** 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

**«Модульное тестирование в Python»**

по предмету

«Базовые компоненты интернет-технологий»

Выполнил:

студент группы № ИУ5-31Б

Искорнев Иван

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

2023 г.

Постановка задачи

* Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
* Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
* Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
  + TDD - фреймворк (не менее 3 тестов).
  + BDD - фреймворк (не менее 3 тестов).
  + Создание Mock-объектов (необязательное дополнительное задание).

Задание

В качестве объекта для тестирования была выбрана программа, находящая корни биквадратного уравнения.

 Код программы на GitHub.

Текст программы

**equation.py**  
  
*import* math  
*import* sys  
  
  
*def* calculate(A, B, C):  
 *if* type(A) *not in* [int, float]:  
 *raise* TypeError("Коэффициент A должен быть положительным вещественным числом!")  
 *if* type(B) *not in* [int, float]:  
 *raise* TypeError("Коэффициент B должен быть неотрицательным вещественным числом!")  
 *if* type(C) *not in* [int, float]:  
 *raise* TypeError("Коэффициент C должен быть неотрицательным вещественным числом!")  
  
 *if* A == 0:  
 *raise* ValueError("Коэффициент A должен быть положительным вещественным числом!")  
  
 D = B \* B - 4 \* A \* C  
  
 *if* D > 0:  
 t = (-B - math.sqrt(D)) / (2 \* A)  
 *if* t > 0:  
 x1 = math.sqrt((-B + math.sqrt(D)) / (2 \* A))  
 x2 = -x1  
 x3 = math.sqrt((-B - math.sqrt(D)) / (2 \* A))  
 x4 = -x3  
 *return* tuple(sorted(set([x1, x2, x3, x4])))  
 *elif* D == 0:  
 x1 = math.sqrt(-B / 2 \* A)  
 x2 = -x1  
 *return* tuple(sorted(set([x1, x2])))  
  
 *return* ()

**test\_equation.py**  
  
*import* unittest  
*from* equation *import* calculate  
  
  
*class* TestEquation(unittest.TestCase):  
  
 *def* test\_calculate(self):  
 self.assertEqual(calculate(1, -10, 9), (-3, -1, 1, 3))  
 self.assertEqual(calculate(-4, 16, 0), (-2, 0, 2))  
 self.assertEqual(calculate(431, -123, 665), ())  
  
 *def* test\_value(self):  
 *with* self.assertRaises(ValueError) *as* e:  
 calculate(0, 33, 9)  
  
 *def* test\_type(self):  
 *with* (self.assertRaises(TypeError)) *as* e:  
 calculate(12, "B", 4)  
  
  
*if* \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

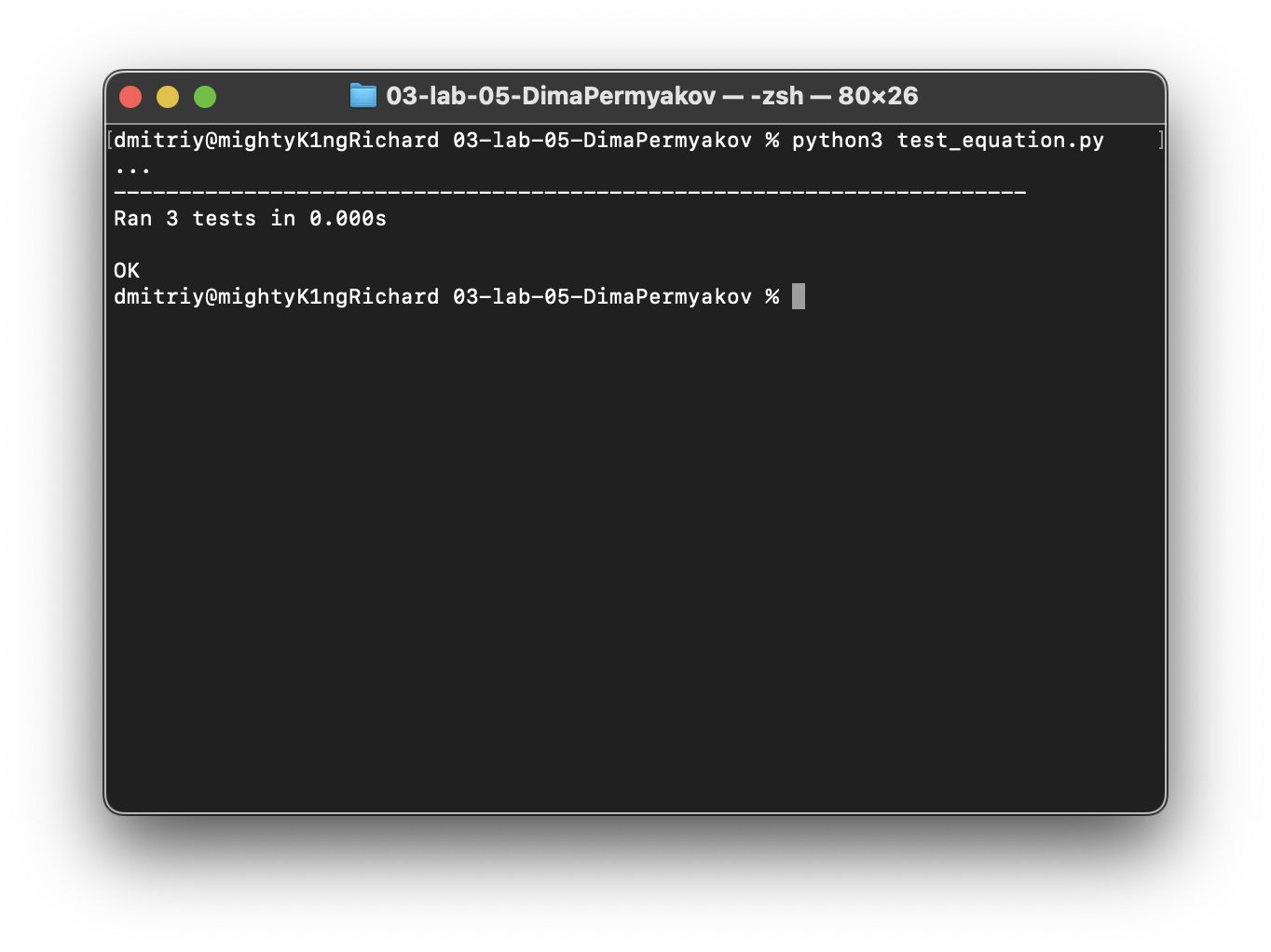
**tests.py**  
*from* behave *import* Given, When, Then  
*from* equation *import* calculate  
  
  
@Given("equation with coef A {A} B {B} C {C}")  
*def* given\_increment(context, A: str, B: str, C: str):  
 context.A = int(A)  
 context.B = int(B)  
 context.C = int(C)  
  
  
@When("we calculate {object}")  
*def* given\_increment(context, object: str):  
 context.results = calculate(context.A, context.B, context.C)  
  
  
@Then("we should see root1 {root1} root2 {root2} root3 {root3} root4 {root4}")  
*def* then\_results(context, root1: str, root2: str, root3: str, root4: str):  
 *if* root1 == "empty":  
 *assert* (context.results == ())  
 *elif* root2 == "empty":  
 *assert* (context.results == (int(root1)))  
 *elif* root3 == "empty":  
 *assert* (context.results == (int(root1), int(root2)))  
 *elif* root4 == "empty":  
 *assert* (context.results == (int(root1), int(root2), int(root3)))  
 *else*:  
 *assert* (context.results == (int(root1), int(root2), int(root3), int(root4)))

**tests.feature**

*Feature*: Testing the Equation  
 *Scenario*: Test calculate 4 roots  
 *Given* equation with coef A 1 B -10 C 9  
 *When* we calculate roots  
 *Then* we should see root1 -3 root2 -1 root3 1 root4 3  
  
 *Scenario*: Test calculate 3 roots  
 *Given* equation with coef A -4 B 16 C 0  
 *When* we calculate roots  
 *Then* we should see root1 -2 root2 0 root3 2 root4 empty  
  
 *Scenario*: Test calculate 0 roots  
 *Given* equation with coef A 431 B -123 C 665  
 *When* we calculate roots  
 *Then* we should see root1 empty root2 empty root3 empty root4 empty

Анализ результатов

В качестве TDD – фреймворка был использован пакет unittest

****

В качестве BDD – фреймворка был использован пакет behave

