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:
студент группы ИУ5-32Б:
Бекетов Роман Александрович
Подпись и дата:

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

Москва, 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).

Листинг

test_check_mock.py

```python
import unittest
import unittest.mock

import sys
sys.path.append('../')
from nums import evens, check_evens_len

class TestMock(unittest.TestCase):

    @unittest.mock.patch("nums.evens")
    def test_check_mock(self, evens_mock):

        evens_mock.return_value = []
        self.assertEqual(check_evens_len([2, 4, 6]), [])

        self.assertTrue(evens_mock.called)
```

test_unique.py

```python
import unittest

import sys
sys.path.append('../')
from unique import Unique

class TestUnique(unittest.TestCase):

    def test_unique_any(self):
        self.assertEqual(list(Unique(["F", "f", "f", "F"])), ["F", "f"])

    def test_unique_ignore_case(self):
        self.assertEqual(list(Unique(["F", "f", "f", "F"], bool_ignore_case = True)), ["f"])

    def test_empty(self):
        self.assertEqual(list(Unique([])), [])
```

test_unique.py (radish)

```python
from radish import given, when, then, custom_type, register_custom_type, TypeBuilder
import os
import sys
sys.path.append(os.getcwd())
from unique import Unique

@custom_type('Number', r'\d*')
def parse_number(text):
    if text.isdigit():
        return int(text)
    elif text == "None":
        return None
    return text

# register the NumberList type
register_custom_type(NumberList=TypeBuilder.with_many0(
    parse_number, listsep=','))

@given("the list {test_list:NumberList}")
def have_list(step, test_list):
    step.context.test_list = test_list

@when("Unique them")
def unique_them(step):
    step.context.result = list(Unique(step.context.test_list))

@then("result to be {result:NumberList}")
def expect_result(step, result):
    assert step.context.result == result
```

unique_f.feature

```
Feature: test Unique

    Scenario: Test my class Unique (any)
        Given the list F, f
        When Unique them
        Then result to be f

    Scenario: Test my class Unique none
        Given the list None
        When Unique them
        Then result to be None

    Scenario: Test my class Unique digit
        Given the list 3, 3, 6
        When Unique them
        Then result to be 3, 6
```

Тесты

```
(.venv) romanbeketov@MacBook-Air-Roman src % radish unique_f.feature
Feature: test Unique  # unique_f.feature

    Scenario: Test my class Unique (any)
        Given the list F, f
        When Unique them
        Then result to be f

    Scenario: Test my class Unique none
        Given the list None
        When Unique them
        Then result to be None

    Scenario: Test my class Unique digit
        Given the list 3, 3, 6
        When Unique them
        Then result to be 3, 6

1 features (1 passed)
3 scenarios (3 passed)
9 steps (9 passed)
Run 1670504691 finished within a moment
```

```
(.venv) romanbeketov@MacBook-Air-Roman test_unittest % python -m unittest test_check_mock.py test_unique.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.000s

OK
```