Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №3
«Модульное тестирование в Python»

Выполнил:
студент группы ИУ5-31Б
Шилина А.Ю.
Подпись и дата: 18.12.24

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

ОГЛАВЛЕНИЕ

1. ОПИСАНИЕ ЗАДАНИЯ

Задание: Необходимо выбрать любой ранее написанный код и модифицировать его для применения модульного тестирования с использованием двух подходов: TDD (Test-Driven Development) и BDD (Behavior-Driven Development).

- Код был взять из лабораторной работы №1 (биквадратное уравнение.
- Для подхода BDD был выбран язык описания поведения системы - *gherkin*.

2. ТЕКСТ ПРОГРАММЫ

```
3. import math


   def get_coefficient(value):
       try:
           return float(value)
       except ValueError:
           raise ValueError(f"Ошибка: коэффициент должен быть числом.
   Введенное значение: {value}")


   def korni(t):
       if t < 0:
           return []
       x1 = math.sqrt(t)
       if x1==-x1:
           return[x1]
       else:
           return [x1, -x1]


   def reshenie(a, b, c):
       if a == 0:
           raise ValueError("Коэффициент A не может быть равен 0 для
   биквадратного уравнения.")

       discriminant = b ** 2 - 4 * a * c
       if discriminant < 0:
           return []

       sqrt_d = math.sqrt(discriminant)
       t1 = (-b + sqrt_d) / (2 * a)
       t2 = (-b - sqrt_d) / (2 * a)

       roots = set()
       for t in [t1, t2]:
           roots.update(korni(t))

       return sorted(roots)


   # Модульные тесты с использованием TDD
   import unittest
```

```
class TestBiquadraticSolver(unittest.TestCase):
"""TDD"""
    def test_discriminant_menshe0(self):
        self.assertEqual(reshenie(1, 0, 1), [])

    def test_2otveta(self):
        self.assertEqual(reshenie(1, -2, 1), [-1.0, 1.0])

    def test_4otveta(self):
        self.assertEqual(reshenie(1, -17, 16), [-4.0,-1.0, 1.0, 4.0])

    def test_a_0(self):
        with self.assertRaises(ValueError):
            reshenie(0, 1, 1)

    def test_korni_otric(self):
        self.assertEqual(korni(-1), [])

    def test_korni_polozhit(self):
        self.assertEqual(korni(4), [2.0, -2.0])
"""BDD gherkin"""
# Feature: Решение квадратного уравнения и извлечение корней
#
#   Scenario: Дискриминант меньше нуля
#     Given коэффициенты уравнения a = 1, b = 0, c = 1
#     When решается уравнение
#     Then результатом является пустой список []
#
#   Scenario: Уравнение имеет два одинаковых корня
#     Given коэффициенты уравнения a = 1, b = -2, c = 1
#     When решается уравнение
#     Then результатом являются корни [-1.0, 1.0]
#
#   Scenario: Уравнение имеет четыре решения
#     Given коэффициенты уравнения a = 1, b = -17, c = 16
#     When решается уравнение
#     Then результатом являются корни [-4.0, -1.0, 1.0, 4.0]
#
#   Scenario: Коэффициент a равен 0
#     Given коэффициенты уравнения a = 0, b = 1, c = 1
#     When решается уравнение
#     Then возникает исключение ValueError
#
#   Scenario: Извлечение корней из отрицательного числа
#     Given число для извлечения корней x = -1
#     When извлекаются корни
#     Then результатом является пустой список []
#
#   Scenario: Извлечение корней из положительного числа
#     Given число для извлечения корней x = 4
#     When извлекаются корни
#     Then результатом являются корни [2.0, -2.0]


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

4. ТЕСТИРОВАНИЕ