Московский государственный технический университет им. Н.Э. Баумана

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

Отчет по лабораторной работе №5
«Парадигмы и конструкции языков программирования»

Выполнил:
студент группы ИУ5-35Б
Удалова Виктория
Подпись и дата:

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

Постановка задачи

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).

Код программы на Python

Main_1.py

```python
class Levenshtein:
    7 usages  new *
    def levenshtein_distance(self, word1, word2):
        matrix = [[0] * (len(word2) + 1) for _ in range(len(word1) + 1)]

        for i in range(len(word1) + 1):
            matrix[i][0] = i
        for j in range(len(word2) + 1):
            matrix[0][j] = j

        for i in range(1, len(word1) + 1):
            for j in range(1, len(word2) + 1):
                if word1[i - 1] == word2[j - 1]:
                    cost = 0
                else:
                    cost = 1
                matrix[i][j] = min(
                    matrix[i - 1][j] + 1,
                    matrix[i][j - 1] + 1,
                    matrix[i - 1][j - 1] + cost
                )

        return matrix[len(word1)][len(word2)]

if __name__ == "__main__":
    word1 = input("Введите первое слово: ")
    word2 = input("Введите второе слово: ")

    lev = Levenshtein()
    distance = lev.levenshtein_distance(word1, word2)
    print("Расстояние Левенштейна равно:", distance)
```

Set_1.py

```python
import unittest
from main import Levenshtein

new *
class TestLevenshtein(unittest.TestCase):
    new *
    def test_distance_equal_strings(self):
        lev= Levenshtein()
        self.assertEqual(lev.levenshtein_distance("kitten", "kitten"), second: 0)

    new *
    def test_distance_different_strings(self):
        lev= Levenshtein()
        self.assertEqual(lev.levenshtein_distance("kitten", "sitting"), second: 3)

    new *
    def test_distance_empty_string(self):
        lev= Levenshtein()
        self.assertEqual(lev.levenshtein_distance("", "abc"), second: 3)

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

Set_2.py

```python
from main import Levenshtein
import pytest

new *
@pytest.fixture
def lev():
    return Levenshtein()

new *
def test_distance_equal_strings(lev):
    assert lev.levenshtein_distance("kitten", "kitten") == 0

new *
def test_distance_different_strings(lev):
    assert lev.levenshtein_distance("kitten", "sitting") == 3

new *
def test_distance_empty_string(lev):
    assert lev.levenshtein_distance("", "abc") == 3

if __name__ == '__main__':
    pytest.main()
```

Результат работы

```
C:\Users\vikau\AppData\Local\Microsoft\Wi
Введите первое слово: fhdkkf
Введите второе слово: fdfue
Расстояние Левенштейна равно: 4

Process finished with exit code 0
```

```
C:\Users\vikau\AppData\Local\Microsoft\WindowsA
Введите первое слово: осминог
Введите второе слово: многоножкасороконожка
Расстояние Левенштейна равно: 17

Process finished with exit code 0
```

```
C:\Users\vikau\AppData\Local\Microsoft\Wi
Введите первое слово: тропинка
Введите второе слово: соринка
Расстояние Левенштейна равно: 3
```