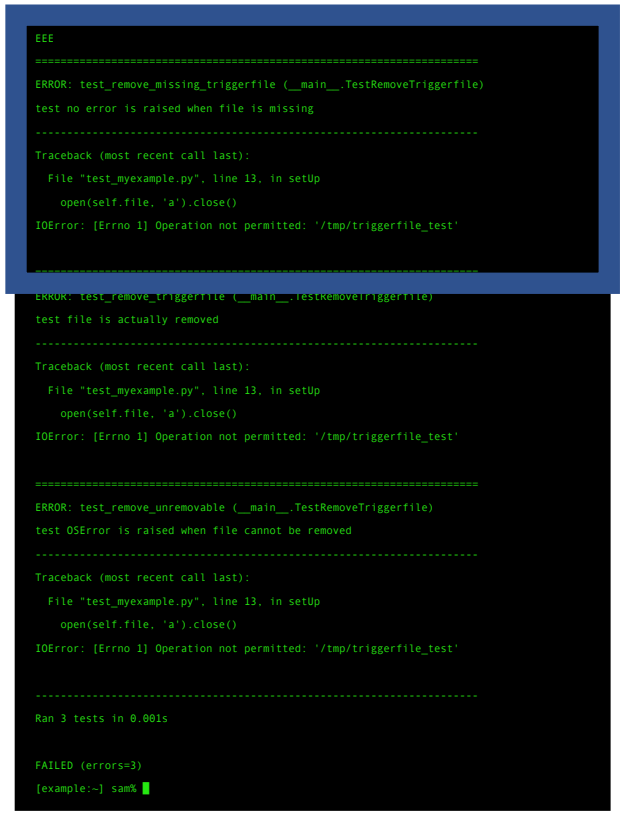


```
def remove_triggerfile(file):
    '''remove specified file, unless it is already missing
    '''
    try:
        os.remove(file)
    except OSError as e:
        # re-raise any errors except "no such file"
        if e.errno != 2: # changed from 17 to 2
            raise
```

```
EEE
======================================================================
ERROR: test_remove_missing_triggerfile (__main__.TestRemoveTriggerfile)
test no error is raised when file is missing
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 13, in setUp
    open(self.file, 'a').close()
IOError: [Errno 1] Operation not permitted: '/tmp/triggerfile_test'

======================================================================
ERROR: test_remove_triggerfile (__main__.TestRemoveTriggerfile)
test file is actually removed
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 13, in setUp
    open(self.file, 'a').close()
IOError: [Errno 1] Operation not permitted: '/tmp/triggerfile_test'

======================================================================
ERROR: test_remove_unremovable (__main__.TestRemoveTriggerfile)
test OSError is raised when file cannot be removed
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 13, in setUp
    open(self.file, 'a').close()
IOError: [Errno 1] Operation not permitted: '/tmp/triggerfile_test'

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (errors=3)
```

Wait… What!?

- Now we have 3 errors!?
- We fixed the code, shouldn't we have one less error!?



*sigh… Why am I doing this again?

Fine… what happened?…

J. Willard Marriott Library
THE UNIVERSITY OF UTAH

ALL U NEED

Running Tests - Errors

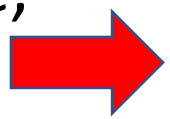
```
EEE
======================================================================
ERROR: test_remove_missing_triggerfile (__main__.TestRemoveTriggerfile)
test no error is raised when file is missing
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 13, in setUp
    open(self.file, 'a').close()
IOError: [Errno 1] Operation not permitted: '/tmp/triggerfile_test'
```

Hmm…

- Same test, different error
- Looks like setUp() failed trying to create the file
  - Automatic failure…
- Error:
  - IOError [Errno 1] Operation not permitted: '/tmp/triggerfile_test’

The Wonderful World of Side Effects!

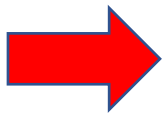
- My bad…
- test_remove_unremovable()
  - **Set the triggerfile to 'uchg'**
- By making the file unmodifiable, we broke the rest of our tests…
- The file was left behind after the tests completed.
- Lesson:
  - If you're going to change something, be sure to undo it



```
class TestRemoveTriggerfile(unittest.TestCase):
    …

    def test_remove_unremovable(self):
        '''test Exception is raised when file cannot be removed
        '''
        subprocess.call(['chflags', 'uchg', self.file])
        with self.assertRaises(OSError):
            myexample.remove_triggerfile(self.file)
```

The Wonderful World of Side Effects!

- Let's add a bit to undo our chflags command
- Oh! And let's not forget to remove the locked file we created by running our tests the first time
- I think that will do it! Time to test!



```
class TestRemoveTriggerfile(unittest.TestCase):
    …

    def test_remove_unremovable(self):
        '''test Exception is raised when file cannot be removed
        '''
        subprocess.call(['chflags', 'uchg', self.file])
        with self.assertRaises(OSError):
            myexample.remove_triggerfile(self.file)

        # undo chflags and delete the file or tearDown() will fail
        subprocess.call(['chflags', 'nouchg', self.file])
        os.remove(self.file)
```

J. Willard Marriott Library
THE UNIVERSITY OF UTAH

ALL U NEED

Running Tests… Again…

```
[example:~] sam% python test_myexample.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.013s

OK
[example:~] sam%
```

No Errors! Wait…

- Cool that we didn't get any errors, but didn't we have two errors to begin with? … seems suspicious.
- Let's break it again and find out!

- Let's break it again on purpose by changing OSError back to 17 in our original code and see what happens.

Running Tests… Yet Again…

```
[example:~] sam% python test_myexample.py
E..
======================================================================
ERROR: test_remove_missing_triggerfile (__main__.TestRemoveTriggerfile)
test no error is raised when file is missing
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 30, in test_remove_missing_triggerfile
    myexample.remove_triggerfile(self.file)
  File "/Users/example/myexample.py", line 7, in remove_triggerfile
    os.remove(file)
OSError: [Errno 2] No such file or directory: '/tmp/triggerfile_test'

----------------------------------------------------------------------
Ran 3 tests in 0.011s

FAILED (errors=1)
[example:~] sam%
```

More Errors…

- Same test…
- Same failure…
- Same error…

- Awesome!

- Let’s fix it again!

```
[example:~] sam% python test_myexample.py
E..
======================================================================
ERROR: test_remove_missing_triggerfile (__main__.TestRemoveTriggerfile)
test no error is raised when file is missing
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test_myexample.py", line 30, in test_remove_missing_triggerfile
    myexample.remove_triggerfile(self.file)
  File "/Users/example/myexample.py", line 7, in remove_triggerfile
    os.remove(file)
OSError: [Errno 2] No such file or directory: '/tmp/triggerfile_test'

----------------------------------------------------------------------
Ran 3 tests in 0.011s

FAILED (errors=1)
[example:~] sam%
```

Running Tests… One More Time…

```
[example:~] sam% python test_myexample.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.013s

OK
[example:~] sam%
```

What Actually Happened…

- Only focused on the first error.
- The second Error happened in tearDown() when the file still existed after removing the un-removable file.
  - Actually, had we tried to remove the file in the test, we could get 4 errors on three tests.
  - (apparently) tearDown() counts as another failure beyond a normal test failure.

- … This example happened while I was creating this presentation

Another Approach

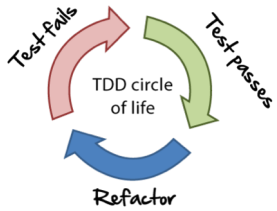
- Test may be a little too complicated…
- Write setUp() so that it checks to see if the file exists before creating the triggerfile.
- More?

```
class TestRemoveTriggerfile(unittest.TestCase):
    …

    def test_remove_unremovable(self):
        '''test Exception is raised when file cannot be removed
        '''
        subprocess.call(['chflags', 'uchg', self.file])
        with self.assertRaises(OSError):
            myexample.remove_triggerfile(self.file)

        # undo chflags and delete the file or tearDown() will fail
        subprocess.call(['chflags', 'nouchg', self.file])
        os.remove(self.file)
```

Conclusion

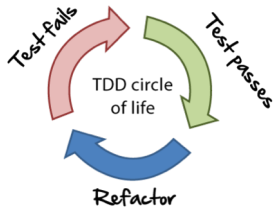
- Write down what you're testing!
  - Docstrings are automatically printed in the event of a failure, you'll thank yourself later.
- Test frequently and often:
  - Make sure your tests work, before writing more.
- If you don't need a fixture, that's fine!
  - A well-written setUp() and no/one-line tearDown() is normal
- Beware of side effects!
  - Unexpected failure is one thing, misleading success is much worse

Conclusion Why TDD?



- Debugging
  - If you have to debug once, write a test, and keep it forever
  - Tests provide instant feedback!
- Structure
  - Testing leads to writing your code in smaller, better defined segments that are easier to test
  - Requires you to really think about what you are trying to accomplish
- Documentation
  - Illustrate your intentions in each tested function.
  - Docstrings are automatically printed on failure (double the benefit, with half the work)

Conclusion Why TDD?



- Longevity
  - Once a test has been written, it doesn't need to be re-written, it just needs to pass
  - Existing tests will ensure reverse-compatibility after refactoring your code.
- Less Error Prone:
  - Tests provide instant feedback!
    - Change something, and test it!
  - Making a change and then watching all 130 tests pass… Feels fantastic!
    - Code with confidence

Additional Resources

- Python unittest Documentation:
  - https://docs.python.org/2/library/unittest.html
- Unittest introduction:
  - http://pythontesting.net/framework/unittest/unittest-introduction/
- Python Unit Testing Tutorial:
  - http://cgoldberg.github.io/python-unittest-tutorial/

Additional Resources

- Presentations:
  - Unit Testing Concepts and Best Practices
    - https://www.slideshare.net/homespothq/unit-testing-concepts-and-best-practices
  - Unit Testing with Python
    - https://www.slideshare.net/micropyramid/unit-testing-with-python
  - Python: Object-Oriented Testing (Unit Testing)
    - https://www.slideshare.net/DamianGordon1/python-objectoriented-testing-unit-testing

Works Cited

- Jonathan Rasmusson, “TDD Circle of Life”, Test Driven Development, http://www.agilenutshell.com/assets/test-driven-development/tdd-circle-of-life.png
- XKCD, “The Difference”, https://xkcd.com/242/
- Unknown, “How To Debug Your Code”

Questions?