Защищено:
Гапанюк Ю.Е.

"__"_____**2025** г.

Демонстрация:
Константинов А.А.

"__"_____**2025** г.

Отчет по лабораторной работе № 4 по курсу Парадигмы и конструкции языков программирования

Тема работы: "Модульное тестирование в Python"

5
(количество листов)

студент группы ИУ5Ц-53Б _____
(подпись)

Константинов А.А. "__"_____**2025** г.

Москва, МГТУ - **2025**

1. **Описание задания**

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).

2. Листинг программы

```
import math
import unittest

# ЛОГИКА (на основе ЛР-1)
def solve_biquadratic(a, b, c):
    """
    Решает биквадратное уравнение A*x^4 + B*x^2 + C = 0
    Возвращает список действительных корней
    """
    if a == 0:
        raise ValueError("A не может быть равно 0")
    D = b ** 2 - 4 * a * c
    if D < 0:
        return []

    y1 = (-b + math.sqrt(D)) / (2 * a)
    y2 = (-b - math.sqrt(D)) / (2 * a)

    roots = []
    for y in (y1, y2):
        if y > 0:
            roots.extend([math.sqrt(y), -math.sqrt(y)])
        elif y == 0:
            roots.append(0.0)

    # Убираем дубликаты (важно для x = 0)
    return sorted(set(roots))

# TDD
class TestSolveBiquadraticTDD(unittest.TestCase):

    def test_four_roots(self):
        # x^4 - 5x^2 + 4 = 0 → ±1, ±2
        self.assertEqual(
            solve_biquadratic(1, -5, 4),
            [-2.0, -1.0, 1.0, 2.0]
        )

    def test_no_real_roots(self):
        # x^4 + x^2 + 1 = 0
        self.assertEqual(
            solve_biquadratic(1, 1, 1),
            []
        )

    def test_a_zero_exception(self):
        with self.assertRaises(ValueError):
            solve_biquadratic(0, 2, 1)

# BDD — Given / When / Then
class BiquadraticBDDTest(unittest.TestCase):
```

```
    def given_equation(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def when_solving(self):
        self.result = solve_biquadratic(self.a, self.b, self.c)

    def then_result_should_be(self, expected):
        self.assertEqual(self.result, expected)

    def test_scenario_1(self):
        # Given
        self.given_equation(1, -5, 4)
        # When
        self.when_solving()
        # Then
        self.then_result_should_be([-2.0, -1.0, 1.0, 2.0])

    def test_scenario_2(self):
        # Given
        self.given_equation(1, 1, 1)
        # When
        self.when_solving()
        # Then
        self.then_result_should_be([])

    def test_scenario_3(self):
        # Given
        self.given_equation(1, 0, 0)
        # When
        self.when_solving()
        # Then
        self.then_result_should_be([0.0])

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

3. **Результаты работы программы**