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 ИУ5-32
Щепетов Дмитрий
Подпись и дата:

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

Москва, 2022 г.

Описание задания

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:
    - TDD - фреймворк (не менее 3 тестов).
    - BDD - фреймворк (не менее 3 тестов).
    - Создание Mock-объектов (необязательное дополнительное задание).

Текст программы

Файл sort.py, который проходит через модульные тесты

```
data1 = [4, -30, 100, -100, 123, 1, 0, -1, -4]
def res(data):
    res = sorted(data, key=abs, reverse = True)
    return res
def res_with_lambda(data):
    result_with_lambda = sorted(data, key=lambda n: n*n, reverse = True)
    return result_with_lambda

if __name__ == '__main__':
    print(res(data1))
    print(res([0, -300, 300, 54, -54, 54, 99, 19, 14, -19]))
    print(res_with_lambda(data1))
```

Файл test_tdd.py с модульными тестами TDD

```
import unittest
import sort
class test_sort(unittest.TestCase):
    def test_res1(self):
        data = [2, -6, 3, 0, 7]
        result = [7, -6, 3, 2, 0]
        assert result == sort.res(data)
    def test_res2(self):
        data = [2, -5, 5, 8, 0, 1, -1, 13]
        result = [13, 8, -5, 5, 2, 1, -1, 0]
        self.assertEqual(sort.res(data), result)
    def test_res3(self):
        self.assertEqual(sort.res([0, -300, 300, 54, -54, 54, 99, 19, 14, -19]),
[-300, 300, 99, 54, -54, 54, 19, -19, 14, 0])

    def test_rwl1(self):
```

```
        data = [2, -6, 3, 0, 7]
        result = [7, -6, 3, 2, 0]
        assert result == sort.res_with_lambda(data)
    def test_rwl2(self):
        data = [2, -5, 5, 8, 0, 1, -1, 13]
        result = [13, 8, -5, 5, 2, 1, -1, 0]
        self.assertEqual(sort.res_with_lambda(data), result)
    def test_rwl3(self):
        self.assertEqual(sort.res_with_lambda([0, -300, 300, 54, -54, 54, 99, 19,
14, -19]), [-300, 300, 99, 54, -54, 54, 19, -19, 14, 0])
if __name__ == "__main__":
    unittest.main()
```

Файл test_bdd.feature для модульных тестов BDD

```
Feature: Unique selection
    Scenario: list of numbers
    Given list of numbers: [1, 1, 3, 1, 2, 2, 3, 3]
    When start Unique iterator
    Then values, numbers [1, 3, 2]

    Scenario: list of letters
    Given list of letters: ["a", "A", "b", "B", "a", "A", "b", "B"]
    When start Unique iterator
    Then values, letters ['a', 'A', 'b', 'B']

    Scenario: list of letters with ignore_case=True
    Given list of letters: ["a", "A", "b", "B", "a", "A", "b", "B"]
    When start Unique iterator + ignore_case
    Then values, letters ['a', 'b']
```

Файл test_bdd.py

```
from behave import given, when, then
import unique


@given('list of numbers: [{arr}]')
def step(context, arr):
    context.data = [int(elem) for elem in arr.split(', ')]


@given('list of letters: [{arr}]')
def step(context, arr):
    context.data = arr.split(', ')


@when('start Unique iterator')
def step(context):
```

```
    context.res = list(unique(context.data))


@when('start Unique iterator + ignore_case')
def step(context):
    context.res = list(unique(context.data, ignore_case=True))


@then('values, numbers [{arr}]')
def step(context, arr):
    assert context.res == [int(i) for i in arr.split(', ')]


@then('values, letters [{arr}]')
def step(context, arr):
    assert sorted(context.res) == sorted(arr.split(', '))
```

Анализ результатов

```
PS C:\Users\Dima\Documents\tasks_c++\Python> & C:/Users/Di
c++/Python/Laba5/test_tdd.py
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
PS C:\Users\Dima\Documents\tasks_c++\Python>
```