Московский Государственный Технический Университет им. Баумана

Отчет по лабораторной работе №5
«Модульное тестирование»
По дисциплине «Базовые компоненты интернет технологий»

Выполнила:
Студентка группы ИУ5-32Б
Алцыбеева Маргарита
01.12.2022

Проверил:
Преподаватель кафедры ИУ5
Гапанюк Ю. Е.

Москва, 2022 г.

Задание:

Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.

Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.

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:

Файл main.py

```
import math

def square(_radius):
    if type(_radius) not in [float, int]:
        raise TypeError('Радиус должен быть числом')
    if _radius < 0:
        raise ValueError('Радиус не может быть отрицательным')
    print(math.pi * (1 ** 2))
    return math.pi * (_radius ** 2)
```

Файл test_tdd.py

```
from math import pi
import unittest
from main import square

class Curcle_test(unittest.TestCase):

    def test_square(self):
        self.assertEqual(square(0), 0)
        self.assertEqual(square(3), pi*3**2)
        self.assertEqual(square(1), pi)

    def test_values(self):
        self.assertRaises(ValueError, square, -7)
        self.assertRaises(ValueError, square, -500)
    def test_type(self):
        self.assertRaises(TypeError, square, 'qwe')
        self.assertRaises(TypeError, square, [123])
        self.assertRaises(TypeError, square, {123: -9})
        self.assertRaises(TypeError, square, 1+8j)
        self.assertRaises(TypeError, square, [4, 90])
        self.assertRaises(TypeError, square, False)

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

Файл test_bdd.py

```
        from radish import given, then, when
from curcle import square

@given("I have the radius {radius:g}")
def have_radius(step, radius):
    step.context.radius = radius

@when("I counting the square")
def square_curcle(step):
    step.context.result = square(step.context.radius)

@then("I expect the result to be {result:g}")
def expext_result(step, result):
    assert step.context.result == result
```

Файл feauture.feauture

```
Feature: Testing the Equation
  Scenario: test1
    Given I have the radius 0
    When I counting the square
    Then I expect the result to be 0

  Scenario: Test2
    Given I have the radius 1
    When I counting the square
    Then I expect the result to be pi

  Scenario: Test3
    Given I have the radius 3
    When I counting the square
    Then I expect the result to be pi*3**2
```

Примеры работы программы: