

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Тестирование программного обеспечения

Отчет

Лабораторная работа № 1

Выполнила:  
Холод В. Д.  
Группа К3322

Проверил:  
Кочубеев Н.С

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ .....	3
1  Ход работы.....	4
1.1  Выбор репозитория Github .....	4
1.2  Анализ функциональности приложения .....	4
1.3  Написание тестов .....	4
1.4  Отчет по результатам тестирования .....	6
ЗАКЛЮЧЕНИЕ .....	7

## ВВЕДЕНИЕ

### Задачи:

1. Создать модульные тесты для выбранных компонентов системы.
2. Протестировать несколько сценариев работы, включая граничные случаи.
3. Минимум 5 тестов должны быть написаны для разных функциональных частей приложения.
4. Тесты должны быть написаны с использованием AAA (arrange, act, assert) и FIRST (fast, isolated, repeatable, self-validating, timely) principles.

**Цель работы:** практическое освоение этапов разработки тестирования и написание Unit тестов.

# 1 Ход работы

## 1.1 Выбор репозитория Github

Для тестирования был выбран следующий репозиторий - <https://github.com/IMREYZ/Calculator>

## 1.2 Анализ функциональности приложения

Основная функциональность:

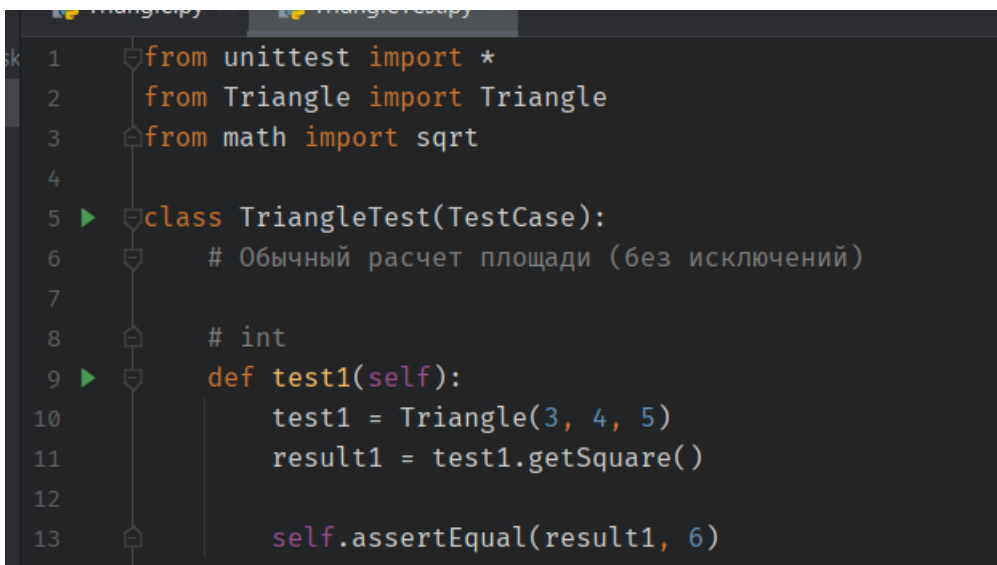
Класс Triangle имеет 3 поля – это стороны a, b и c. Основной функцией класса является вычитывание площади треугольника – функция getSquare. Перед подсчетом площади в функции getSquare происходит проверка на исключения функцией exception – происходит проверка того, что стороны положительны, стороны валидны и что треугольник существует.

## 1.3 Написание тестов

Мы создадим 5 тестов с использованием AAA и FIRST principles., охватывающих различные части функциональности. Будем использовать unittest для модульного тестирования

### 1. Обычный расчет площади (без исключений)

#### 1.1 Подсчет площади с целыми числами



```
1 from unittest import *
2 from Triangle import Triangle
3 from math import sqrt
4
5 class TriangleTest(TestCase):
6     # Обычный расчет площади (без исключений)
7
8     # int
9     def test1(self):
10         test1 = Triangle(3, 4, 5)
11         result1 = test1.getSquare()
12
13         self.assertEqual(result1, 6)
```

## 1.2 Подсчет чисел с типом float

```
# float
def test2(self):
    test1 = Triangle(2.5, 2.5, sqrt(12.5))
    result1 = test1.getSquare()

    self.assertEqual(result1, 3.125)

#float
def test3(self):
    test1 = Triangle(sqrt(4), sqrt(99), sqrt(103))
    result1 = test1.getSquare()

    self.assertEqual(result1, sqrt(99))
```

## 2. Расчет площади с исключениями

### 2.1 Сторона должны быть корректными

```
35
36 # Первая сторона некорректная!
37 def test4(self):
38     test4 = Triangle('6', 8, 3)
39
40     with self.assertRaisesRegex(TypeError, 'Первая сторона некорректная!'):
41         test4.getSquare()
42
43
44 # Вторая сторона некорректная!
45 def test5(self):
46     test5 = Triangle(6, '8', 3)
47
48     with self.assertRaisesRegex(TypeError, 'Вторая сторона некорректная!'):
49         test5.getSquare()
50
51
52 # Третья сторона некорректная!
53 def test6(self):
54     test6 = Triangle(6, 8, '3')
55
56     with self.assertRaisesRegex(TypeError, 'Третья сторона некорректная!'):
57         test6.getSquare()
```

## 2.2 Сторона треугольника должна быть положительной

```
60 # Первая сторона должна быть положительной!
61 def test7(self):
62     test7 = Triangle(-6, 8, 3)
63
64     with self.assertRaisesRegex(ValueError, 'Первая сторона должна быть положительной!'):
65         test7.getSquare()
66
67
68 # Вторая сторона должна быть положительной!
69 def test8(self):
70     test8 = Triangle(6, -8, 3)
71
72     with self.assertRaisesRegex(ValueError, 'Вторая сторона должна быть положительной!'):
73         test8.getSquare()
74
75
76 # Третья сторона должна быть положительной!
77 def test9(self):
78     test9 = Triangle(6, 8, -3)
79
80     with self.assertRaisesRegex(ValueError, 'Третья сторона должна быть положительной!'):
81         test9.getSquare()
```

## 2.3 Проверка несуществующего треугольника ( $5 > 1 + 2$ )

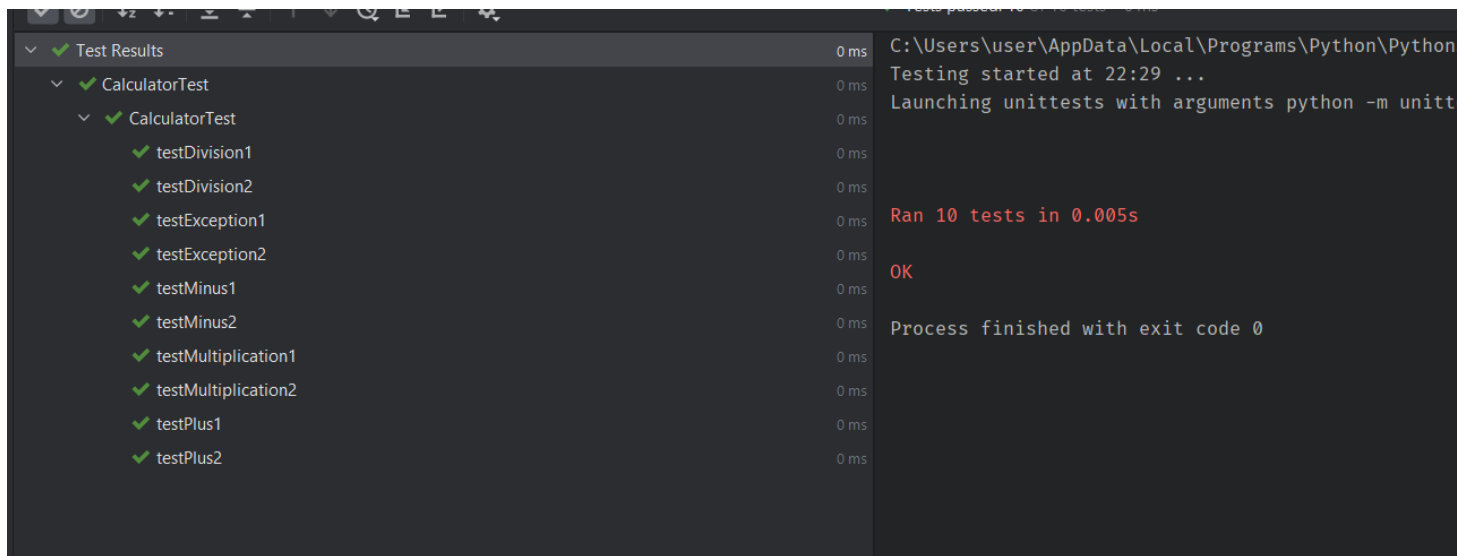
```
# Такого треугольника не существует!
def test10(self):
    test10 = Triangle(1, 2, 5)

    with self.assertRaisesRegex(ValueError, 'Такого треугольника не существует!'):
        test10.getSquare()
```

## 1.4 Отчет по результатам тестирования

Функция getSquare была протестирована всеми возможными случаями:

- 1) Валидный расчет площади
- 2) Невалидная сторона
- 3) Неположительная сторона
- 4) Несуществующий треугольник



## **ЗАКЛЮЧЕНИЕ**

Проведенные тесты обеспечивают хорошее покрытие кода, проверяя ключевые функциональные элементы программы. Все основные сценарии работы программы протестированы и соответствуют ожидаемым результатам.