Python
Unit testing framework

Stéphane Vialette

LIGM, Université Paris-Est Marne-la-Vallée

October 28, 2009

`unittest` – Unit testing framework

Description

- The Python unit testing framework (PyUnit) is a Python language version of JUnit.
- `unittest` supports:
  - test automation,
  - sharing of setup and shutdown code tests,
  - aggreagation of tests into collections, and
  - independence of the tests from the reporting framework.
- `unittest` provides classes that make it easy to support these qualifies for a set of tests.

`unittest` – Unit testing framework

Concepts

- **test fixture**: A test fixture represents the preparation needed to perform one or more tests, and any associate cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.
- **test case**: A test case is the smallest unit of testing. It checks for a specific response to a particular set of inputs. unittest provides a base class, TestCase, which may be used to create new test cases.
- **test suite**: A test suite is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.
- **test runner**: A test runner is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

Example

```
import random
import unittest

class TestSequenceFunctions(unittest.TestCase):

    def setUp(self):
        self.seq = range(10)

    def testshuffle(self):
        # make sure the shuffled sequence does not lose any elements
        random.shuffle(self.seq)
        self.seq.sort()
        self.assertEqual(self.seq, range(10))

    def testchoice(self):
        element = random.choice(self.seq)
        self.assert_(element in self.seq)

    def testsample(self):
        self.assertRaises(ValueError, random.sample, self.seq, 20)
        for element in random.sample(self.seq, 5):
            self.assert_(element in self.seq)

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

`unittest` – Unit testing framework

Description

- A testcase is created by subclassing `unittest.TestCase`. The three individual tests are defined with methods whose names start with the letters test. This naming convention informs the test runner about which methods represent tests.
- The crux of each test is a call to `assertEqual()` to check for an expected result, `assert_()` to verify a condition; or `assertRaises()` to verify that an expected exception gets raised.
- When a `setUp()` method is defined, the test runner will run that method prior to each test. Likewise, if a `tearDown()` method is defined, the test runner will invoke that method after each test.

`unittest` – Unit testing framework

Description

The final block shows a simple way to run the tests.
`unittest.main()` provides a command line interface to the test script.

Output

```
$ python testrandom.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
$
```

Other ways to run the tests

```
suite =
unittest.TestLoader().loadTestsFromTestCase(TestSequenceFunctions)
unittest.TextTestRunner(verbosity=2).run(suite)
```

Output

```
$ python testrandom2.py
testchoice (__main__.TestSequenceFunctions) ... ok
testsample (__main__.TestSequenceFunctions) ... ok
testshuffle (__main__.TestSequenceFunctions) ... ok
----------------------------------------------------------------------
Ran 3 tests in 0.001s
OK
$
```