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»

Курс «Базовые компоненты интернет-технологий»

Отчёт по лабораторной работе №5
«Модульное тестирование в Python.»

Выполнил:
студент группы ИУ5-33Б Кузнецов В. А.
подпись: , дата:
Проверил:
лектор Гапанюк Ю. Е.
подпись: , дата:

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).

Текст программы:

Field_test.py

```
import unittest
from lab_python_fp.field import field


class TestField(unittest.TestCase):
    def setUp(self):
        self.goods = [
           {'title': 'Ковер', 'price': 2000, 'color': 'green'},
           {'title': 'Диван для отдыха', 'price': 5300, 'color': 'black'}
        ]

    def test_one_argument(self):
        result = list(field(self.goods, 'title'))
        answer = ['Ковер', 'Диван для отдыха']
        self.assertEqual(result, answer)

    def test_many_arguments(self):
        result = list(field(self.goods, 'title', 'price'))
        answer = [{'title': 'Ковер', 'price': 2000}, {'title': 'Диван для
отдыха', 'price': 5300}]
        self.assertEqual(result, answer)


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

test_unique.py

```
import unittest
from lab_python_fp.unique import Unique
from lab_python_fp.gen_random import gen_random


class TestUnique(unittest.TestCase):
    def test_numbers(self):
        data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
        result = list(Unique(data))
        answer = [1, 2]
        self.assertEqual(result, answer)

    def test_random_generator(self):
        data = gen_random(10, 1, 3)
        result = set(Unique(data))
        answer = set(range(1, 4))
```

```
        self.assertTrue(answer.issubset(result))

    def test_letters(self):
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        result = list(Unique(data))
        answer = ['a', 'A', 'b', 'B']
        self.assertEqual(result, answer)

    def test_letters_ignoring_case(self):
        data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
        result = list(Unique(data, ignore_case=True))
        answer = ['a', 'b']
        self.assertEqual(result, answer)


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

test_gen_random.feature

```
Feature: Testing generator gen_random
  Scenario: Testing a range with positive boundaries
    Given amount of numbers 10 and minimum 1 and maximum 10
    When we generate numbers
    Then we have to get the specified number of numbers in a given range

  Scenario: Testing a range with negative boundaries
    Given amount of numbers 10 and minimum -10 and maximum -1
    When we generate numbers
    Then we have to get the specified number of numbers in a given range

  Scenario: Testing a range with positive and negative boundaries
    Given amount of numbers 10 and minimum -10 and maximum 10
    When we generate numbers
    Then we have to get the specified number of numbers in a given range
```

test_gen_random.py

```
from behave import Given, When, Then
from lab_python_fp.gen_random import gen_random


@Given('amount of numbers {amount} and minimum {min_} and maximum {max_}')
def step_impl(context, amount, min_, max_):
    context.amount = int(amount)
    context.min = int(min_)
    context.max = int(max_)
    context.generator = gen_random(context.amount, context.min, context.max)


@When('we generate numbers')
def step_impl(context):
    context.numbers = list(context.generator)


@Then('we have to get the specified number of numbers in a given range')
def step_impl(context):
    assert len(context.numbers) == context.amount
    assert all(context.min <= i <= context.max for i in context.numbers)
```

Пример выполнения

```
Launching unittests with arguments python -m unittest C:\Users\kuzva\PycharmProjects\BKIT\tests\field_test.py in
 C:\Users\kuzva\PycharmProjects\BKIT\tests



Ran 2 tests in 0.002s

OK

Process finished with exit code 0
```

```
Launching unittests with arguments python -m unittest C:\Users\kuzva\PycharmProjects\BKIT\tests\test_unique.py in
 C:\Users\kuzva\PycharmProjects\BKIT\tests



Ran 4 tests in 0.002s

OK

Process finished with exit code 0
```

```
(venv) PS C:\Users\kuzva\PycharmProjects\BKIT> behave tests/features
Feature: Testing generator gen_random # tests/features/test_gen_random.feature:1

  Scenario: Testing a range with positive boundaries                    # tests/features/test_gen_random.feature:2
    Given amount of numbers 10 and minimum 1 and maximum 10             # tests/features/steps/test_gen_random.py:5
    When we generate numbers                                            # tests/features/steps/test_gen_random.py:13
    Then we have to get the specified number of numbers in a given range # tests/features/steps/test_gen_random.py:18

  Scenario: Testing a range with negative boundaries                    # tests/features/test_gen_random.feature:7
    Given amount of numbers 10 and minimum -10 and maximum -1           # tests/features/steps/test_gen_random.py:5
    When we generate numbers                                            # tests/features/steps/test_gen_random.py:13
    Then we have to get the specified number of numbers in a given range # tests/features/steps/test_gen_random.py:18

  Scenario: Testing a range with positive and negative boundaries       # tests/features/test_gen_random.feature:12
    Given amount of numbers 10 and minimum -10 and maximum 10           # tests/features/steps/test_gen_random.py:5
    When we generate numbers                                            # tests/features/steps/test_gen_random.py:13
    Then we have to get the specified number of numbers in a given range # tests/features/steps/test_gen_random.py:18

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
9 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.004s
```