Introduction to Python

Christopher Barker

UW Continuing Education / Isilon

August 29, 2012

Table of Contents

Review of Previous Class

- Decorators
- Debugging
- Packages and Packaging

Lightning talk today: Chris

Homework review

Did any of you find a package you couldn't install?

Did any of you write a `setup.py` for you own code?

Testing

A bit strange to be "teaching" testing to a room full of professional testers.

Testing

I don't need to tell you why testing is important

I'll focus on testing your Python code – not another system

Mostly unit testing

And an introduction to some of the tools

Python Testing Taxonomy

1. Unit Testing Tools
2. Mock Testing Tools
3. Fuzz Testing Tools
4. Web Testing Tools
5. Acceptance/Business Logic Testing Tools
6. GUI Testing Tools
7. Source Code Checking Tools
8. Code Coverage Tools
9. Continuous Integration Tools
10. Automatic Test Runners
11. Test Fixtures
12. Miscellaneous Python Testing Tools

`http://wiki.python.org/moin/PythonTestingToolsTaxonomy`

Testing

Regression Testing:

Regression testing is any type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system after changes

Unit Testing

unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine if they are fit for use

doctest

doctest

In the stdlib

Unique to Python?

Literate programming

verify examples in in docs

`http://docs.python.org/library/doctest.html`

doctest

doctest example

```
def get(self):
    """ get() -> return TestClass's associated value.
    >>> x = _TestClass(-42)
    >>> print x.get()
    -42
    """
    return self.val
```

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

doctest

As mentioned in the introduction, doctest has grown to have three primary uses:

- Checking examples in docstrings. Regression testing.
- Executable documentation / literate testing.

These uses have different requirements, and it is important to distinguish them. In particular, filling your docstrings with obscure test cases makes for bad documentation.

`http://docs.python.org/library/doctest.html`

running doctests

In `__name__ == "__main__"` block:

```
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```

(Tests the current module)

`http://docs.python.org/library/doctest.html`

running doctests

In an external file:

```
doctest.testmfile("name_of_test_file.txt")
```

Tests the docs in the file (reSt format...)

With a test runner: more on that later

`http://sphinx.pocoo.org/`

running doctests

In a documentation system:

Sphinx:

Sphinx is a tool that makes it easy to create intelligent and beautiful documentation

– Lots of output options: html, pdf, epub, etc, etc...

– Can auto-generate docs from docstrings

– And run the doctests

`http://sphinx.pocoo.org/`

Code Checking

Not Really testing, but...

pychecker
`http://pychecker.sourceforge.net/`

pylint
`http://www.logilab.org/857/`

pyflakes
`http://pypi.python.org/pypi/pyflakes`

Will help you keep your source code clean

pyunit

Not Really testing, but...

LAB

Testing LAB:

- doctest
- unittest
- pytest
- run doctest with nose, pytest

Lightning Talk

Lightning Talk:

Peter

Performance Testing

"Premature optimization is the root of all evil"

LAB

Profiling lab

Wrap up

- something

Next Week:

Votes are in:

4 votes for persistence: 0 for Extending with C

Persistence it is