Углубленный Python

Лекция 7

Опрышко Александр

“

Не забудьте отметиться на занятии!

Цитата великих

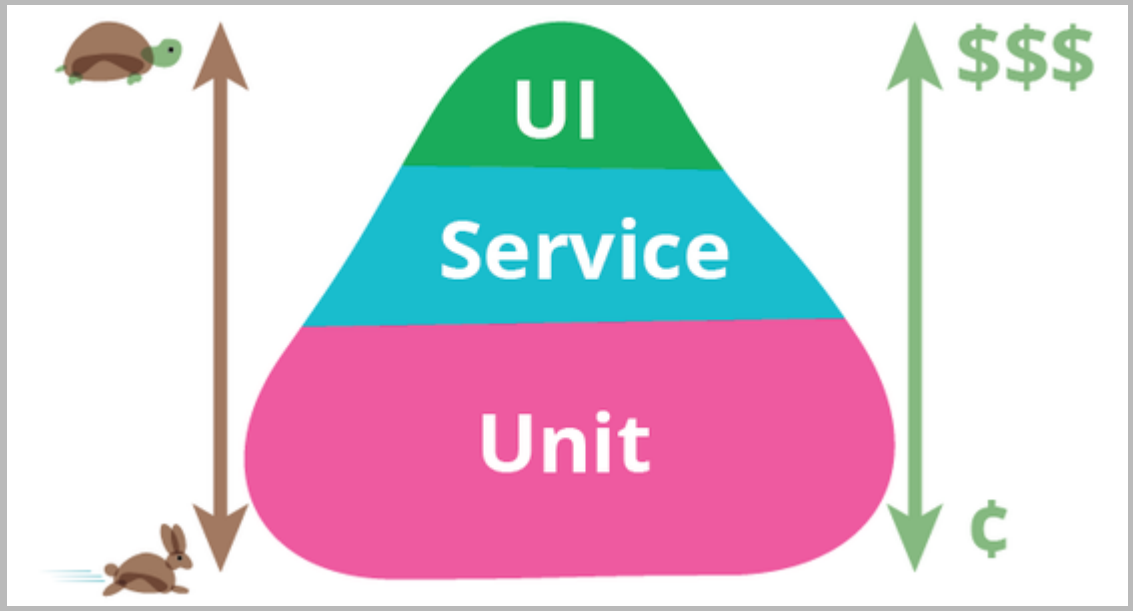
Лекция 6. Что было?

1. ctypes
2. ffi
3. C – extentions
4. Cython

Лекция 7. Что будет?

1. Автоматизация тестирования
2. Юнит-тестирование
3. Unittest
4. Pytest
5. Антипаттерны
6. Тесты на тесты

Как должно быть?



Unit – тестирование

Принципы автоматизации тестов

- Атомарность
- Независимость
- Изолированность/герметичность

Unit – тестирование

Атомарность

```
def test_add_smth():
    user = create_new_user(email='some@ema.il')
    user.register()
    user.auth()
    smth = user.create_smth()
    smth.add()
    assert user.is_authorized()
    assert smth.is_added()
    assert user.email == 'some@ema.il'
```

Unit – тестирование

Атомарность

```
def test_add_smth():
    user = create_new_user(email='some@ema.il')
    user.register()
    user.auth()
    smth = user.create_smth()
    smth.add()
    assert user.is_authorized()
    assert smth.is_added()
    assert user.email == 'some@ema.il'
```

Unit – тестирование

Атомарность

```
def test_add_smth():
    user = create_new_user(email='some@ema.il')
    smth = user.create_smth()
    smth.add()
    assert smth.is_added()
```

Unit – тестирование

Независимость

test1 -> test2 -> test3

test2 -> test1 -> test3

test3 -> test2 -> test1

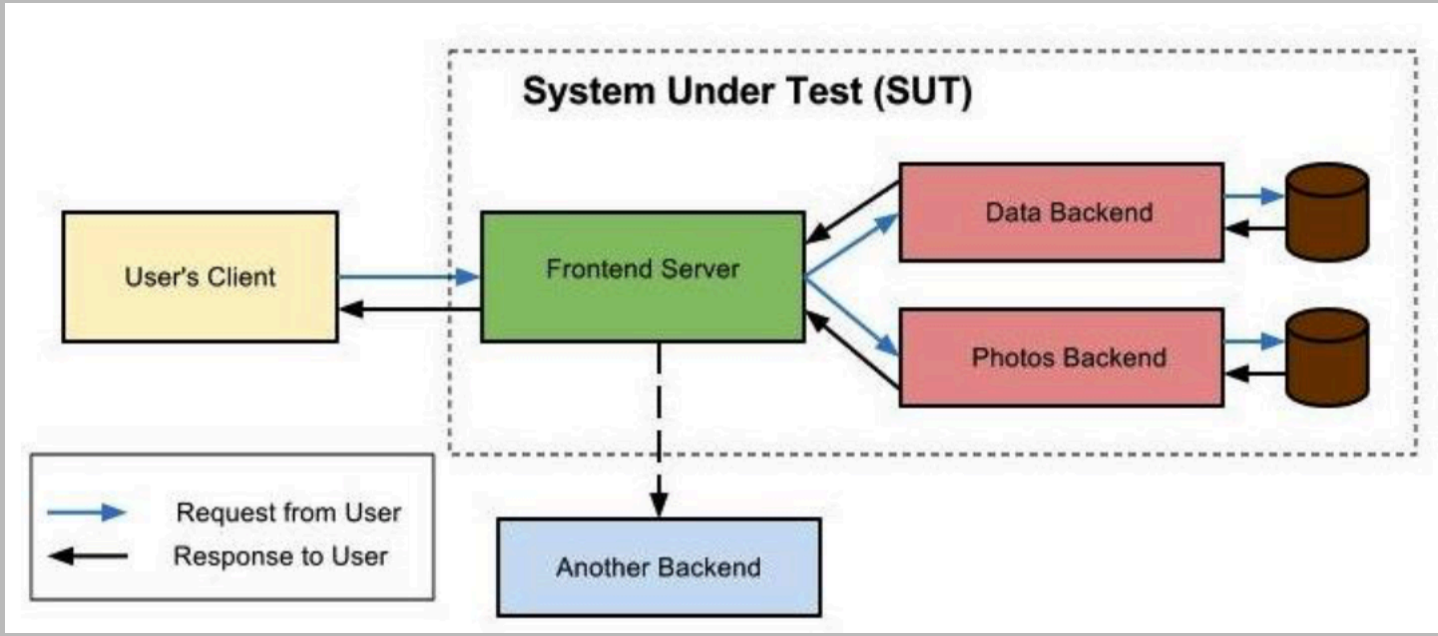
test1 -> test3 -> test2

test3 -> test1 -> test2

test2 -> test3 -> test1

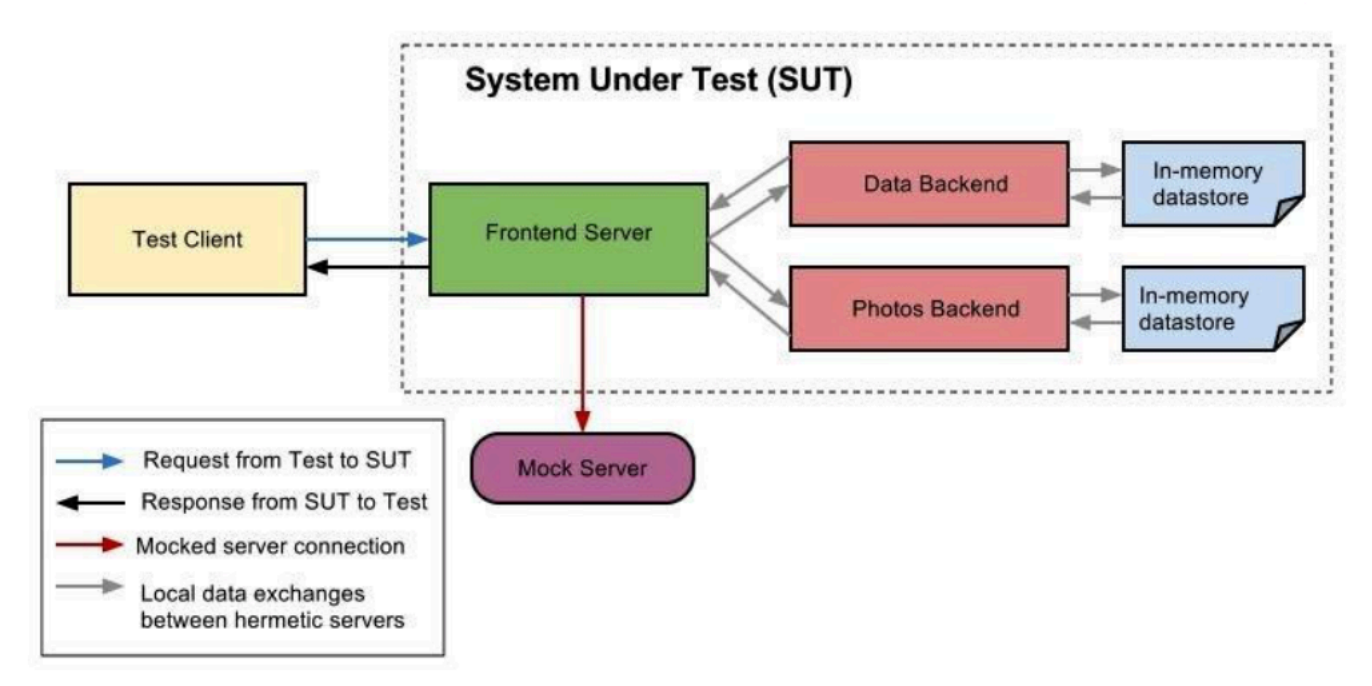
Unit – тестирование

Изолированность/герметичность



Unit – тестирование

Изолированность/герметичность



Unit – тестирование

Как придумать тесты?

- Разбор тестируемоего метода
- Определение граничных значении
- Определение классов эквивалентности

Unit – тестирование

Какие тесты сделать для функции a / b?

Инструменты для тестирования

1. Unittest
2. Pytest

Unittest

```python
class TestStringMethods(unittest.TestCase):
    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a string
        with self.assertRaises(TypeError):
            s.split(2)

    def something(self):
        1 / 0
```

>>> unit/test_simple.py

Unittest

Подготовка состояния (фикстуры)

- setUp()
- tearDown()
- setUpClass()
- tearDownClass()
- setUpModule()
- tearDownModule()

Unittest

Проверка результата

Method	Checks that	New in
`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`	3.1
`assertIsNot(a, b)`	`a is not b`	3.1
`assertIsNone(x)`	`x is None`	3.1
`assertIsNotNone(x)`	`x is not None`	3.1
`assertIn(a, b)`	`a in b`	3.1
`assertNotIn(a, b)`	`a not in b`	3.1
`assertIsInstance(a, b)`	`isinstance(a, b)`	3.2
`assertNotIsInstance(a, b)`	`not isinstance(a, b)`	3.2

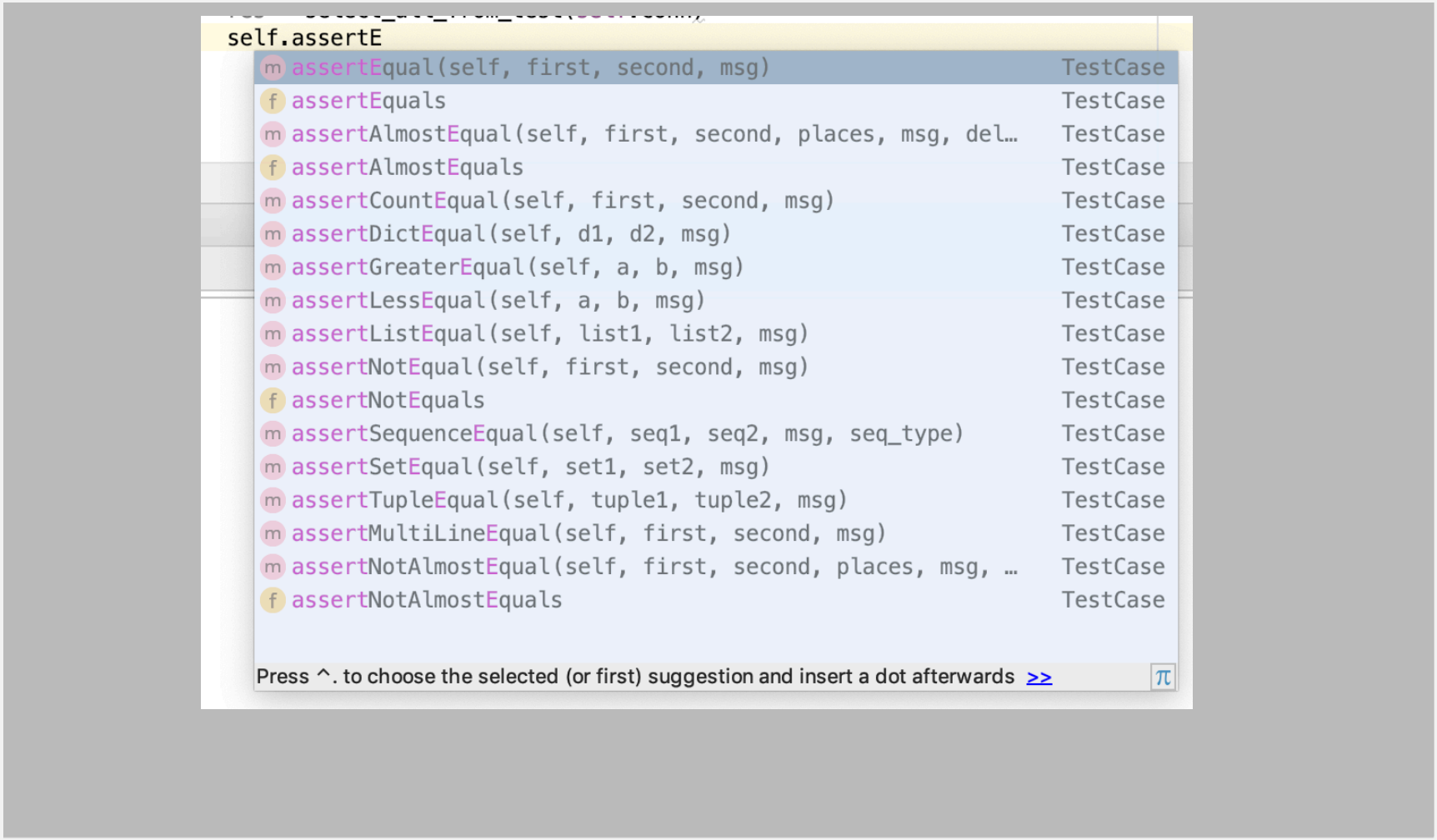
Unittest

```
>>> python -m unittest unit/test_simple.py
>>> python -m unittest unit/test_db.py
>>> python -m unittest discover -s unit/ -p "test_*.py"
```

Почему мне не нравится unittest?

1. 1 класс – 1 тест
2. Fixtures в формате setUp tearDown
3. Скоуп фикстуры
4. Приходится наследоваться если у тестов одинаковый setUp
5. Сложные assert

Почему мне не нравится unittest?



Pytest


```python
def test_upper():
    assert 'foo'.upper() == 'FOO'


def test_isupper():
    assert 'FOO'.isupper()
    assert not 'Foo'.isupper()


def test_split():
    s = 'hello world'
    assert s.split() == ['hello', 'world']

    try:
        s.split(1)
        assert False
    except TypeError:
        assert True
```

>>> pyt/test_simple.py

Pytest

Подготовка состояния (фикстуры)

```
@pytest.fixture(scope='function', autouse=False)
def f():
    print(1)
    yield
    print(2)
```

Pytest

Проверка результата

```
    assert res is None
    assert res is False
    assert res == {}
    assert res == []
    assert isinstance(res, list)
```

Pytest


```
>>> pytest pyt/test_simple.py
>>> pytest pyt/test_db.py
>>> pytest pyt/
```

Pytest

>>> pyt/test_i_like_it.py

Pytest

Как тестировать:

- Aiohttp server (async_pyt/test_api.py)
- Aiohttp client (async_pyt/test_client.py)
- Mock (async_pyt/test_mock.py)

Антипаттерны

- Liar
- Giant
- Secret Catcher
- Mockery
- Enumerator
- Slowpoke

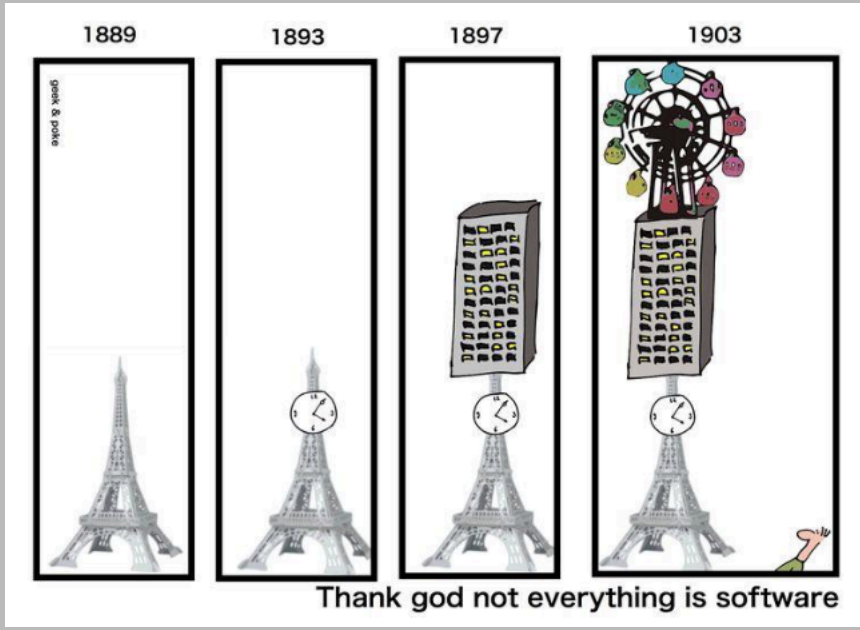
Антипаттерны

Liar

```
def test_some_smth():
    do_smth()
    assert True
```

Антипаттерны

Giant



Антипаттерны

The Secret Catcher

```
def test_smth_one_two_three():
    do_smth1()
    do_smth2()
    do_smth3()
```

Антипаттерны

Mockery

>>> async_pyt/test_mock.py

Антипаттерны

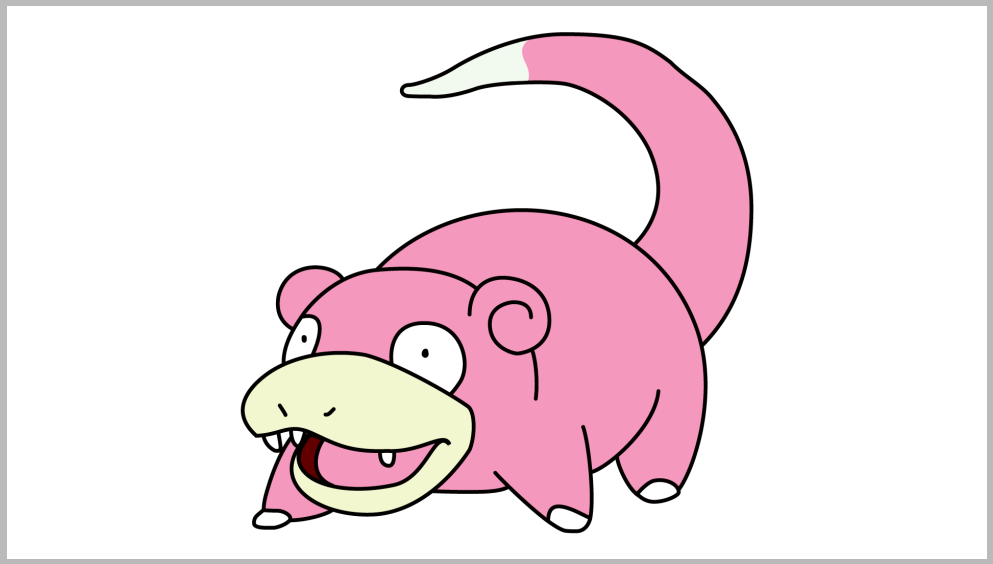
Enumerator

```
def test1():
    pass

def test2():
    pass

def test3():
    pass
```

Антипаттерны

Slowpoke

```
def test_pokemon():
    do_smth1()
    time.sleep(N)
    assert True
```



Тесты на тесты

Анализ покрытия

- По файлам
- По классам
- По методам
- По строкам
- По ветвям

Тесты на тесты

Mutation testing

Метод тестирования основанный на внесении небольших изменений в код программы.

Тесты на тесты

Smoke test

Зачем писать тесты?

- Улучшение качества
- Облегчение внесения изменений
- Документация на продукт

Зачем писать тесты?

[redacted]

Скинь пожалуйста какой метод и что отправлять на стату

Alexander Opryshko

tests.apps.user.test_client.TestClientBulk

тесты

```
res = await cli.post(
    '/client.bulk',
    json={
        'token': token,
        'actions': [
            {
                'type': 'triggers_flushed',
                'time_delta': 10,
                'mark': 'test'
            },
            {
                'type': 'call',
                'time_delta': 20,
            }
        ]
    }
)
assert res.status == 200
```

Ссылки

1. https://docs.python.org/3/library/unittest.html
2. https://docs.pytest.org/en/latest/
3. https://github.com/pytest-dev/pytest-asyncio
4. https://github.com/aio-libs/pytest-aiohttp
5. https://aiohttp.readthedocs.io/en/stable/testing.html
6. https://pypi.org/project/pytest-mock/
7. https://docs.python.org/dev/library/unittest.mock.html
8. https://docs.pytest.org/en/latest/monkeypatch.html
9. https://github.com/spulec/freezegun
10. https://github.com/pnuckowski/aioresponses

telegram: alexopryshko
email: alexopryshko@gmail.com

telegram: igorcoding
email: igor.latkin@outlook.com

Спасибо за внимание!