Programming in Python

17 – Python 단위 테스트

2016년 8월, 국민대학교 컴퓨터공학부

단위 테스트의 중요성

- 소프트웨어 개발에 있어, 잠재적 문제는 빨리 발견할수록 적은 비용으로 대응할 수 있다.
- 소프트웨어 통합 (integration) 이전에 단위 테스트를 통하여 작은 단위의 검증을 선행하는 것이 중요

- Python 의 특수성
  - 스크립팅 언어에 가까움 - 컴파일 타임 에러가 없다!
  - 대화형 세션을 통하여 쉽게 작은 단위의 동작 확인이 가능하다.
  - 단위 테스트를 구현하는 것이 쉽다.

- 단위 테스트에서 주의를 기울일 점
  - 테스트 케이스는 미리 설계, 가능한 한 발생할 수 있는 모든 경우를 커버하도록
  - 자동으로 테스트할 수 있는 환경을 마련해 두고, 코드에 변경이 생기는 경우 항상 테스트 실행
    - CI 서버 등을 이용하는 것도 좋은 방안 (프로젝트 규모에 따라)
  - 코드 작성보다 테스트 케이스 작성을 먼저?

단위 테스트에서의 주요 개념

- Test fixture
  - 테스트를 수행하기 위한 준비와, 테스트가 끝난 후의 정리 작업
    - 예: 테스트를 위한 가상의 데이터베이스 및 그 안의 데이터 생성/파괴
- Test case
  - 테스트가 실행되는 (따라서 성공/실패하는) 가장 작은 단위
- Test suite
  - 테스트 케이스의 모음
- Test runner
  - 테스트 케이스를 실행하고 그 결과를 보여주고 집계하는 도구

Python 에서의 단위 테스트 프레임워크 - unittest

```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])

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

```
% python foo.py
...
----------------------------------
Ran 3 tests in 0.000s

OK
```

```
% python -m unittest -v foo
test_isupper (foo.TestStringMethods) ... ok
test_split (foo.TestStringMethods) ... ok
test_upper (foo.TestStringMethods) ... ok

----------------------------------------------------
Ran 3 tests in 0.000s

OK
```

테스트 코드의 작성

palindrome.py

```
class Palindrome:

    def __init__(self, s):
        self.x = s

    def normal(self):
        l = 0; r = len(self.x) - 1
        bPalin = False
        while l < r:
            if self.x[l] != self.x[r]:
                break
            l += 1; r -= 1
        else:
            bPalin = True
        return bPalin
```

testPalindrome.py

```
import unittest

from palindrome import Palindrome

class TestPalindrome(unittest.TestCase):

    def setUp(self):
        self.p1 = Palindrome('abcd')
        self.p2 = Palindrome('abcdedcba')

    def tearDown(self):
        pass

    def testNormal(self):
        self.assertFalse(self.p1.normal())
        self.assertTrue(self.p2.normal())

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

테스트의 실행

1. 스크립트를 실행

```
% python testPalindrome.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

2. 대화형 세션으로

```
>>> import unittest
>>> loader = unittest.TestLoader()
>>> suite = loader.discover('.')
>>> runner = unittest.TextTestRunner(verbosity=2)
>>> runner.run(suite)
testNormal (testPalindrome.TestPalindrome) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
<unittest.runner.TextTestResult run=1 errors=0 failures=0>
```

3. Discovery mode

```
% python -m unittest discover -v
testNormal (testPalindrome.TestPalindrome) ... ok

----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

2번과 3번의 경우에는
테스트 케이스를 정의한 .py 파일의 이름이
반드시 "test" 로 시작해야 함

테스트 케이스 추가

```
import unittest

from palindrome import Palindrome

class TestPalindrome(unittest.TestCase):

    def setUp(self):
        self.p1 = Palindrome('abcd')
        self.p2 = Palindrome('abcdedcba')

    def tearDown(self):
        pass

    def testNormal(self):
        self.assertFalse(self.p1.normal())
        self.assertTrue(self.p2.normal())

    def testRecursive(self):
        self.assertFalse(self.p1.recursive())
        self.assertTrue(self.p2.recursive())
```

```
% python -m unittest discover -v
testNormal (testPalindrome.TestPalindrome) ... ok
testRecursive (testPalindrome.TestPalindrome) ... ERROR

======================================================================
ERROR: testRecursive (testPalindrome.TestPalindrome)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/…/testPalindrome.py", line 19, in testRecursive
    self.assertFalse(self.p1.recursive())
AttributeError: Palindrome instance has no attribute 'recursive'

----------------------------------------------------------------------
Ran 2 tests in 0.000s

FAILED (errors=1)
```

새로운 메서드의 구현

```
class Palindrome:

    def __init__(self, s):
        self.x = s

    def normal(self):
        …

    def recursive(self):
        return self.r(self.x)

    def r(self, s):
        return s[0] == s[-1] and self.r(s[1:-1])
```

무엇이 잘못되었을까?
이 잘못은 어떻게 찾아내야 할까?

```
 python -m unittest discover -v
testNormal (testPalindrome.TestPalindrome) … ok
testRecursive (testPalindrome.TestPalindrome) … ERROR

======================================================================
ERROR: testRecursive (testPalindrome.TestPalindrome)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/…/testPalindrome.py", line 20, in testRecursive
    self.assertTrue(self.p2.recursive())
  File "/…/palindrome.py", line 18, in recursive

    …

IndexError: string index out of range

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (errors=1)
```

메서드 구현의 수정

```
def recursive(self):
        return self.r(self.x)

def r(self, s):
        if len(s) == 0 or len(s) == 1:
                return True
        return s[0] == s[-1] and self.r(s[1:-1])
```

```
% python -m unittest discover -v
testNormal (testPalindrome.TestPalindrome) ... ok
testRecursive (testPalindrome.TestPalindrome) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```

또다른 테스트 케이스의 추가

```
class TestPalindrome(unittest.TestCase):

    def setUp(self):
        self.p1 = Palindrome('abcd')
        self.p2 = Palindrome('abcdedcba')
        self.p3 = Palindrome('ab c d cba')

    def tearDown(self):
        pass

    def testNormal(self):
        self.assertFalse(self.p1.normal())
        self.assertTrue(self.p2.normal())
        self.assertFalse(self.p3.normal())

    def testRecursive(self):
        self.assertFalse(self.p1.recursive())
        self.assertTrue(self.p2.recursive())
        self.assertFalse(self.p3.recursive())

    def testIgnoreSpaces(self):
        self.assertFalse(self.p1.ignoreSpaces())
        self.assertTrue(self.p2.ignoreSpaces())
        self.assertTrue(self.p3.ignoreSpaces())
```

사실은 p3 와 같은 경우에 대해서도
이전의 두 테스트 케이스, 즉
normal() 과 recursive() 에 대해서도
미리 고려하였다면
보다 나은 테스트 케이스였을 것

메서드 추가 구현, 테스트 실행

```
class Palindrome:

    def __init__(self, s):
        self.x = s

    def normal(self):
        ...

    def recursive(self):
        ...

    def r(self, s):
        ...

    def ignoreSpaces(self):
        s = ''.join(self.x.split())
        return self.r(s)
```

```
python -m unittest discover -v
testIgnoreSpaces (testPalindrome.TestPalindrome) ... ok
testNormal (testPalindrome.TestPalindrome) ... ok
testRecursive (testPalindrome.TestPalindrome) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

Assert 의 종류와 문법

Method	Checks that	New in
`assertEqual(a, b)`	`a == b`	
`assertNotEqual(a, b)`	`a != b`	
`assertTrue(x)`	`bool(x) is True`	
`assertFalse(x)`	`bool(x) is False`	
`assertIs(a, b)`	`a is b`	2.7
`assertIsNot(a, b)`	`a is not b`	2.7
`assertIsNone(x)`	`x is None`	2.7
`assertIsNotNone(x)`	`x is not None`	2.7
`assertIn(a, b)`	`a in b`	2.7
`assertNotIn(a, b)`	`a not in b`	2.7
`assertIsInstance(a, b)`	`isinstance(a, b)`	2.7
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`	2.7

예: assertEqual(first, second, msg=None)

Test Fixture

```
import unittest

class WidgetTestCase(unittest.TestCase):

    def setUp(self):
        self.widget = Widget('The widget')

    def tearDown(self):
        self.widget.dispose()
        self.widget = None

    def test_default_size(self):
        self.assertEqual(self.widget.size(), (50, 50),
                         'incorrect default size')

    def test_resize(self):
        self.widget.resize(100, 150)
        self.assertEqual(self.widget.size(), (100, 150),
                         'wrong size after resize')
```

예제

```
class TestPalindrome(unittest.TestCase):

    def setUp(self):
        ...
        self.p4 = Palindrome("Madam, I'm Adam.")

    def tearDown(self):
        pass

    def testNormal(self):
        ...
        self.assertFalse(self.p4.normal())

    def testRecursive(self):
        ...
        self.assertFalse(self.p4.recursive())

    def testIgnoreSpaces(self):
        ...
        self.assertFalse(self.p4.ignoreSpaces())

    def testIgnoreSpecials(self):
        self.assertTrue(False, 'not yet implemented')
```

```
% python testPalindrome.py
.F..
======================================================================
FAIL: testIgnoreSpecials (__main__.TestPalindrome)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "testPalindrome.py", line 35, in testIgnoreSpecials
    self.assertTrue(False, 'not yet implemented')
AssertionError: not yet implemented

----------------------------------------------------------------------
Ran 4 tests in 0.001s

FAILED (failures=1)
```

Quiz

- 단위 테스트란 무엇인가?
  - 무엇을 테스트해야 하는가?
- Python 에서 단위 테스트를 편리하게 해주기 위한 도구에는 어떤 것이 있는가?
- unittest 에서 이용할 수 있는 assert 의 종류에는 어떤 것들이 있는가?
- 앞에서 예제로 사용한 testPalindrome 에는 중요한 경우가 빠져 있다. 무엇일까?
- Test fixture 란 무엇이며, 어디에 이용하는가?

Exercise

(1) 앞의 예제에서 ignoreSpecials() 메서드를 완성해 보자.

(2) 여러 모듈로 이루어진 테스트 케이스를 실험해 보자.

testA.py

```
import unittest

class TestOne(unittest.TestCase):

    def testOne(self):
        print 'test one'
```

testB.py

```
import unittest

class TestTwo(unittest.TestCase):

    def testTwo(self):
        print 'test two'
```

testC.py

```
import unittest

class TestThree(unittest.TestCase):

    def testThree(self):
        print 'test three'
```

파일의 배치:

./testA.py
./testB.py
./sub/testC.py

```
% python -m unittest discover
testThree (sub.testC.TestThree) ... test three
ok
testOne (testA.TestOne) ... test one
ok
testTwo (testB.TestTwo) ... test two
ok

----------------------------------------------------------------------
Ran 3 tests in 0.000s

OK
```

예상대로 동작하나요?
아닌가요?
아니라면, 왜 아닐까요?

Q & A