

Python kodo testavimas

Albertas Gimbutas

¹Matematikos ir informatikos institutas
Vilniaus universitetas

2014 pavasaris

- 1 Įvadas į testų rašymą
- 2 Modulių testai
- 3 Įmitavimas
- 4 Tinklapių testavimas

- **TDD** (Test Driven Development) - reikalauja pastangų ir laiko. Naudojamas kai reikia ypatingos kokybės, pavyzdžiui taisant sistemos klaidas
- Skiriant 25-50% laiko pasiekiamas 60-70% testų padengimas
- **Modulių testai** (angl. unit tests) - testuoja logines dalis
 - **Doktestai** (angl. doctests) - testuoja logines dalis, bet rašomi aprašomajame tekste (angl. docstrings).
- **Integraciniai testai** (angl. integration tests) - testuoja bendrą produkto veikimą

```
from random import seed, shuffle, sample
from unittest import TestCase, main

class SeedAndShuffleTestCase(TestCase):
    def setUp(self):
        '''Vykdomas prieš kiekvieną testą.'''
        self.sample = list('123abc')
        seed('qIUdma')

    def test_shuffle(self): # Testai prasideda "test_"
        shuffle(self.sample)
        self.result = ''.join(self.sample)
        self.assertEqual(self.result, 'a2b3c1')

    def test_random(self):
        # self.result - nebeegzistuoja
        self.assertEqual(''.join(self.sample), '123abc')

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

Patikrinimų pavyzdžiai

```
assertEqual(a, b)
assertNotEqual(a, b)
assertTrue(x)
assertFalse(x)
assertIs(a, b)
assertIsNot(a, b)
assertIsNone(x)
assertIn(a, b)
assertNotIn(a, b)
assertIsInstance(a, b)
assertNotIsInstance(a, b)
assertRaises(e)
```

```
a == b
a != b
bool(x) is True
bool(x) is False
a is b
a is not b
x is None
a in b
a not in b
isinstance(a, b)
not isinstance(a, b)
Exception e is raised
```

```
>>> with self.assertRaises(SomeException):
...     do_something()    # should raise SomeException
```

Testų vykdymas

```
# Testų esančių faile vykdymas
```

```
$ python tests.py
```

```
# Testų rinkinio vykdymas
```

```
$ python tests.py SeedAndShuffleTestCase
```


```
# Vieno testo vykdymas
```

```
$ python tests.py SeedAndShuffleTestCase.test_shuffle
```

```
# Projekto testų vykdymas
```

```
$ make test
```

- python.org: **unittest**

-  **Jenkins** - įrankis automatizuotam testų vykdymui. Testai įvykdomi po kiekvieno kodo patalpinimo į saugyklą. Apie lūžtančius testus informuojama (el. paštu/IRC). Vizualizuojama testų vykdymo statistika, istorija.

Objektų ir metodų įmitavimas su Mock

```
>>> from unittest.mock import Mock      # Nuo 3.3 versijos
>>> ai = Mock()
>>> ai.dream('Nature')    # Kviečiame išgalvotą metodą
<Mock name='mock.dream()' id='140389589616464'>
>>> ai.dream.call_count
1
>>> ai.dream.return_value = 'Trees, rivers, sun...'
>>> ai.dream('Nature')
'Trees, rivers, sun...'
>>> ai.dream.assert_called_with('Nature')
>>> ai.dream.reset_mock()
>>> ai.dream.call_count
0
>>> ai.dream.assert_called_with('Nature')
AssertionError: Expected call: dream('Nature')
```

- python.org: unittest.mock

Duomenų bazės įrašai testavimui

- Testavimo duomenų bazė užpildoma pavyzdiniais objektais. Testui reikalingi objektai taip pat gali būti sukuriami tiek `TestCase.setUp()`, tiek testo blokuose.

```
from django.test import TestCase

class MethodAnalytesWidgetTestCase(TestCase):
    fixtures = ['development.json']
    ...
```

fixtures/development.json

```
[
    {
        "pk": 1,
        "model": "mymodule.user",
        "fields": {
            "username": "root",
        }, ...
    ]
```


HTTP protokolo ir HTML kalbos apžvalga

- Internetas yra grįstas HTTP/HTTPS protokolu, kuris skirtas keisti informacija tarp naršyklės ir serverio. Apsikeitimas informacija dažniausiai vyksta GET ir POST užklausomis.
- GET užklausos metu nurodytam URL adresui serveris grąžina HTML dokumentą, kuris yra pavaizduojamas naršyklėje.
- HTML dokumentas gali turėti form žymę, kuri turi:
 - atributą `action`, kuris nurodo URL į kurį bus siunčiami duomenys
 - atributą `method` - HTTP metodui nurodyti, pvz.: POST
 - vaikines duomenų įvedimo žymes, pvz.: `input`, `checkbox`, `textarea`. Išsiuntus formą šių žymių reikšmės yra perduodamos serveriui.
 - Išsiuntimo veiksmą inicijuojantį mygtuką.

HTTP protokolo ir HTML kalbos apžvalga

- HTML žymėms galima nurodyti `id` arba `class` atributų reikšmes, kad jas būtų lengviau identifikuoti.

```
<html>
  <form id="login-form" action="." method="post">
    <input name='username' type='text' />
    <input name='password' type='password' />
    <button>Prisijungti</button>
  </form>
</html>
```

- Šią formą galima pasirinkti su CSS selector išraiška:
`#login-form`

Django virtuali naršyklė

```
>>> from django.test import Client
>>> c = Client()
>>> response = c.post('/login/', {
...     'username': 'vardenis',
...     'password': 'slaptažodis'
... })
>>> response.status_code
200
>>> response.content
'<html>...'
>>> c.login(username='vardenis', password='slaptažodis')
>>> c.logout()
```

- djangoproject.com: testing tools

Django puslapių testavimas

- 1 Atidaromas testuojamas puslapis su virtualia naršykle
- 2 Pasirenkama forma, ji užpildoma duomenimis ir išsiunčiama
- 3 Patikrinamas virtualios naršyklės gautasis atsakymas iš serverio

```
from django.core.urlresolvers import reverse
from hack4lt.test_utils import StatefulTesting

class ProfileTestCase(StatefulTesting):
    def test_login(self):
        self.open(reverse('login'))
        self.selectForm('#login-form')
        self.submitForm({
            'username': 'vardenis',
            'password': 'mano_slaptažodis',
        })
        self.assertStatusCode(302)
        self.selectTable('#profile-table')
        self.assertTableHasRows(u'Vardenis')
```