Programavimo kalba

Python

penktoji paskaita

Marius Gedminas
<mgedmin@b4net.lt>

http://mg.b4net.lt/python/

Matematikos ir Informatikos fakultetas

2006 ruduo

Programas reikia testuoti

Programą pakeitus reikia ją testuoti iš naujo

Rankomis kartoti tuos pačius testus sunku ir nuobodu

Automatizuotas testavimas

```
$ python test.py
Ran 120 tests in 3 seconds.

OK
```

Kokie būna testai?

Modulių testai

Sistemos testai

Modulių testai:

kiekviena funkcija testuojama atskirai

Sistemos testai:

testuojama visa programa

import unittest

Funkcija

```
# fact.py

def fact(n):
    f = 1
    for i in range(n):
        f *= i
    return f
```

Funkcijos testai

```
class TestFact(unittest.TestCase):
    def test(self):
        self.assertEquals(fact(0), 1)
        self.assertEquals(fact(1), 1)
        self.assertEquals(fact(2), 2)
        self.assertEquals(fact(3), 6)
        self.assertEquals(fact(4), 24)
```

Testų rinkinys

```
import unittest
from fact import fact

class TestFact(unittest.TestCase):
    ...

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

```
$ python test.py
F
======================================================================
FAIL: test (__main__.TestFact)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 8, in test
    self.assertEquals(fact(1), 1)
AssertionError: 0 != 1

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
```

```
# fact.py

def fact(n):
    f = 1
    for i in range(1, n):
        f *= i
    return f
```

```
$ python test.py
F
======================================================================
FAIL: test (__main__.TestFact)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 9, in test
    self.assertEquals(fact(2), 2)
AssertionError: 1 != 2

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
```

```
# fact.py

def fact(n):
    f = 1
    for i in range(1, n + 1):
        f *= i
    return f
```

```
$ python test.py
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

Testai pirma

(Test Driven Development)

1. Rašai naują testą
2. Leidi testų rinkini
   (naujas testas nepraeina)
3. Rašai kodą, kad testas veiktų
4. Leidi testų rinkinį
   (testas praeina)
5. Kartoji

TDD nauda:

testų rinkinys yra pilnas

TDD nauda:

kodas bus toks, kokį lengva naudoti

Realesnis pavyzdys sprendžiam kvadratines lygtis

1. Įvedimas

2. Sprendimas

3. Išvedimas

2. Sprendimas

```
# qeq.py

def solve(a, b, c):
    """Solve ax**2 + bx + c = 0.
    Returns a list of solutions.
    """
    raise NotImplementedError
```

```
# test.py

class TestSolve(unittest.TestCase):
    def test_no_solutions(self):
        # x**2 + 4 = 0
        self.assertEquals(
            solve(1, 0, 4), [])
```

```
$ python test.py

======================================================================
ERROR: test_no_solutions (__main__.TestSolve)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 10, in test_no_solutions
    self.assertEquals(
  File "qeq.py", line 8, in solve
    raise NotImplementedError
NotImplementedError


FAILED (errors=1)
```

```
# qeq.py

def solve(a, b, c):
    """Solve ax**2 + bx + c = 0.
    Returns a list of solutions.
    """
    return []
```

```
$ python test.py

...

Ran 1 test in 0.001s

OK
```

```
# test.py

    def test_two_solutions(self):
        # (x - 3)(x + 2) = 0
        # x**2 - x - 6 = 0
        self.assertEquals(
            solve(1, -1, -6),
            [-2, 3])
```

```
$ python test.py

======================================================================
ERROR: test_two_solutions (__main__.TestSolve)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test2.py", line 18, in test_two_solutions
    [-2, 3])
AssertionError: [] != [-2, 3]


FAILED (failures=1)
```

```
# qeq.py
from math import sqrt
def solve(a, b, c):
    d = b ** 2 - 4 * a * c
    x1 = (-b - sqrt(d)) / (2 * a)
    x2 = (-b + sqrt(d)) / (2 * a)
    return [x1, x2]
```

```
$ python test.py
======================================================================
ERROR: test_no_solutions (__main__.TestSolve)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test.py", line 10, in test_no_solutions
    self.assertEquals(
  File "qeq.py", line 11, in solve
    x1 = (-b - sqrt(d)) / (2 * a)
ValueError: math domain error

Ran 2 tests in 0.001s

FAILED (errors=1)
```

```
# qeq.py
from math import sqrt
def solve(a, b, c):
    d = b ** 2 - 4 * a * c
    if d < 0: return []
    x1 = (-b - sqrt(d)) / (2 * a)
    x2 = (-b + sqrt(d)) / (2 * a)
    return [x1, x2]
```

```
$ python test.py

...

Ran 2 tests in 0.001s

OK
```

```
# test.py

    def test_one_solution(self):
        # (x - 5)**2 = 0
        # x**2 - 10x + 25 = 0
        self.assertEquals(
            solve(1, -10, 25),
            [5])
```

```
$ python test.py
======================================================================
ERROR: test_one_solution (__main__.TestSolve)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test2.py", line 25, in test_one_solution
    [5])
AssertionError: [5.0, 5.0] != [5]

Ran 3 tests in 0.001s

FAILED (failures=1)
```

```
# qeq.py
def solve(a, b, c):
    d = b ** 2 - 4 * a * c
    if d < 0: return []
    x1 = (-b - sqrt(d)) / (2 * a)
    if d == 0: return [x1]
    x2 = (-b + sqrt(d)) / (2 * a)
    return [x1, x2]
```

```
$ python test.py

...

Ran 3 tests in 0.001s

OK
```

1. Įvedimas

```
# test.py

class TestInput(unittest.TestCase):
    sampleinput = [
      '3, 4, 5',
      '1, -3, 22',
      '0.5, -.16, 42.3'
    ]
```

```
# test.py

    def test_input(self):
        self.assertEquals(
        readInput(self.sampleinput),
        [(3, 4, 5),
         (1, -3, 22),
         (.5, -0.16, 42.3)])
```

```
# qeq.py

def readInput(f):
    """Read input from a file object.

    Each line is of the form
      a, b, c

    Returns a list of (a, b, c) tuples.
    """
    raise NotImplementedError
```

```
$ python test.py
======================================================================
ERROR: test_input (__main__.TestInput)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "test2.py", line 37, in test_input
    self.assertEquals(
  File "qeq.py", line 28, in readInput
    raise NotImplementedError
NotImplementedError

Ran 4 tests in 0.001s

FAILED (errors=1)
```

```
# qeq.py

import csv
def readInput(f):
    results = []
    for a, b, c in csv.reader(f):
        results.append((float(a),
float(b), float(c)))
    return results
```

```
$ python test.py
.
----------------------------------------------------------------------
Ran 4 tests in 0.001s

OK
```

3. Išvedimas

Praktiškai tas pats.

StringIO modulis naudingas!

Jei liko laiko: daugiau pavyzdžių