



Тестирование приложений на языке Python

НИКОЛАЙ МАРКОВ (@ENCHANTNER)



Виды тестирования

- Функциональное тестирование (functional testing)
- Системное тестирование (system testing), сюда же интеграционное тестирование (integrational testing)
- Тестирование производительности (performance testing)
- Регрессионное тестирование (regression testing)
- Модульное тестирование (unit testing)
- Тестирование безопасности (security testing)
- Тестирование локализации (localization testing)
- Юзабилити тестирование (usability testing)

Дымовое тестирование (smoke testing) - это не вид, скорее, подход

TDD = Test Driven Development, подход, когда тесты пишутся до кода

- Фикстура (fixture) - подготовительная и завершающая логика, еще часто - фейковые данные, в том числе для тестирования “угловых ситуаций”.
- Кейс (case) - проверка, сравнение группы вариантов ввода с ожидаемыми выводами.
- Набор тестов (suite) - группа тестов, предназначенная для запуска вместе (в одних и тех же условиях, возможно, даже параллельно).
- Раннер или оркестратор (runner/orchestrator) - компонент, который запускает тесты, обрабатывает их результаты и выдает отчеты в человеко- или машиночитаемом виде.
- Мок/макет (mock), он же monkeypatch - тестовый объект, предназначенный для эмуляции поведения/интерфейса “боевого” объекта, использующийся для изолирования области тестирования.



NEW
PRO
LAB

Инструменты экосистемы Python



Чего почитать

<https://docs.python.org/3/library/unittest.html>

<http://docs.python-guide.org/en/latest/writing/tests/>

`~$ pip install unittest2 mock` # если вам очень не повезло с версией питона (≤ 2.6)

```
import unittest

class MyTestCase1(unittest.TestCase):
    def setUp(self):
        ... КОД для подготовки окружения ...
    def tearDown(self):
        ... КОД для запуска после тестов (очистки) ...
    def test_feature_one(self):
        # Тест функциональности 1
        ... КОД теста ...
    def test_feature_two(self):
        # Тест функциональности 2
        ... КОД теста ...

class MyTestCase2(unittest.TestCase):
    ... ТО ЖЕ САМОЕ ...

... больше классов ...

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

а еще есть setUpClass()/tearDownClass()

~\$ *python3 -m unittest test.MyTestCase1*

```
import os
import unittest
import unittest.mock as mock

def make_dir():
    os.mkdir("/tmp/test")

class TestMkdir(unittest.TestCase):

    @mock.patch('os.mkdir') # важно помнить области видимости!
    def test_mkdir(self, mkdir_mock):
        make_dir()
        mkdir_mock.assert_called_once_with("/tmp/test")
```

<https://docs.python.org/3/library/unittest.mock.html>

<https://www.toptal.com/python/an-introduction-to-mocking-in-python>

~\$ *pip install pytest*

~\$ *py.test # без точки тоже работает*

```
def func(x):  
    return x * 2  
  
def test_func():  
    assert func(2) == 4
```

```
def f():  
    return 3  
  
def test_function():  
    assert f() == 4
```

```
import pytest  
  
def divide(a, b):  
    return a / b  
  
def test_zero_division():  
    with pytest.raises(ZeroDivisionError):  
        divide(1, 0)
```

```
def test_recursion_depth():  
    with pytest.raises(RuntimeError) as excinfo:  
        def f():  
            f()  
        f()  
    assert 'maximum recursion' in str(excinfo.value)
```



```
import pytest
```

```
@pytest.fixture
```

```
def smtp():
```

```
    import smtplib
```

```
    return smtplib.SMTP("smtp.gmail.com", 587, timeout=5)
```

```
def test_ehlo(smtp):
```

```
    response, msg = smtp.ehlo()
```

```
    assert response == 250
```

```
    assert 0 # для демонстрации
```

а еще можно использовать
request.addfinalizer()

```
import smtplib
```

```
import pytest
```

```
@pytest.fixture(scope="module")
```

```
def smtp():
```

```
    smtp = smtplib.SMTP("smtp.gmail.com", 587, timeout=5)
```

```
    yield smtp # вернуть значение фикстуры
```

```
    print("teardown smtp")
```

```
    smtp.close()
```

```
def test_myoutput(capsys): # магический параметр!  
    print("hello")  
    sys.stderr.write("world\n")  
    captured = capsys.readouterr()  
    assert captured.out == "hello\n"  
    assert captured.err == "world\n"  
    print("next")  
    captured = capsys.readouterr()  
    assert captured.out == "next\n"
```

```
import functools  
  
def test_partial(monkeypatch):  
    with monkeypatch.context() as m:  
        m.setattr(functools, "partial", 3)  
        assert functools.partial == 3
```

~\$ *py.test -x* # упасть при первом же шухере

~\$ *py.test --maxfail=2* # упасть после двух шухеров



Еще инструменты

- [nose/nose2](#) - чуть устарелый раннер
- [coverage](#) - показывает “покрытие”
- [hypothesis](#) - проверка “гипотез”
- [tox](#) - тестирование в разных окружениях
- [responses](#) - моки для requests
- [moto](#) - моки для AWS

<https://www.jetbrains.com/help/pycharm/testing-your-first-python-application.html>

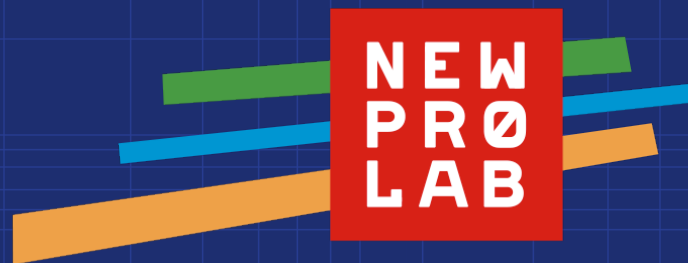
<http://flask.pocoo.org/docs/1.0/testing/>

<https://pypi.org/project/pytest-django/>

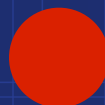
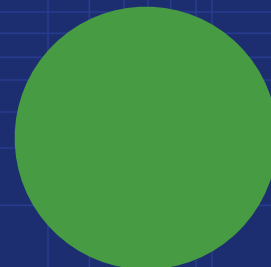
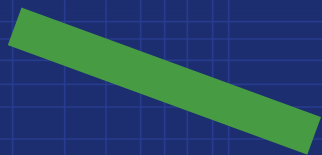
<https://stackoverflow.com/questions/29881236/how-to-mock-asyncio-coroutines>

<https://stackoverflow.com/questions/17001010/how-to-run-unittest-discover-from-python-setup-py-test>

<https://wiki.python.org/moin/PythonTestingToolsTaxonomy>



Прочие инструменты





Нагрузочное тестирование

Берем и генерируем миллионы запросов

- [Apache Benchmark](#)
- [Siege](#) (чуть устарело)
- [Locust](#)
- [WRK](#)

<https://www.softwaretestinghelp.com/performance-testing-tools-load-testing-tools/>



Нагрузочное тестирование

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

~\$ ab -kc 10 -t 60 http://127.0.0.1:5000/



Тестирование интерфейса

Web:

- [Selenium](#)
- [Nightmare](#)

Any GUI:

- [Sikuli](#)
- [LDTP](#)
- [xdotool](#)
- [xautomation](#)


```
import unittest
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

class PythonOrgSearch(unittest.TestCase):

    def setUp(self):
        self.driver = webdriver.Firefox()

    def test_search_in_python_org(self):
        driver = self.driver
        driver.get("http://www.python.org")
        self.assertIn("Python", driver.title)
        elem = driver.find_element_by_name("q")
        elem.send_keys("pycon")
        assert "No results found." not in driver.page_source
        elem.send_keys(Keys.RETURN)

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

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

*Для удаленного тестирования
нужен Selenium Standalone [Server](#)*

<https://github.com/mozilla/geckodriver/releases>

<https://sites.google.com/a/chromium.org/chromedriver/downloads>