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» Кафедра «Системы обработки информации и управления»

Лабораторная работа №5

по предмету

«Базовые компоненты интернет-технологий»

Выполнил:

студент группы ИУ5-35Б

Клементьев Артем

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

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).

Код программы

Программа состоит из нескольких файлов.

main.py

```
import sys
import math


def get_coef(index, prompt):
    flag = 0
    while flag == 0:
        try:
            coef_str = sys.argv[index]
        except:
            print(prompt)
            coef_str = input()
        try:
            coef = float(coef_str)
            flag = 1
        except ValueError:
            print('Вы ввели не число! Попробуйте снова')
    return coef


def get_roots(a, b, c):
    result = []
    D = b * b - 4 * a * c
    if a == 0.0 and b != 0.0:
        if -c / b >= 0.0:
            root = math.sqrt(-c/b)
            if root != 0.0:
                result.append(-root)
                result.append(root)
            else:
                result.append(abs(root))
    elif a == 0.0 and b == 0.0:
        if c == 0.0:
            print('Бесконечное множество корней')
            exit(1)
        else:
            print('Нет корней')
            exit(1)
    elif D == 0.0:
        if -b / (2.0 * a) >= 0.0:
            root = math.sqrt(-b / (2.0 * a))
            if root != 0.0:
                result.append(root)
                result.append(-root)
            else:
                result.append(abs(root))
    elif D > 0.0:
```

```
        if (-b + math.sqrt(D)) / (2.0 * a) >= 0.0:
            root1 = math.sqrt((-b + math.sqrt(D)) / (2.0 * a))
            if root1 != 0.0:
                result.append(root1)
                result.append(-root1)
            else:
                result.append(abs(root1))
        if (-b - math.sqrt(D)) / (2.0 * a) >= 0.0:
            root2 = math.sqrt((-b - math.sqrt(D)) / (2.0 * a))
            if root2 != 0.0:
                result.append(root2)
                result.append(-root2)
            else:
                result.append(abs(root2))
    return result


def main():
    a = get_coef(1, 'Введите коэффициент A:')
    b = get_coef(2, 'Введите коэффициент B:')
    c = get_coef(3, 'Введите коэффициент C:')
    roots = get_roots(a, b, c)
    if len(roots) == 0:
        print('Нет корней')
    elif len(roots) == 1:
        print('Один корень: {}'.format(roots[0]))
    elif len(roots) == 2:
        print('Два корня: {} и {}'.format(roots[0], roots[1]))
    elif len(roots) == 3:
        print('Три корня: {}, {} и {}'.format(roots[0], roots[1], roots[2]))
    elif len(roots) == 4:
        print('Четыре корня: {}, {}, {} и {}'.format(roots[0], roots[1], roots[2],
roots[3]))


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

Testing.py

```
from math import *
from main import *
import unittest
from behave import Given, Then, When

class Test_get_rootsTTD(unittest.TestCase):
    def test1(self):
        self.assertEqual(get_roots(1, -4, 4), [sqrt(2), -sqrt(2)])

    def test2(self):
        self.assertEqual(get_roots(9, -10, 1), [1, -1, 1/3, -1/3])

    def test3(self):
        self.assertEqual(get_roots(0, 1, -4), [-2, 2])

    def test4(self):
        self.assertEqual(get_roots(1, 11, 10), [])


@Given("Data for equation")
def test1():
    print("Data for equation")
    print(f"Data: {[9, -10, 1]}")
    print(f"Data: {[1, -4, 4]}")
```

```python
@When("I want to solve the equation")
def test2():
    print("I want to solve the equation")


@Then("I get roots")
def test3():
    print("I get roots")
    assert get_roots(9, -10, 1) == [1, -1, 1/3, -1/3]
    assert get_roots(1, -4, 4) == [sqrt(2), -sqrt(2)]


def main():
    unittest.main()


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

f.feature

```
Feature: Testing

  Scenario: 4 roots
    Given Data for equation

    When I want to solve the equation

    Then I get roots
```

Результаты выполнения