Praktyczne Aspekty Wytwarzania Oprogramowania

PYTHON
PYTHON
PYTHON

Maciej Chmiel <maciej.chmiel@nsn.com>

Maciej Ogrodniczek <maciej.ogrodniczek@nsn.com>

Nokia Siemens Networks

Importowanie modułów - przypomnienie

```
[ogrodnic@wain sandbox2]$ ls
pola.py
[ogrodnic@wain sandbox2]$ tree
.
`-- pola.py

0 directories, 1 file
```

```
>>> import pola
>>> pola.pole_kwadratu(4)
16
```

```
>>> from pola import pole_kwadratu
>>> pole_kwadratu(5)
25
```


Nokia Siemens Networks

Importowanie modułów - PYTHONPATH

```
[ogrodnic@wain subfolder]$ tree ..
..
|-- pola.py
|-- pola.pyc
`-- subfolder

1 directory, 2 files
[ogrodnic@wain subfolder]$ python

>>> import pola
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named pola
>>>
```

Importowanie modułów - PYTHONPATH

```
[ogrodnic@wain subfolder]$ echo $PYTHONPATH

[ogrodnic@wain subfolder]$ export PYTHONPATH='../'
[ogrodnic@wain subfolder]$ echo $PYTHONPATH
../
[ogrodnic@wain subfolder]$ python
Python 2.7.5 (default, Feb 19 2014, 13:47:28)
[GCC 4.8.2 20131212 (Red Hat 4.8.2-7)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import pola
>>>
```

Importowanie modułów – sys.path

```
[ogrodnic@wain subfolder]$ python
Python 2.7.5 (default, Feb 19 2014, 13:47:28)
[GCC 4.8.2 20131212 (Red Hat 4.8.2-7)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.path.append('../')
>>> import pola
>>>
```


Nokia Siemens Networks

Importowanie modułów - __init__.py

```
[ogrodnic@wain sandbox]$ tree
.
`-- pola
    |-- __init__.py
    |-- kwadrat.py
    |-- nie_do_importu_by_default.py
    `-- trojkat.py

1 directory, 4 files
```

```
[ogrodnic@wain sandbox]$ cat pola/__init__.py
__all__ = ["kwadrat", "trojkat"]
```


Nokia Siemens Networks

Importowanie modułów - __init__.py

```
>>> from pola import *

>>> import sys
>>> 'kwadrat' in sys.modules.keys()
False
>>> 'pola.kwadrat' in sys.modules.keys()
True
>>> 'pola.trojkat' in sys.modules.keys()
True
>>> 'pola.nie_do_importu_by_default' in sys.modules.keys()
False


>>> trojkat.pole(4,5)
10.0
```

Importowanie modułów - __init__.py

```
>>> from pola import *

>>> import sys
>>> 'kwadrat' in sys.modules.keys()
False
>>> 'pola.kwadrat' in sys.modules.keys()
True
>>> 'pola.trojkat' in sys.modules.keys()
True
>>> 'pola.nie_do_importu_by_default' in sys.modules.keys()
False


>>> trojkat.pole(4,5)
10.0
```

Importowanie modułów - __init__.py

```
1 import sys
2 import os
3 sys.path.insert(1, os.path.abspath(os.path.dirname(__file__)+'/../inny_modul/'))
4
5 __all__ = ["kwadrat", "trojkat"]
```

```
[ogrodnic@wain sandbox]$ tree
.
|-- inny_modul
|   `-- plik.py
`-- pola
    |-- __init__.py
    |-- kwadrat.py
    |-- nie_do_importu_by_default.py
    `-- trojkat.py

2 directories, 5 files
[ogrodnic@wain sandbox]$
```


Nokia Siemens Networks

Importowanie modułów - __init__.py

```
[ogrodnic@wain sandbox]$ cat inny_modul/plik.py
print('Inny modul: plik, zglasza sie')
[ogrodnic@wain sandbox]$ python
Python 2.7.5 (default, Feb 19 2014, 13:47:28)
[GCC 4.8.2 20131212 (Red Hat 4.8.2-7)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from pola import *
Inny modul: plik, zglasza sie
>>>
```

```
[ogrodnic@wain sandbox]$ cat pola/kwadrat.py
import plik
```

Testowanie

“...jeden z procesów zapewniania jakości oprogramowania”

“... testowanie ma na celu weryfikację oraz walidację oprogramownia”

Weryfikacja - sprawdzenie czy oprogramowanie jest zgodne ze specyfikacją

Walidacja - sprawdzenie czy oprogramowanie jest zgodne z oczekiwaniami użytkownika

Nokia Siemens Networks

Testowanie

Należy pamiętać że:

Trudno stwierdzić czy błąd jest błędem.

Programu nie da sie przetestować całkowicie.

Test nie udowodni braku błędów.

Nokia Siemens Networks

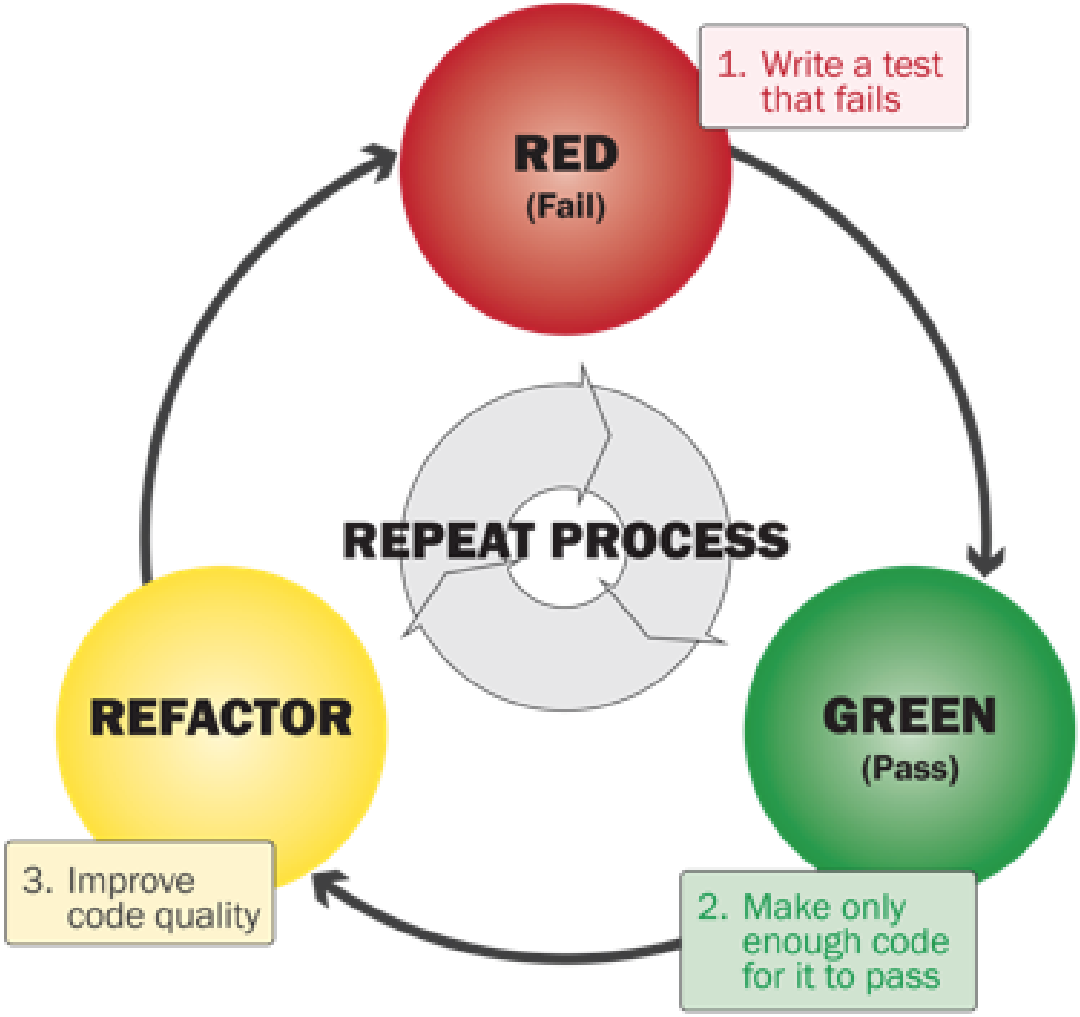
Testowanie – czym jest błąd?

Oprogramowanie nie robi czegoś co zostało wymienione w specyfikacji.

Oprogramowanie wykonuje coś czego według specyfikacji nie powinno być.

Nokia Siemens Networks

Testowanie – TDD



Nokia Siemens Networks

Testowanie – Testy jednostkowe

- testowanie pojedynczego elementu (jednostki) programu np. funkcji, klasy, modułu

- mogą składać się z kilku innych “mniejszych” testów

- są formą specyfikacji produktu

Testowanie – Testy jednostkowe

Sposób testowania:

- sprawdzanie czy dla przykładowych danych sa zwracane właściwe wyniki - porównywanie uzyskanych wyników z wynikami wzorcowymi (oczekiwanymi)

- sprawdzenie reakcji na błędne dane: rzucenie wyjątku

środowiska xUnit: np. jUnit

Testy implementowane są jako metody

Unittest

Najpopularniejszy framework: unittest (od wersji Python 2.1 standardowo dołączany)

Dokumentacja: http://docs.python.org/2/library/unittest.html

Framework obsługujący testy jednostkowe

Pythonowa wersja JUnit (Java)

Bogaty zestaw funkcji (API)

Unittest

```
 1 def pole_kwadratu(a):
 2     """
 3     Zwraca pole kwadratu.
 4     """
 5     return a**2
 6
 7
 8 def pole_trojkata(a, h):
 9     """
10     Zwraca pole trojkata.
11     """
12     return  a*h/2.0
13
~
```


Nokia Siemens Networks

Unittest

```
 1 import unittest
 2 import pola
 3
 4 class PolaTest(unittest.TestCase):
 5     def test_proste_pole_kwadratu(self):
 6         wynik = pola.pole_kwadratu(2)
 7         oczekiwany_wynik = 4
 8         self.assertEqual(wynik, oczekiwany_wynik)
 9
10 if __name__ == "__main__":
11     unittest.main()
```

Unittest - asserts

`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`
`assertIsNot(a, b)`	`a is not b`
`assertIsNone(x)`	`x is None`
`assertIsNotNone(x)`	`x is not None`
`assertIn(a, b)`	`a in b`
`assertNotIn(a, b)`	`a not in b`
`assertIsInstance(a, b)`	`isinstance(a, b)`
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`

`assertAlmostEqual(a, b)`	`round(a-b, 7) == 0`
`assertNotAlmostEqual(a, b)`	`round(a-b, 7) != 0`
`assertGreater(a, b)`	`a > b`
`assertGreaterEqual(a, b)`	`a >= b`
`assertLess(a, b)`	`a < b`
`assertLessEqual(a, b)`	`a <= b`
`assertRegexpMatches(s, re)`	`regex.search(s)`
`assertNotRegexpMatches(s, re)`	`not regex.search(s)`
`assertItemsEqual(a, b)`	sorted(a) == sorted(b) and works with unhashable objs
`assertDictContainsSubset(a, b)`	all the key/value pairs in *a* exist in *b*

`assertRaises(exc, fun, *args, **kwds)`	`fun(*args, **kwds)` raises *exc*
`assertRaisesRegexp(exc, re, fun, *args, **kwds)`	`fun(*args, **kwds)` raises *exc* and the message matches *re*

Nokia Siemens Networks

Unittest

```
 9      def test_zle_pole_kwadratu(self):
10          wynik = pola.pole_kwadratu(-2)
11          oczekiwany_wynik = '?'
12          self.assertEqual(wynik, oczekiwany_wynik)
```


Nokia Siemens Networks

Unittest - mock

```
 1 def pole_kwadratu(a):
 2     """
 3     Zwraca pole kwadratu.
 4     """
 5     niebezpieczna_funkcja(a)
 6     return a**2
 7
 8 def niebezpieczna_funkcja(bok):
 9     for iter in range(bok):
10         print bok
11
```


Nokia Siemens Networks

Unittest - mock

```
 1 import unittest
 2 import pola
 3 from mock import MagicMock
 4
 5 class PolaTest(unittest.TestCase):
 6     def test_proste_pole_kwadratu(self):
 7         pola.niebezpieczna_funkcja = MagicMock(return_value=None)
 8         wynik = pola.pole_kwadratu(20)
 9         oczekiwany_wynik = 400
10         self.assertEqual(wynik, oczekiwany_wynik)
11
12
13
14
15 if __name__ == "__main__":
16     unittest.main()
~
```


Nokia Siemens Networks

Unittest - nosetests

```
(pywork2)[ogrodnic@wain sandbox_mock]$ nosetests --cover-package=pola --cover-html --with-coverage
.
Name    Stmts   Miss  Cover   Missing
---------------------------------------
pola        6      2    67%   9-10
----------------------------------------------------------------------
Ran 1 test in 0.007s

OK
(pywork2)[ogrodnic@wain sandbox_mock]$
```


Nokia Siemens Networks

Unittest – nosetests + coverage

Coverage for **pola** : 67%

6 statements | 4 run | 2 missing | 0 excluded

```
1  def pole_kwadratu(a):
2      """
3      Zwraca pole kwadratu.
4      """
5      niebezpieczna_funkcja(a)
6      return a**2
7
8  def niebezpieczna_funkcja(bok):
9      for iter in range(bok):
10         print bok
11
```

« index coverage.py v3.7.1

Nokia Siemens Networks

Pytania?

Nokia Siemens Networks