**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана (национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

**Курс «Базовые компоненты интернет-технологий»**

Отчёт по лабораторной работе №5

Выполнил:

студент группы ИУ5-33Б

Александр Саргсян

Проверил:

к.т.н., доц., Ю. Е. Гапанюк

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

**Текст программы:**

*sort.py*

def sort1(data):  
 result = sorted(data, key=abs, reverse=True)  
 return result  
  
def sort2(data):  
 result = sorted(data, key=abs, reverse=True)  
 return result  
  
def sort3(data):  
 result = sorted(data, key=abs, reverse=True)  
 return result  
  
def sort\_bdd(data):  
 it = sorted(data, key=abs, reverse=True)  
 return it

*main.py*

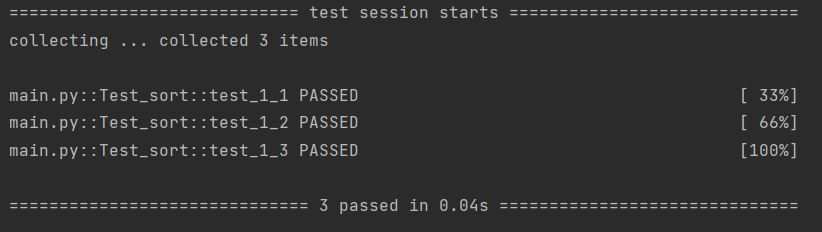
import unittest  
import sort  
class Test\_sort(unittest.TestCase):  
 def test\_1\_1(self):  
 self.assertEqual(sort.sort1([5, -9, 7, 1, -3]), [-9, 7, 5, -3, 1])  
 def test\_1\_2(self):  
 self.assertEqual(sort.sort2([7, -11, 13, 3, 2, 5, -8]), [13, -11, -8, 7, 5, 3, 2])  
 def test\_1\_3(self):  
 self.assertEqual(sort.sort3([15, 11,-2, -6, 12, 3, 8, 0]), [15, 12, 11, 8, -6, 3, -2, 0])

*behavior.py*

from pytest\_bdd import \*  
from sort import sort\_bdd  
  
@scenario("C:/Users/5/PycharmProjects/Lab\_5/features/behave.feature", 'sorting of list')  
def testing\_sort():  
 pass  
  
@given('list', target\_fixture = 'given\_list')  
def given\_list():  
 return [5, -7, 13, 2, -9, 10]  
  
@when('we sort it', target\_fixture = 'sorting')  
def sorting(given\_list):  
 return sort\_bdd(given\_list)  
  
@then('we shall get sorted list')  
def final\_list(sorting):  
 assert(sorting == [13, 10, -9, -7, 5, 2])

*behave.feature*

Feature: Some list sorting  
 Scenario: sorting of list  
 Given list  
 When we sort it  
 Then we shall get sorted list

**Результаты выполнения:**

