Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по лабораторной работе №5

Выполнил:
студент группы ИУ5-34Б
Мкртчян Давид
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:

o TDD - фреймворк (не менее 3 тестов).

o BDD - фреймворк (не менее 3 тестов).

o Создание Mock-объектов (необязательное дополнительное задание).

Текст программы

Main.py

```
import sys
import math


def get_coef(index, prompt):
    '''
    Читаем коэффициент из командной строки или вводим с клавиатуры
    Args:
        index (int): Номер параметра в командной строке
        prompt (str): Приглашение для ввода коэффицента
    Returns:
        float: Коэффициент квадратного уравнения
    '''
    try:
        # Пробуем прочитать коэффициент из командной строки
        coef_str = sys.argv[index]
    except:
        # Вводим с клавиатуры
        print(prompt)
        coef_str = input()
    # Переводим строку в действительное число
        while True:
            try:
                float(coef_str)
                break
            except:
                print('Ошибка, введите число')
                coef_str = input()
    coef = float(coef_str)
    return coef


def get_roots(a, b, c):
    '''
    Вычисление корней квадратного уравнения
    Args:
```

```
        a (float): коэффициент A
        b (float): коэффициент B
        c (float): коэффициент C
    Returns:
        list[float]: Список корней
    '''
    result = []
    D = b * b - 4 * a * c
    #print(D)
    if D == 0.0:
        root = -b / (2.0 * a)
        if root > 0:
            result.append(math.sqrt(root))
            result.append(-math.sqrt(root))
        elif root == 0:
            result.append(0)
    elif D > 0.0:
        sqD = math.sqrt(D)
        root1 = (-b + sqD) / (2.0 * a)
        root2 = (-b - sqD) / (2.0 * a)
        if root1 > 0:
            result.append(math.sqrt(root1))
            result.append(-math.sqrt(root1))
        elif root1 == 0:
            result.append(root1)
        if root2 > 0:
            result.append(math.sqrt(root2))
            result.append(-math.sqrt(root2))
        elif root2 == 0:
            result.append(math.fabs(root2))
    result = sorted(result)
    return result


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


# Если сценарий запущен из командной строки
if __name__ == "__main__":
    main()
```

BDD_test

```
from ast import Try
from behave import *
import sys
sys.path.append("../../")
from lab1 import get_roots


@given(u'{given_a} coef a, coef b is {given_b} and c is {given_c}')
def step_impl(context, given_a, given_b, given_c):
    global a
    global b
    global c
    a = int(given_a)
    b = int(given_b)
    c = int(given_c)
    return True


@When("starting function")
def step_impl(context):
    global result
    result = get_roots(a, b, c)
    # return True
    if type(a) == int:
        return True

@Then("we shoud see {given_result}")
def step_impl(context, given_result):
    try:
        assert(result == given_result)
        return True
    except:
        return False
```

TDD_test

```
import unittest
from lab1 import get_roots

class SquareEqSolverTestCase(unittest.TestCase):
   def test_no_root(self):
       res = get_roots(1, 11, 10)
       self.assertEqual(len(res), 0)

   def test_single_root(self):
       res = get_roots(10, 0, 0)
       self.assertEqual(len(res), 1)
       self.assertEqual(res, [0])

   def test_two_roots(self):
       res = get_roots(1, -2, -8)
       self.assertEqual(len(res), 2)
       self.assertEqual(res, [-2, 2])

   def test_three_roots(self):
       res = get_roots(-4, 16, 0)
       self.assertEqual(len(res), 3)
       self.assertEqual(res, [-2, 0, 2])

   def test_four_roots(self):
       res = get_roots(1, -10, 9)
```

```
        self.assertEqual(len(res), 4)
        self.assertEqual(res, [-3, -1, 1, 3])
```

Анализ результатов

```
Feature: testing roots # tests2.feature:1

  Scenario Outline: multiple roots roots -- @1.1   # tests2.feature:10
    Given 1 coef a, coef b is 10 and c is 11       # steps/test.py:8
    When starting function                         # steps/test.py:19
    Then we shoud see "[]"                         # steps/test.py:27

  Scenario Outline: multiple roots roots -- @1.2   # tests2.feature:11
    Given 10 coef a, coef b is 0 and c is 0        # steps/test.py:8
    When starting function                         # steps/test.py:19
    Then we shoud see "[0]"                        # steps/test.py:27

  Scenario Outline: multiple roots roots -- @1.3   # tests2.feature:12
    Given 1 coef a, coef b is -2 and c is -8       # steps/test.py:8
    When starting function                         # steps/test.py:19
    Then we shoud see "[-2, 2]"                    # steps/test.py:27

  Scenario Outline: multiple roots roots -- @1.4   # tests2.feature:13
    Given 4 coef a, coef b is 16 and c is 0        # steps/test.py:8
    When starting function                         # steps/test.py:19
    Then we shoud see "[-2, 0, 2]"                 # steps/test.py:27

  Scenario Outline: multiple roots roots -- @1.5   # tests2.feature:14
    Given 1 coef a, coef b is -10 and c is 9       # steps/test.py:8
    When starting function                         # steps/test.py:19
    Then we shoud see "[-3, 1, 1, 3]"              # steps/test.py:27

1 feature passed, 0 failed, 0 skipped
5 scenarios passed, 0 failed, 0 skipped
15 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.018s
```

```
Testing started at 13:03 ...
============================= test session starts ==============================
collecting ... collected 5 items

test.py::SquareEqSolverTestCase::test_four_roots PASSED                  [ 20%]
test.py::SquareEqSolverTestCase::test_no_root PASSED                     [ 40%]
test.py::SquareEqSolverTestCase::test_single_root PASSED                 [ 60%]
test.py::SquareEqSolverTestCase::test_three_roots PASSED                 [ 80%]
test.py::SquareEqSolverTestCase::test_two_roots PASSED                   [100%]

============================== 5 passed in 0.04s ===============================
```