**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: |  | Проверил: |
| студент группы ИУ5-32 |  | преподаватель каф. ИУ5 |
| Щепетов Дмитрий |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | 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**

