Юнит-тестирование. unittest

- http://pycheesecake.org/wiki/PythonTestingToolsTaxonomy
- assert condition
- unittest
- Фильтрация тестов, автопоиск, etc
- assertEqual, assertIn, assertRaises, assertAlmostEqual, assertDictContainsSubset
- python -m unittest test_module1 test_module2
- python -m unittest test_module.TestClass
- python -m unittest test_module.TestClass.test_method
- python -m unittest discover project_directory '*_test.py'

Юнит-тестирование. unittest

```
1   import unittest
2
3   class DefaultWidgetSizeTestCase(unittest.TestCase):
4       def setUp(self):
5           self.widget = Widget('The␣widget')
6
7       def runTest(self):
8           assert self.widget.size() == (50, 50), "incorrect␣de
9           self.assertEqual(self.widget.size(), \
10              (50, 50), "incorrect␣default␣size")
11
12      def tearDown(self):
13          self.widget.dispose()
14          self.widget = None
15
16  if __name__ == '__main__':
17      unittest.main()
```

Юнит-тестирование. nose

- Автопоиск и исполнение тестов, плагины (coverage, исполнение тестов в отдельных процессах)
- Юнит тесты в функциях

```
1       from nose.tools import eq_
2
3       # test_some.py
4       def test1():
5           "test, that 1 == 2"
6           assert 1 == 2
7           eq_(1, 2)
```

Юнит-тестирование. oktest

- py.test (pytest)
- oktest

```
1       from oktest import ok
2
3       def test_func():
4           ok (s) == 'foo'
5           ok (s) != 'foo'
6           ok (n) > 0
7           ok (fn).raises(Error)
8           ok ([]).is_a(list)
```

Юнит-тестирование. mock

```
1       import mock
2
3       mock = mock.Mock()
4       mock.method(1, 2, 3, test='wow')
5       mock.called == True
6       mock.method.assert_called_with(1, 2, 3, test='wow')
7
8       attrs = {'method.return_value': 3, \
9                'other.side_effect': KeyError}
10      mock.configure_mock(**attrs)
11      mock.method() == 3
12      mock.other() # KeyError raised
```

Юнит-тестирование. mock

```
1   from mock import patch
2
3   class Class(object):
4       def method(self):
5           pass
6
7   with patch('__main__.Class') as MockClass:
8       instance = MockClass.return_value
9       instance.method.return_value = 'foo'
10      assert Class() is instance
11      assert Class().method() == 'foo'
12
13  @patch.object(SomeClass, 'class_method')
14  def test(mock_method):
15      SomeClass.class_method(3)
16      mock_method.assert_called_with(3)
17
18  test()
```

Юнит-тестирование. ludibrio

```
1   from ludibrio import Mock
2
3   with Mock() as MySQLdb:
4       con = MySQLdb.connect('server', 'user', 'XXXX')
5       con.select_db('DB') >> None
6       cursor = con.cursor()
7       cursor.execute('select␣*␣from␣numbers') >> None
8       cursor.fetchall() >> [1,2,3,4,5]
9
10  con = MySQLdb.connect('server', 'user', 'XXXX')
11  con.select_db('DB')
12  cursor = con.cursor()
13  cursor.execute('select␣*␣from␣numbers')
14  cursor.fetchall() == [1, 2, 3, 4, 5]
15
16  MySQLdb.validate()
```