Python2018

compscicenter.ru

aleksey.kladov@gmail.com

Лекция 11: Тестирование

Терминологическая Путаница

unit-testing

тестирование изолированных компонент / автоматизированное тестирование

integration-testing

тестирование взаимодействия компонент / не unit тестирование

Свойства Тестов

- Простота написания (!)
- Скорость работы (unit vs. integration)
- Стабильность при изменении кода (code churn)
- Стабильность при изменении внешних компонент
- Надёжность (тесты проходят, но работает ли программа?)

Личный Опыт

- Тестирование это сложно и не понятно
- Лучшая метрика — покрытие user-visble фич
- Лучшие тесты — быстрые интеграционные тесты без внешних зависимостей
- Нужны медленные flacky тесты для зависимостей
- Data-driven тесты эффективны
- Не умею тестировать асинхронный код (в том числе UI)

System Under Test

```
import itertools

def rle(iterable):
    """Apply run-length encoding to an iterable."""
    for item, g in itertools.groupby(iterable):
        yield item, sum(1 for _ in g)
```

Тестирование `print`

```
def rle(iterable):
    ...

print(list(rle("mississippi")))
```

Тестирование `print`

```
def rle(iterable):
    ...

print(list(rle("mississippi")))
```

:-(

Тестирование `assert`

```
def rle(iterable):
    ...

assert list(rle("")) == []
assert list(rle("foo")) == [('f', 1), ('o', 2)]
```

- Не используйте в production :)
- Сильно лучше `print` для “write-once” скриптов
- Лучше, чем ничего/`print`, для домашек по алгоритмам
- Исполняется **каждый** раз

Сообщение `assert`

```
def rle(iterable):
    return []

assert list(rle("foo")) == [('f', 1), ('o', 2)]
```

```
Traceback (most recent call last):
  File "rle.py", line 4, in <module>
    assert list(rle("foo")) == [('f', 1), ('o', 2)]
AssertionError
```

Сообщение `assert`

```
def rle(iterable):
    return []

assert list(rle("foo")) == [('f', 1), ('o', 2)],  list(rle("foo"))
```

```
Traceback (most recent call last):
  File "rle.py", line 4, in <module>
    assert list(rle("foo")) == [('f', 1), ('o', 2)]
AssertionError: []
```

Сообщение `assert`

- Вычисляется лениво: `assert cond, long_computation()`
- Лениво пишется (работает любой объект)

```
assert foo(x, y) == bar, (x, y, bar)
```

doctest

```
import itertools

def rle(iterable):
    """Apply run-length encoding to an iterable.

    >>> list(rle(""))
    []
    >>> list(rle("foo"))
    [('f', 1), ('o', 2)]
    """
    for item, g in itertools.groupby(iterable):
        yield item, sum(1 for _ in g)
```

Запуск doctest изнутри

```
def rle(iterable):
    """Apply run-length encoding to an iterable.
    >>> list(rle("foo"))
    [('f', 1), ('o', 2)]
    """
    return []
```

Запуск doctest изнутри

```
$ python rle.py

**********************************************************
File "rle.py", line 4, in __main__.rle
Failed example:
    list(rle("foo"))
Expected:
    [('f', 1), ('o', 2)]
Got:
    []
**********************************************************
1 items had failures:
   1 of   1 in __main__.rle
***Test Failed*** 1 failures.
```

Запуск doctest снаружи

```
$ python -m doctest rle.py
```

Директивы doctest

```
import itertools

def rle(iterable):
    """Apply run-length encoding to an iterable.
    >>> list(rle("mississippi"))
    [('m', 1), ('i', 1), ('s', 2), ('i', 1),
     ('s', 2), ('i', 1), ('p', 2), ('i', 1)]
    """
    for item, g in itertools.groupby(iterable):
        yield item, sum(1 for _ in g)
```

```
***Test Failed*** 1 failures.
```

Директивы doctest

```
import itertools

def rle(iterable):
    """Apply run-length encoding to an iterable.
    >>> list(rle("mississippi")) # doctest: +NORMALIZE_WHITESPACE
    [('m', 1), ('i', 1), ('s', 2), ('i', 1),
     ('s', 2), ('i', 1), ('p', 2), ('i', 1)]
    """
    for item, g in itertools.groupby(iterable):
        yield item, sum(1 for _ in g)
```

Директивы doctest

```
import itertools

def rle(iterable):
    """Apply run-length encoding to an iterable.
    >>> list(rle("mississippi")) # doctest: +ELLIPSIS
    [('m', 1), ... ('i', 1)]
    """
    for item, g in itertools.groupby(iterable):
        yield item, sum(1 for _ in g)
```

Когда использовать doctest?

- Документация устаревает.
- `doctest` — отличное решение для тестирования **документации**.
- Тестировать функциональность при помощи `doctest` — не лучшая идея.

unittest

```
class TestHomework(unittest.TestCase):
    def test_rle_empty(self):
        self.assertEqual(list(rle("")), [])

    def test_rle(self):
        expected = [
            ('f', 2), ('o', 2)
        ]
        self.assertEqual(list(rle("foo")), expected)
```

unittest

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

```
$ python -m unittest rle
```

```
F.
======================================================================
FAIL: test_rle (rle.TestHomework)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "rle.py", line 27, in test_rle
    self.assertEqual(list(rle("foo")), expected)
AssertionError: Lists differ: [] != [('f', 1), ('o', 2)]

Second list contains 2 additional elements.
First extra element 0:
('f', 1)

- []
+ [('f', 1), ('o', 2)]
----------------------------------------------------------------------
Ran 2 tests in 0.001s
FAILED (failures=1)
```

unittest

- Доступен из коробки
- API из Java
- Отличный пример неудачного использования ООР

Quiz: assertions

```
class TestHomework(unittest.TestCase):
    def test_rle_empty_1(self):
        self.assertEqual(list(rle("")), [])

    def test_rle_empty_2(self):
        assert list(rle("") == []
```

В чём разница между двумя `assert`?

Fluent assertions:

```
assertThat(list(rle(""))).isEqualTo([])
```

- Генерируют подробное сообщение об ошибке.
- Многословные.
- Хуже, чем hand-made сообщение об ошибке.
- There is a better way in Python!

setUp / tearDown

```
import unittest
import tempfile

class TestWithTempFile(unittest.TestCase):
    def setUp(self):
        self.tempfile = tempfile.TemporaryFile()

    def tearDown(self):
        self.tempfile.close()

    def test_rle(self):
        self.tempfile.write('mississippi')
        self.assertEqual(rle_encode(self.tempfile),  [
            ...
        ])
```

Проблемы setUp / tearDown

Неявные инварианты

- Почему в `test_rle` `self.tempfile` не `none`?

Отсутствие композиции

- тест **foo** требует базу данных
- тест **bar** требует временный файл
- тесту **baz** нужны и БД, и файл

pytest

- helps you write better programs
- Лучшая библиотека для тестирования
- https://docs.pytest.org/en/latest/

API

```
def rle(iterable):
    return []

def test_rle_foo():
    assert rle("foo") == [('f', 1), ('o', 1)]
```

Преимущества

- API почти отсутствует.
- Добавить новый тест — просто.
- Вероятность, что тесты есть, повышается.

Lifehack

Пишите тривиальные тесты для новой функциональности.

- Написать первый тест сложно — нужно настроить инфраструктуру.
- Когда есть тривиальный тест, написать первый настоящий тест — не лень.
- Копировать тесты — это ОК.

Запуск

```
$ python -m pytest rle.py
```

```
================= test session starts =================
platform linux -- Python 3.6.6, pytest-3.8.2, py-1.7.0, pluggy-
0.7.1
rootdir: /home/matklad/python-course/python-2018/11/code_bits
collected 1 item

rle.py F

====================== FAILURES =======================
_____________________ test_rle_foo ____________________

    def test_rle_foo():
>       assert rle("foo") == [('f', 1), ('o', 1)]
E       AssertionError: assert [] == [('f', 1), ('o', 1)]
E         Right contains more items, first extra item: ('f', 1)
E         Use -v to get the full diff

rle.py:18: AssertionError
================= 1 failed in 0.03 seconds ================
```

Подробные сообщения об ошибках

```
    def test_rle_foo():
>       assert rle("foo") == [('f', 1), ('o', 1)]
E       AssertionError: assert [] == [('f', 1), ('o', 1)]
E         Right contains more items, first extra item: ('f', 1)
E         Full diff:
E         - []
E         + [('f', 1), ('o', 1)]
```

Принцип работы: переписывание AST.

http://pybites.blogspot.com/2011/07/behind-scenes-of-pytests-new-assertion.html

Fixtures

```
# setUp/tearDown => менеджер контекста

@pytest.fixture()
def tempfile():
    with TemporaryFile() as f:
        yield f

def test_with_tempdir(tempfile):
    tempfile.write(b'hello')
    tempfile.seek(0)
    assert tempfile.read() == b'hello'
```

Композиция

```
@pytest.fixture()
def db():
    ...

@pytest.fixture()
def tempfile():
    ...

def test_dump_db_to_file(db, tempfile):
    pass
```

Зависимости

```
@pytest.fixture()
def tempfile():
    with TemporaryFile() as f:
        yield f

@pytest.fixture()
def hello_file(tempfile):
    tempfile.write(b'hello')
    tempfile.seek(0)
    return tempfile

def test_hello(hello_file):
    assert hello_file.read() == b'hello'
```

Больше pytest

- test discovery
- module fixtures
- параметризованные тесты
- тестирование исключений и предупреждений
- встроенные fixtures
- замечательная документация ;)

Зависимости между компонентами

Как тестировать отсылку Emailов?

unittest.mock

- `unittest.mock` позволяет заменить любой объект на магический всемогутор.

```
>>> from unittest.mock import Mock
>>> email_service = Mock()
>>> email_service.send_email('alice@example.com', 'hello')
<Mock name='mock.send_email()' id='140552719212728'>
>>> email_service.send_email.assert_called_once()
>>> email_service.send_spam.assert_called_once()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "mock.py", line 795, in assert_called_once
    raise AssertionError(msg)
AssertionError: Expected 'send_spam' to have been called once.
Called 0 times.
```

To mock or not to mock

Достоинства

- Весь код тестируем.
- Можно проверить тонкие аспекты поведения кода.
- Относительно просто писать тесты.

Недостатки

- Тестируется код, а не функциональность.
- Рефакторинги ломают тесты.
- API mock может отличатся от реального объекта.

Property based testing

Как протестировать сортировку?

```
def sort(xs):
    ...
```

Property based testing

Как протестировать сортировку?

```
def sort(xs):
    ...
```

- Давайте проверим, что результат отсортирован!

Property based testing

```
def is_sorted(xs):
    return all(
        x < y
        for x, y in zip(*[iter(xs)]*2):  # спорная идиома
    )

def check_sorts(xs):
    ys = xs[:]
    sort(ys)
    assert is_sorted(ys), f"failed to sort {xs}"
```

Как выбрать `xs`?

Случайный список

```
import random

def random_list():
    return random.choices(range(10**9), k=1000)
```

Случайный список

```
import random

def random_list():
    return random.choices(range(10**9), k=1000)
```

в этом списке никогда не будет повторяющихся элементов.

Более интересный список

```
import random

# pattern: hard-coded constant -> аргумент по умолчанию
def random_list(max_element=10, max_len=100):
    return random.choices(
        range(max_element + 1),
        k=random.randrange(max_len + 1),
    )
```

hypothesis

```python
from hypothesis import given, strategies as st

@given(st.lists(st.integers()))  # композиция стратегий!
def test_sorted(xs):
    result = sorted(xs)
    result[:2] = result[-2:]  # BUG!
    assert all(xi <= xj for xi, xj in zip(result, result[1:]))
```

hypothesis

Фреймворк для генерации интересный случайных данных и поиска *минимальных* контрпримеров

```
xs = [0, 0, 0, 1]

    @given(st.lists(st.integers()))
    def test_sorted(xs):
        result = sorted(xs)
        result[:2] = result[-2:]  # BUG!
>       assert all(xi <= xj for xi, xj in zip(result, result[1:]))
E       assert False
E        +  where False = all(<generator ... >)

rle.py:38: AssertionError
---------------------- Hypothesis ----------------------
Falsifying example: test_sorted(xs=[0, 0, 0, 1]) 1
================ 1 failed in 0.29 seconds ================
```

1 **минимальный** пример!

Встроенные стратегии

```
st.just(x)           # ==>  x, x, x
st.none()            # ==>  None, None, None
st.one_of(a, b, c)   # ==>  a, a, b, c, a
st.booleans()        # ==>  True, False, True
st.integers()        # ==>  1, -10, 2, 42
st.floats()          # ==>  math.pi, 42.42

# str and bytes
st.text()            # ==>  "abra", "cadabra"
st.binary()          # ==>  b"\xff\xef", b"ascii"

# Collections
st.sampled_from(iterable)
st.tuples(st.foo(), st.bar(), st.boo())
st.lists(st.foo())
st.sets(st.foo())
st.dictionaries(st.foo(), st.bar())
```

Тестируем rle

```
from itertools import chain, repeat, tee
from hypothesis import given, strategies as st

iterables = st.one_of(st.tuples(st.integers(0, 10)),
                      st.lists(st.integers(0, 10)),
                      st.text().map(iter))

@given(iterables)
def test_rle(it):
    def encode_decode(it):
        return chain.from_iterable(
            repeat(item, count) for item, count in rle(it))

    it, copy = tee(it)
    expected = list(copy)
    assert list(encode_decode(it)) == expected
```

Победа

```
rle.py::test_rle PASSED  [100%]
==== 1 passed in 0.81 seconds ====
```

Что читать в транспорте?

- https://docs.pytest.org/en/latest/
- https://hypothesis.readthedocs.io/en/latest/