Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5

Выполнил:
студент группы ИУ5-34Б
Баширов Г. К.
Подпись и дата:

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

Москва, 2022 г.

Задание

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

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

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

o TDD - фреймворк (не менее 3 тестов).

o BDD - фреймворк (не менее 3 тестов).

o Создание Mock-объектов (необязательное дополнительное задание).

Текст программы

field.py

```
def field(items, *args):
    assert len(args) > 0
    if len(args) == 1:
        return [d[args[0]] for d in items if args[0] in d]
    else:
        return [{key: d[key] for key in args if key in d} for d in items]


goods = [
    {'title': 'Ковер', 'price': 2000, 'color': 'green'},
    {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
]
data = ['title', 'price']
print(field(goods, *data))
```

sort.py

```
def sort_without_lambda(data):
    return sorted(data, reverse=True, key=abs)


def sort_with_lambda(data):
    return sorted(data, reverse=True, key=lambda x: abs(x))
```

tdd field.py

```
import unittest

from field import field


class Test(unittest.TestCase):
    def test_field_one_argument(self):
        goods = [
           {'title': 'Ковер', 'price': 2000, 'color': 'green'},
           {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
        ]
        res = field(goods, 'title')
        exp = ['Ковер', 'Диван для отдыха']
```

```
        self.assertEqual(res, exp)

    def test_field_many_arguments(self):
        goods = [
           {'title': 'Ковер', 'price': 2000, 'color': 'green'},
           {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
        ]
        res = field(goods, 'title', 'price')
        exp = [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для
отдыха', 'price': 5300}]
        self.assertEqual(res, exp)

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

tdd_sort.py

```
import unittest

from sort import sort_without_lambda, sort_with_lambda


class Test(unittest.TestCase):
    def test_sort_without_lambda(self):
        data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
        res = sort_without_lambda(data)
        exp = [123, 100, -100, -30, 4, -4, 1, -1, 0]
        self.assertEqual(res, exp)

    def test_sort_with_lambda(self):
        data = [4, -30, 100, -100, 123, 1, 0, -1, -4]
        res = sort_with_lambda(data)
        exp = [123, 100, -100, -30, 4, -4, 1, -1, 0]
        self.assertEqual(res, exp)


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

bdd_field.py

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


@scenario('field.feature', 'one argument')
def test_field_one_argument():
    pass


@scenario('field.feature', 'many arguments')
def test_field_many_arguments():
    pass


@given('data: "goods" and argument: "title"', target_fixture='data1')
def data():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]
    args = ['title']
    return [goods, args]


@given('data: "goods" and argument: "title", "price"',
target_fixture='data2')
```

```
def data():
    goods = [
        {'title': 'Ковер', 'price': 2000, 'color': 'green'},
        {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
    ]
    args = ['title', 'price']
    return [goods, args]


@when('running field function with one argument', target_fixture='result1')
def field_one_arg(data1):
    return field(data1[0], *data1[1])


@when('running field function with many arguments', target_fixture='result2')
def field_many_args(data2):
    return field(data2[0], *data2[1])


@then('the result is: "Ковер", "Диван для отдыха"')
def result(result1):
    assert result1 == ["Ковер", "Диван для отдыха"]


@then('the result is: {"title": "Ковер", "price": 2000}, {"title": "Диван для
отдыха", "price": 5300}')
def result(result2):
    assert result2 == [{"title": "Ковер", "price": 2000}, {"title": "Диван
для отдыха", "price": 5300}]
```

bdd_sort.py

```
from sort import sort_without_lambda, sort_with_lambda
from pytest_bdd import scenario, given, when, then


@scenario('sort.feature', 'sorting without lambda')
def test_sort_without_lambda():
    pass


@scenario('sort.feature', 'sorting with lambda')
def test_sort_with_lambda():
    pass


@given('data: [4, -30, 100, -100, 123, 1, 0, -1, -4]', target_fixture='data')
def data():
    return [4, -30, 100, -100, 123, 1, 0, -1, -4]


@when('running sort_without_lambda function', target_fixture='result')
def sort_wo_l(data):
    return sort_without_lambda(data)


@when('running sort_with_lambda function', target_fixture='result')
def sort_w_l(data):
    return sort_with_lambda(data)


@then('the result is sorted data: [123, 100, -100, -30, 4, -4, 1, -1, 0]')
def result(result):
    assert result == [123, 100, -100, -30, 4, -4, 1, -1, 0]
```

field.feature

```
Feature: field

  Scenario: one argument
    Given data: "goods" and argument: "title"
    When running field function with one argument
    Then the result is: "Ковер", "Диван для отдыха"

  Scenario: many arguments
    Given data: "goods" and argument: "title", "price"
    When running field function with many arguments
    Then the result is: {"title": "Ковер", "price": 2000}, {"title": "Диван
для отдыха", "price": 5300}
```

sort.feature

```
Feature: sorting

  Scenario: sorting without lambda
    Given data: [4, -30, 100, -100, 123, 1, 0, -1, -4]
    When running sort_without_lambda function
    Then the result is sorted data: [123, 100, -100, -30, 4, -4, 1, -1, 0]

  Scenario: sorting with lambda
    Given data: [4, -30, 100, -100, 123, 1, 0, -1, -4]
    When running sort_with_lambda function
    Then the result is sorted data: [123, 100, -100, -30, 4, -4, 1, -1, 0]
```

Анализ результатов

```
tdd_field.py::Test::test_field_many_arguments PASSED                  [ 50%]
tdd_field.py::Test::test_field_one_argument PASSED                    [100%]
```

```
tdd_sort.py::Test::test_sort_with_lambda PASSED                       [ 50%]
tdd_sort.py::Test::test_sort_without_lambda PASSED                    [100%]
```

```
bdd_field.py::test_field_one_argument <- venv/lib/python3.8/site-packages/pytest_bdd/scenario.py PASSED [ 50%]
bdd_field.py::test_field_many_arguments <- venv/lib/python3.8/site-packages/pytest_bdd/scenario.py PASSED [100%]
```

```
bdd_sort.py::test_sort_without_lambda <- venv/lib/python3.8/site-packages/pytest_bdd/scenario.py PASSED [ 50%]
bdd_sort.py::test_sort_with_lambda <- venv/lib/python3.8/site-packages/pytest_bdd/scenario.py PASSED [100%]
```