Московский государственный технический университет им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №5
«Модульное тестирование в Python»

Выполнил:
Студент группы ИУ5-31Б
Попов А.С.
Подпись и дата:

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

Москва, 2025 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

field.py

```
def field(items, *args):
    assert len(args) > 0

    if len(args) == 1:
        key = args[0]
        for item in items:
            val = item.get(key)
            if val is not None:
                yield val
    else:
        for item in items:
            new_item = {key: item.get(key) for key in args if
item.get(key) is not None}
            if len(new_item) > 0:
                yield new_item
```

gen_random.py

```
import random

def gen_random(num_count, begin, end):
    for _ in range(num_count):
        yield random.randint(begin, end)
```

tests_tdd.py

```
import unittest
from field import field

class TestFieldFunction(unittest.TestCase):
    def setUp(self):
        self.goods = [
            {'title': 'Ковер', 'price': 2000, 'color': 'green'},
            {'title': 'Диван для отдыха', 'price': 5300, 'color':
'black'},
            {'title': 'Стеллаж', 'price': None, 'color': 'white'},
            {'title': None, 'price': 1000, 'color': 'red'}
        ]

    def test_single_argument(self):
        result = list(field(self.goods, 'title'))
        expected = ['Ковер', 'Диван для отдыха', 'Стеллаж']
        self.assertEqual(result, expected)
```

```
    def test_multiple_arguments(self):
        result = list(field(self.goods, 'title', 'price'))
        expected = [
            {'title': 'Ковер', 'price': 2000},
            {'title': 'Диван для отдыха', 'price': 5300},
            {'title': 'Стеллаж'},
            {'price': 1000}
        ]
        self.assertEqual(result, expected)

    def test_missing_key(self):
        result = list(field(self.goods, 'weight'))
        expected = []
        self.assertEqual(result, expected)

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

tests_mock.py

```
import unittest
from unittest.mock import patch
from gen_random import gen_random

class TestGenRandom(unittest.TestCase):
    @patch('gen_random.random.randint')
    def test_gen_random_mock(self, mock_randint):
        mock_randint.return_value = 5

        result = list(gen_random(3, 1, 10))

        self.assertEqual(result, [5, 5, 5])

        self.assertEqual(mock_randint.call_count, 3)

    @patch('gen_random.random.randint')
    def test_gen_random_sequence(self, mock_randint):
        mock_randint.side_effect = [10, 20, 30]

        result = list(gen_random(3, 1, 100))

        self.assertEqual(result, [10, 20, 30])

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

Скриншоты работы программы

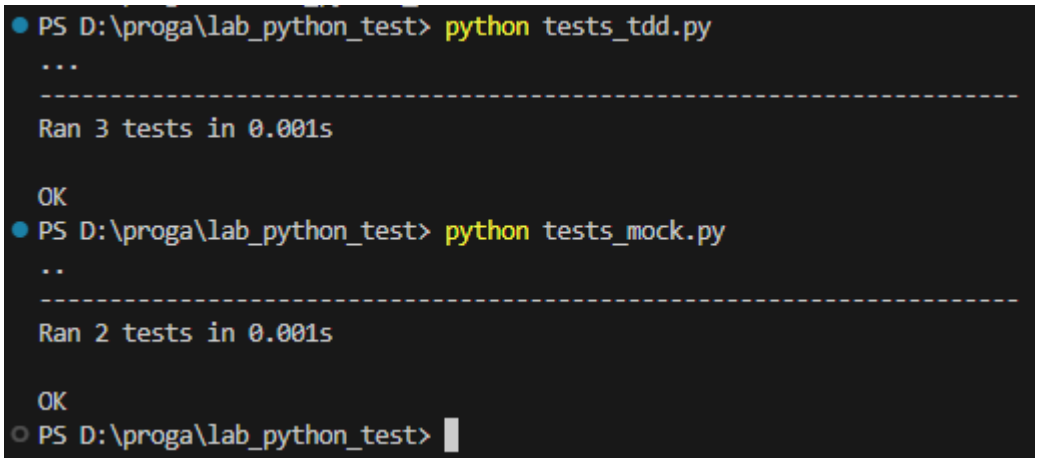

Рис. 1 – Работа приложения