Python Unit Testing

By MicroPyramid

Agenda

Unit Testing

Nose

Doc tests

Why Software testing is important?

To point out the defects during the development phases.

To Ensure that the application works as expected.

Test Driven Development.

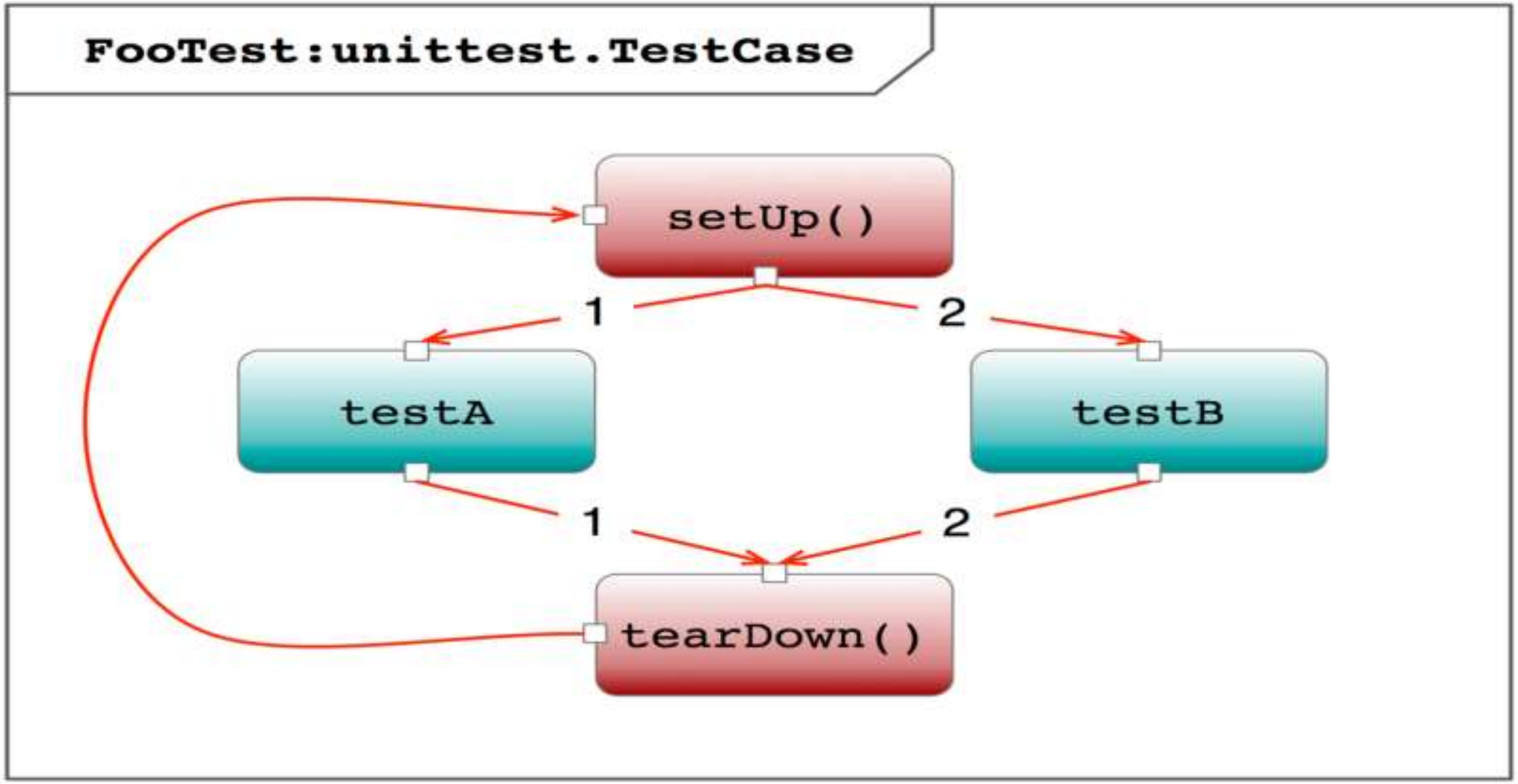
The process of implementing code by writing your tests first, seeing them fail, then writing the code to make the tests pass.

- Write Tests
- Make Them fail
- Write code.
- Make them pass
- Repeat

Python Unit test

Some Important Points

- Every test class must be sub class of **`unittest.TestCase`**
- Every test function should start with **test** name.
- to check for an expected result use **assert** functions.
- The **setUp()** method define instructions that will be executed before test case.
- The **tearDown()** method define instructions that will be executed after test case.
- Run Test with **python -m unittest -v test_module**
- Only test single part of code


FooTest:unittest.TestCase
setUp()
1
2
testA
testB
1
2
tearDown()

Let’s start

```
# tests/mul.py

def multiply(a, b):
    return a*b


def add(a, b):
    return a+b
```

Test case

```
import unittest
from mul import multiply


class MultiplyTestCase(unittest.TestCase):


    def test_multiplication_with_correct_values(self):
        self.assertEqual(multiply(5, 5), 25)


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

SetUp() and TearDown()

```python
class MulTestCase(unittest.TestCase):
    def setUp(self):   # Runs before every test method
        self.a = 10
        self.b = 20
    def test_mult_with_correct_values(self):
        self.assertEqual(multiply(self.a, self.b), 200)
    def tearDown(self): # runs after every test method
        del self.a
        del self.b

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

Assert functions

- assertEqual(a, b)
- assertNotEqual(a, b)
- assertTrue(x)
- assertFalse(x)
- assertIs(a, b)
- https://docs.python.org/2/library/unittest.html#test-cases

Nose:

```
$ pip install nose
# Running tests
$ nosetests
```

Doc Tests:

```
# tests/mul_dc.py
def multiply(a, b):
    """
    >>> multiply(4, 3)
    12
    """
    return a * b

# running
$ python -m doctest -v file_name
```

Questions?