Test Fixtures

Fixtures

As you write more tests, you'll create test case classes whose methods start or end with the same lines of code.

Consolidate these in `setUp()` and `tearDown()`.

```
from testutils import connect_to_test_database
class TestSample(unittest.TestCase):
    def setUp(self):
        "Run before every test method starts."
        self.conn = connect_to_test_database()
    def tearDown(self):
        "Run after every test method ends."
        self.conn.close()
    def test_create_tables(self):
        # Tests for creating DB tables
        # ...
    def test_stored_proc(self):
        # Tests exercising stored procecures
        # ...
```

Watch Out: It's "setUp", not "setup". "tearDown", not "teardown".

Example

Imagine writing a program that saves it state between runs. It saves it to a special file, called the "state file".

```
# statefile.py
class State:
    def __init__(self, state_file_path):
        # Load the stored state data, and save
        # it in self.data.
        self.data = { }
    def close(self):
        # Handle any changes on application exit.
```

Planning The Test

Tests on the `State` class should verify:

- If you add a new key-value pair to the state, it is recorded correctly in the state file.
- If you remove a key, that's recorded correctly, too.
- If you alter the value of an existing key, that updated value is written to the state file.
- If the state is not changed, the state file's content stays the same.

test_statefile: initial code

```
# test_statefile.py
import os
import unittest
import shutil
import tempfile
from statefile import State

INITIAL_STATE = '{"foo": 42, "bar": 17}'
```

test_statefile: setUp and tearDown

```
class TestState(unittest.TestCase):
    def setUp(self):
        self.testdir = tempfile.mkdtemp()
        self.state_file_path = os.path.join(
            self.testdir, 'statefile.json')
        with open(self.state_file_path, 'w') as outfile:
            outfile.write(INITIAL_STATE)
        self.state = State(self.state_file_path)

    def tearDown(self):
        shutil.rmtree(self.testdir)
```

test_statefile: the tests

```
    def test_change_value(self):
        self.state.data["foo"] = 21
        self.state.close()
        reloaded_statefile = State(self.state_file_path)
        self.assertEqual(21,
            reloaded_statefile.data["foo"])

    def test_remove_value(self):
        del self.state.data["bar"]
        self.state.close()
        reloaded_statefile = State(self.state_file_path)
        self.assertNotIn("bar", reloaded_statefile.data)

    def test_no_change(self):
        self.state.close()
        with open(self.state_file_path) as handle:
            checked_content = handle.read()
        self.assertEqual(checked_content, INITIAL_STATE)
```