

Юнит-тесты

Код, который проверяет код, что бы уволить тестировщиков.

```
def my_test():  
    if the_answer_to_the_ultimate_question() != 42:  
        raise SomeError("It does not look like anything")
```

Структура

- Дерево файлов в отдельной папке(tests), которая частично моделирует структуру основного проекта(если только не ява)
- Версионирование вместе с кодом

Тесты

- Методы в классах или функции
- Инициализация
- Код теста - выполняется действие и результат сверяется с требуемым

Фреймворк

- Поиск тестов
- Инициализация, контроль исполнения
- Отчеты, взаимодействие с CI/CD
- Библиотека вспомогательных функций
- Контроль логов
- Вызов отладчика по ошибке, etc
- Шедулинг исполнения
- Профилирования
- История исполнения
-

assert & AssertionError

- AssertionError - lingua franca юнит-тестов
- Все тестирующие ф-ции выбрасывают их, все фреймворки - ловят
- `assert expr [, msg] — assert x == 3`

```
x = 1
```

```
y = 2
```

```
assert x == y, "x(={}) must be equals y(={})".format(x, y)
```

```
assert the_answer_to_the_ultimate_question() == 42
```

Фреймворки

- unittest - stdlib, тяжелое наследия явы, в 3 починили слегка
- nose - <http://nose.readthedocs.io/en/latest/>, использует unittest
- pytest - <https://docs.pytest.org/en/latest>

Проверка

```
1      assert response == 250      <<< pytest
2      self.assertEqual(response, 250)  <<< unittest
3      ok(response) == 250      <<< oktest
```

Инициализация и очистка(nose+unittest)

- setUpClass
- setUp
- tearDown
- tearDownClass

Инициализация pytest

- `setup()` - module level
- `teardown()` - module level
- `setup_module(module)`
- `teardown_module(module)`
- `setup_function(function)`
- `teardown_function(function)`
- `setup_class(cls)`
- `teardown_class(cls)`

pytest fixtures

```
1 @pytest.fixture(params=[v1, v2])
2 def resource_setup(request):
3     print("resource_setup")
4     yield (some_data, request.param)
5     print("resource_teardown")
6
7 @pytest.mark.slowtest
8 def some_test(resource_setup):
9     pass
```

Совместное использование fixtures

Поиск и исполнение тестов

```
1 $ pytest test_mod.py::test_func
2 $ pytest -s test_mod.py::TestClass::test_method
3 $ pytest -v -m webtest
```

Описывайте записимости явно

```
1 def some_test():
2     some_initialization("x", "y", "z")
3     test_code()
4     some_deinitialization()
5
6 def some_test2():
7     with some_initialization("x", "y", "z"):
8         test_code()
9
10 @require_some_init("x", "y", "z")
11 def some_test3():
12     test_code()
```

oktest

```
1  from oktest import ok
2
3  def simple_test():
4      ok(1) == 1
5      ok([]).is_a(list)
6      ok("/tmp/x.txt").is_file()
```

mock

```
1  import os
2  import glob
3
4  def remove_tmp_files():
5      for fname in glob.glob("/tmp/*.tmp"):
6          os.unlink(fname)
```

```
1  import mock
2
3  def glob_mock( path ):
4      return [ "/tmp/x.tmp" , "/tmp/y.sh" ]
5
6  all_unlinked = []
7  def unlink_mock( path ):
8      return all_unlinked.append( path )
9
10 @mock.patch( "os.unlink" , unlink_mock )
11 @mock.patch( "glob.glob" , glob_mock )
12 def test_remove_tmp():
13     remove_tmp_files()
14     ok( all_unlinked ) == [ "/tmp/x.tmp" ]
```