



Tests in your Docstrings: Using the **doctest** module

Wesley J. Chun

wescpy@gmail.com

Software Engineer and Author of
Core Python Programming
& *Python Fundamentals*

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



Commenting and Documentation

- Language syntax allows for documentation
 - Applicable to functions, classes, and modules
- "Docstrings:" 1st unassigned string in suite

```
#!/usr/bin/env python

"docstring for entire module goes here"

class User(object):
    "User class docstring goes here"
    pass

def foo():
    "function foo() docstring goes here"
    pass
```

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



Documentation Strings

- Docstrings better than "comments;" accessible at run-time:

```
def foo(x):
    'foo(x) - display argument "x"'
    print x

>>> help(foo)
Help on function foo in module __main__:

foo(x)
    foo(x) - display argument "x"

>>> foo.__doc__
'foo(x) - display argument "x"'
```

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



Effective Docstrings have Code

- Useful comments: help out fellow coders!
- Put example usage/output in a docstring:

```
"doctestDemo.py - demo doctest module"

def foo(x):
    """foo(x): display argument 'x'

    >>> foo(123)
    123
    """
    print x
```

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



doctest Module

- Add auto-regression testing to your code:

```
def _test():
    import doctest
    doctest.testmod()

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

- If everything worked, there will be no output:

```
$ doctestDemo.py
$
```

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



Verbose doctest Output

```
$ doctestDemo.py -v
Trying:
    foo(123)
Expecting:
    123
ok
3 items had no tests:
    __main__
    __main__.User
    __main__._test
1 items passed all tests:
   1 tests in __main__.foo
1 tests in 4 items.
1 passed and 0 failed.
Test passed.
```

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



Combining Unit Tests and Doctests

- Using the `unittest` module
 - Combine doctests with unit tests

```
import unittest, doctest, doctestDemo

class TestDocTestDemo(unittest.TestCase):
    # run doctests
    def test_docstrings(self):
        doctest.testmod(doctestDemo)

    # run main regression and unit tests
    def test_regression_and_unit_tests(self):
        self.assert_(...)
        :
if __name__ == '__main__':
    unittest.main()
```

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



Running Unit Tests *and* Doctests!

- Running your unit tests as normal
- Doctests executed transparently!

```
$ python test_doctestDemo.py
```

```
..
```

```
-----  
Ran 2 tests in 0.061s
```

```
OK
```

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



Run Doctests with Nose

●Nose

<http://somethingaboutorange.com/mrl/projects/nose>

●Automatically run doctests with Nose

●Use `--with-doctest` option

●Can also put in `.noserc` or `nose.cfg`

●Also see other options

●`--doctest-tests` and `--doctest-extension`

```
$ nosetests doctestDemo.py --with-doctest
```

```
.
```

```
-----  
Ran 1 test in 0.005s
```

OK

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.



Epydoc: Autogenerate Documentation

●Epydoc <http://epydoc.sf.net>

●Automatically generate HTML, PDF, etc.

```
def foo(x):  
    """foo(x): display argument 'x'  
  
    @type x: any Python object  
    @param x: the object to display  
    @rtype: None  
    @return: there is no output; always return None  
    @author: wesc (originally created Summer 2008)  
  
    >>> foo(123)  
    123  
    """  
    print x
```

(c) 1998-2009 CyberWeb Consulting.
All rights reserved.