Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5
“Модульное тестирование в Python”

Выполнил:
студент группы ИУ5-34Б:
Стукалов Иван Дмитриевич
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

2022

Описание задания:

Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.

Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.

Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:

TDD - фреймворк (не менее 3 тестов).
BDD - фреймворк (не менее 3 тестов).

Создание Mock-объектов (необязательное дополнительное задание).

Текст программы:

TDD-тестирование:

// файл tdd_tests.py

```
import unittest
from lab_python_fp.field import field
from lab_python_fp.unique import Unique
from lab_python_fp.sort import sortedList


class TestingLab(unittest.TestCase):

    def test_field(self):
        goods = [
            {'title': 'Ковер', 'price': 2000, 'color': 'green'},
            {'title': 'Диван для отдыха', 'color': 'black'}
        ]
        self.assertEqual(field(goods, 'title'), [{'title': 'Ковер'},
{'title': 'Диван для отдыха'}])
        self.assertEqual(field(goods, 'title', 'price'),
                         [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван
для отдыха'}])

    def test_unique(self):
        count_list = ["aaa", "b", "c", "b", "A", "b", "AAA", "Aaa", "a"]
        unique = Unique(count_list)
        unique_it = Unique.__iter__(unique)
        self.assertEqual(Unique.__next__(unique_it), "aaa")
        self.assertEqual(Unique.__next__(unique_it), "b")
        self.assertEqual(Unique.__next__(unique_it), "c")

    def test_sort(self):
        list_arr = [-3, 6, -2, 5, -4, 3, -10, -53, 8]
        self.assertEqual(sortedList(list_arr), [8, 6, 5, 3, -2, -3, -4, -10,
-53])
        self.assertEqual(sortedList(list_arr, lambda x: abs(x)), [-53, -10,
8, 6, 5, -4, 3, -3, -2])
```

BDD-тестирование:

//файл bdd_field.feature

```
# content for bdd_field.feature

  Feature: Filtering fields

    Scenario: Connecting field
      Given I have a data

      When func works with 2 args
      And func works with 3 args

      Then I have filtered fields from dict with 2 args
      And I have filtered fields from dict with 3 args
```

// файл bdd_sort.feature

```
# content for sort

  Feature: Sorting the array

    Scenario: Connecting sorting

      Given Array for sorting

      When sorting works no lambda
      And sorting works with lambda

      Then I have sorted arr no lambda
      And I have sorted arr with lambda
```

//файл bdd_unique.feature

```
# content for sort

  Feature: Sorting the array

    Scenario: Connecting sorting

      Given Array for sorting

      When sorting works no lambda
      And sorting works with lambda

      Then I have sorted arr no lambda
      And I have sorted arr with lambda
```

// файл bdd_test_field.py

```
from pytest_bdd import scenario, given, when, then
from lab_python_fp.field import field


@scenario('bdd_tests/bdd_field.feature', 'Connecting field')
def test_field_1():
    pass


@given("I have a data", target_fixture="data")
def data():
    return [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'color': 'black'}
```

```
    ]


@when("func works with 2 args", target_fixture="result_1")
def func_1(data):
    return field(data, 'title')


@when("func works with 3 args", target_fixture="result_2")
def func_2(data):
    return field(data, 'title', 'price')


@then("I have filtered fields from dict with 2 args")
def res_1(result_1):
    assert result_1 == [{'title': 'Ковер'}, {'title': 'Диван для отдыха'}]


@then("I have filtered fields from dict with 3 args")
def res_2(result_2):
    assert result_2 == [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван
для отдыха'}]
```

// файл bdd_test_sort.py

```
from pytest_bdd import scenario, given, when, then
from lab_python_fp.sort import sortedList


@scenario('bdd_tests/bdd_sort.feature', 'Connecting sorting')
def test_sort():
    pass


@given('Array for sorting', target_fixture="data")
def get_data():
    return [-3, 6, -2, 5, -4, 3, -10, -53, 8]


@when('sorting works no lambda', target_fixture='result_1')
def func_1(data):
    data_to_no_lambda = data
    return sortedList(data_to_no_lambda)


@when('sorting works with lambda', target_fixture='result_2')
def func_2(data):
    data_to_lambda = data.copy()
    return sortedList(data_to_lambda, lambda x: abs(x))


@then('I have sorted arr no lambda')
def res_1(result_1):
    assert result_1 == [8, 6, 5, 3, -2, -3, -4, -10, -53]


@then('I have sorted arr with lambda')
def res_2(result_2):
    assert result_2 == [-53, -10, 8, 6, 5, -4, 3, -3, -2]
```

// файл bdd_test_unique.py

```
from pytest_bdd import scenario, given, when, then
from lab_python_fp.unique import Unique
```

```python
@scenario('bdd_tests/bdd_unique.feature', 'Connecting generator')
def test_unique():
    pass


@given("Some args", target_fixture="data")
def get_data():
    count_list = ["aaa", "b", "c", "b", "A", "b", "AAA", "Aaa", "a"]
    unique = Unique(count_list)
    unique_it = Unique.__iter__(unique)
    return unique_it


@when("func works first time", target_fixture="result_1")
def func_1(data):
    return Unique.__next__(data)


@when("func works second time", target_fixture="result_2")
def func_2(data):
    return Unique.__next__(data)


@when("func works third time", target_fixture="result_3")
def func_3(data):
    return Unique.__next__(data)


@then("I have unique arr first time")
def res_1(result_1):
    assert result_1 == "aaa"


@then("I have unique arr second time")
def res_2(result_2):
    assert result_2 == "b"


@then("I have unique arr third time")
def res_3(result_3):
    assert result_3 == "c"
```

Результаты работы программы:

```
============================= test session starts =============================
collecting ... collected 1 item

bdd_test_field.py::test_field_1 <- venv\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 1 passed in 0.05s ==============================
```

```
============================= test session starts =============================
collecting ... collected 1 item

bdd_test_sort.py::test_sort <- venv\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 1 passed in 0.04s ==============================
```

```
============================= test session starts ==============================
collecting ... collected 1 item

bdd_test_unique.py::test_unique <- venv\lib\site-packages\pytest_bdd\scenario.py PASSED [100%]

============================== 1 passed in 0.04s ===============================
```