Unit Testing in Python

James Brucker

Python Testing Frameworks

- unittest - part of the Python distro, similar to JUnit 3
- DocTest - test by example, also part of Python distro
- Py.Test - very simple "assert" syntax
- Nose
- Many testing frameworks:

  https://wiki.python.org/moin/PythonTestingToolsTaxonomy

So many choices... you have no excuse to <u>not</u> test.

unittest simple example

class extends TestCase

```
import unittest

class TestBuiltins(unittest.TestCase):
    """Test some python built-in methods"""
    def test_len(self):
        self.assertEqual(5, len("hello"))
        self.assertEqual(3, len(['a','b','c']))
        # edge case
        self.assertEqual(0, len(""))

    def test_str_upper(self):
        self.assertTrue( "ABC".isupper() )
        self.assertFalse( "ABc".isupper() )
        s = ""  # edge case
        self.assertFalse( s.isupper() )
```

Running the tests

1. Let the IDE run them for you.
2. Use a separate test runner or build script.
3. Add a "main" script to end of Test class (as below)

```
import unittest

class TestBuiltins(unittest.TestCase):
    """Test some python built-in method"""
    def test_len(self):
        self.assertEqual(5, len("hello"))
        self.assertEqual(3, len(['a','b','c']))

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

Run from the good-old command line

Run all tests or just specific test.

```
>>> python -m unittest test_module1 test_module2

>>> python -m unittest module.TestClass

>>> python -m tests/test_module.py
```

More Interesting Example

- A Stack implements common stack data structure.
- You can push(), pop(), and peek() elements.
- Throws StackException if you do something stupid.

```
Stack<T>
--------------------------
+ Stack( capacity )
+ capacity( ): int
+ size( ): int
+ isEmpty( ): boolean
+ isFull( ): boolean
+ push( T ): void
+ pop( ): T
+ peek( ): T
```

Use setUp() to create test fixture

```
import unittest

class StackTest(unittest.TestCase):
    """Create a test fixture for the tests"""
    def setUp(self):
        self.capacity = 5
        self.stack = Stack(capacity)

    def test_newStackIsEmpty(self):
        self.assertTrue( self.stack.isEmpty() )
        self.assertFalse( self.stack.isFull() )
        self.assertEqual( 0, self.stack.size() )
```

Test for Exceptions

```
import unittest

class StackTest(unittest.TestCase):
    """Create a test fixture for the tests"""
    def setUp(self):
        self.capacity = 5
        self.stack = Stack(capacity)

    def test_popEmptyStack(self):
        """stack.pop() should throw exception"""
        with self.assertRaises(StackException):
            self.stack.pop()
```

Many ways to test for exception. Python Docs 27.4.8.1

doctest

Include runnable code in Python DocStrings.

```python
def add(a,b):
    """Compute the sum of two numbers.

    >>> add(3,4)
    7
    >>> add(0,99999)
    99999
    """
    return a+b

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

References

Python Official Docs (easy to read, has examples)

`https://docs.python.org/3/library/unittest.html`

Python Hitchhiker's Guide to Testing

`https://docs.python-guide.org/writing/tests/`

- Overview and examples of common test tools

Python Cookbook, Chapter 14

How to test many common situations, including I/O