**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: |  | Проверил: |
| студент группы ИУ5-31Б |  | преподаватель каф. ИУ5 |
| Койбаев Тамерлан |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | 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 А  
 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 А:')  
 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**



