

# Testing in Python At PyCon Oh-Nine

(say that ten times fast)

Aaron Maxwell  
amax@redsymbol.net  
<http://redsymbol.net/>

29 May 2009  
Bay Area Python Interest Group  
hosted by Symantec

this talk archived at <http://redsymbol.net/talks/>

# Testing Talks

About ten talks

<http://us.pycon.org/2009/conference/talks/?filter=testing>

Three BOFs

And some lightning talks.

Theme #1:

# CODE COVERAGE

Presenters include Ned Batchelder  
(`coverage.py`), C. Titus Brown (`figleaf`)

# code coverage: swiss army knife

What's a code coverage tool useful for?

**testing**, of course

And several other things...

help understand large, strange code bases

a code analysis tool

suggesting dependencies and potential bottlenecks

# CC tools: tackling legacy code

“Building Tests for Large, Untested Codebases”

C. Titus Brown

legacy code,  $n$ :

code without tests

Use the code coverage tool to clarify what does what.

Exercise one function (method, etc.) and examine the code executed from it.

# “Coverage Testing: The Good and the Bad”

Ned Batchelder

Survey of CC blessings and pitfalls

Code coverage tools are useful. But they have some limits that are important to understand.

```
# Limits of statement coverage!  
def branch(thing):  
    return thing or x + 2  
# 100% code coverage, no error  
branch(2) == 2  
# raises NameError  
branch(0) == 42
```

# The Achilles Heel of CC Tools

Current Python code-coverage tools only do *statement* coverage.

This actually misses a lot.

Other kinds of coverage:

- branch

- path

- loop (special case of “path”)

- coverage of data-driven code

# Data-Driven code...

... is awesome. But covering it is hard.

```
divisors = {  
    'x' : 1,  
    'y' : 0,  
}
```

```
# Bug is in the data, not the function itself  
def data_driven(thing):  
    return 2/divisors[thing]
```

# The FUTURE of code coverage

## **BYTECODE** coverage!

Use `sys.settrace` as normal

However, modify code object's line number table

Each bytecode is its own “line”

Powerful and very fine grain coverage!  
(Potentially.)

Proof of concept done. Weaponization in progress... watch <http://nedbatchelder.com>

Theme #2:

# FUNCTIONAL TESTING

Lots of presenters, including a whole functional testing panel

## Functional testing:

Tests that drive the system from the outside... from the point of view of the user, making assertions about the application's behavior and features.

# FT Panel

“Functional Testing Tools in Python”

Adam D. Christian and Mikeal Rogers (Windmill)

Holger Krekel (py.test)

Jason Huggins (Selenium)

Ian Bicking

Kumar McMillan (nose)

C. Titus Brown (twill)

Terry Peppers (awesome moderator)

Holger Krekel (of py.test):

One characteristic of a successful testing tool:

The tool is able to be helpful when tests *fail*.

If it's not:

the test tool needs to be improved, or

different tests are needed

Other FT panel takeaways:

A variety of different functional testing tools

Each has different strengths and shortcomings

The TIP (testing in python mailing list) is where  
the cool kids are

# “Functional Testing of Desktop Applications”

by Michael J. Foord

(Resolver Systems Ltd)

Many claim that automation of GUI testing is impossible or not worth the trouble.

Counterpoint: Resolver's home-grown FT framework

# Resolver's FT framework

xUnit-like

```
class FunctionalTest(unittest.TestCase):
```

Key idea: a functional test can act as an  
*executable specification*

Use user stories. Specified in terms of user actions  
and expected results

Overall, impressive. Yet still illustrates that  
good FT frameworks are in relative infancy

Theme #3:

# TESTING TOOLS

nose

py.test

# py.test

Release 1.0 *imminent*

plugin arch

more easily extend reporting, supported test types,  
etc.

distributed testing

run tests on different platforms and machines

better support for parameterized tests

And more. See <http://pytest.org/>

# nose

Version 0.11 released! (as of May 7)

parallize test runs (--processes)

run only last batch's failing test with --failed switch

test collection

And more. See <http://farmdev.com/thoughts/78/nose-0-11-released-nifty-new-features/>

# Unsolved Problems

Automating functional tests is still hard

Limitations of code coverage tool granularity

With code bases that have lots of tests: how do you know that tests aren't disappearing on you?

Test reporting. TAP may be our salvation (<http://testanything.org>)

# Thanks!



to the Python community  
for Pycon

to those who gave talks  
to Wesley, Daryl, Jim and  
Glen

to Symantec for hosting  
this meeting

to YOU for being here!

This talk archived at  
<http://redsymbol.net/talks/>