**Министр науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

Факультет информационных технологий и программирования

Лабораторная работа №4

*Написание Unit тестов.*

**Выполнила студентка группы № M3307**

Старцева Арина Михайловна

**Подпись:**

**Проверил:**

Повышев Владислав Вячеславович

Санкт-Петербург

2024

**План тестирования**

1. Цели и задачи тестирования:

- проверка правильности работы функций вычисления площади и периметра для различных геометрических фигур

- необходимо протестировать функции на корректных данных, граничных значениях (нулевые значения, отрицательные числа) и некорректных данных (строки, None)

1. Описание тестируемого продукта:

Набор функций для расчёта площади и периметра различных фигур: круга, прямоугольника, квадрата и треугольника

1. Область тестирования:

Функции для расчёта площади и периметра из следующих модулей:

- circle.py: area(r) и perimeter(r)

- rectangle.py: area(a, b) и perimeter(a, b)

- square.py: area(a) и perimeter(a)

- triangle.py: area(a, h) и perimeter(a, b, c)

1. Стратегия тестирования:

Для каждой функции будут созданы тесты с корректными значениями, граничными случаями и некорректными данными. Все тесты будут выполнены с использованием библиотеки unittest

1. Критерии приемки:

Тесты считаются пройденными, если они возвращают ожидаемые результаты без ошибок и исключений. Для каждого модуля должны быть пройдены все предназначенные тесты

1. Ожидаемые результаты:

Все тесты должны успешно выполнить расчёты площади и периметра для корректных данных, а также корректно обрабатывать граничные и некорректные входные данные

**Приложение**

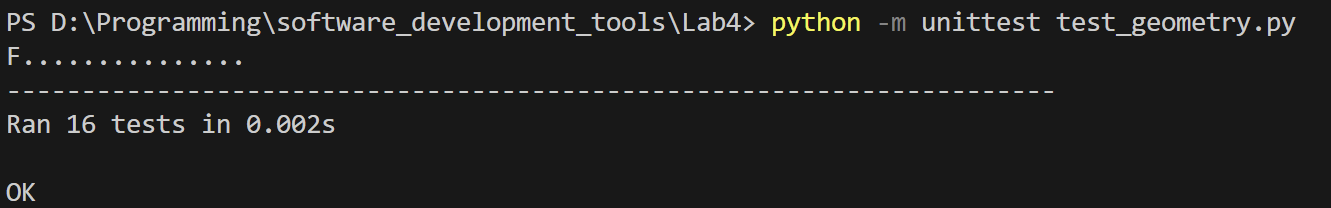


Рис.1 – запуск тестов в терминале

Для дополнительного тестирования изменим формулу вычисления площади круга на неправильную. В таком случае в терминале при запуске тестов отображается ошибка: тест не пройден

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис.2 – результат работы тестов при неправильно написанной функции

Листинг файла с тестами test\_geometry.py:

import unittest

from circle import area as circle\_area, perimeter as circle\_perimeter

from rectangle import area as rectangle\_area, perimeter as rectangle\_perimeter

from square import area as square\_area, perimeter as square\_perimeter

from triangle import area as triangle\_area, perimeter as triangle\_perimeter

class CircleTestCase(unittest.TestCase):

    def test\_circle\_area\_zero(self):

        res = circle\_area(0)

        self.assertEqual(res, 0)

    def test\_circle\_area\_positive(self):

        res = circle\_area(3)

        self.assertAlmostEqual(res, 28.274333882308138)  # Pi \* 3^2

    def test\_circle\_perimeter\_zero(self):

        res = circle\_perimeter(0)

        self.assertEqual(res, 0)

    def test\_circle\_perimeter\_positive(self):

        res = circle\_perimeter(3)

        self.assertAlmostEqual(res, 18.84955592153876)  # 2 \* Pi \* 3

class RectangleTestCase(unittest.TestCase):

    def test\_zero\_area(self):

        res = rectangle\_area(10, 0)

        self.assertEqual(res, 0)

    def test\_square\_area(self):

        res = rectangle\_area(10, 10)

        self.assertEqual(res, 100)

    def test\_rectangle\_area(self):

        res = rectangle\_area(4, 5)

        self.assertEqual(res, 20)

    def test\_zero\_perimeter(self):

        res = rectangle\_perimeter(0, 5)

        self.assertEqual(res, 10)

    def test\_rectangle\_perimeter(self):

        res = rectangle\_perimeter(4, 5)

        self.assertEqual(res, 18)

class SquareTestCase(unittest.TestCase):

    def test\_zero\_area(self):

        res = square\_area(0)

        self.assertEqual(res, 0)

    def test\_square\_area(self):

        res = square\_area(4)

        self.assertEqual(res, 16)

    def test\_zero\_perimeter(self):

        res = square\_perimeter(0)

        self.assertEqual(res, 0)

    def test\_square\_perimeter(self):

        res = square\_perimeter(4)

        self.assertEqual(res, 16)

class TriangleTestCase(unittest.TestCase):

    def test\_zero\_area(self):

        res = triangle\_area(10, 0)

        self.assertEqual(res, 0)

    def test\_triangle\_area(self):

        res = triangle\_area(10, 5)

        self.assertEqual(res, 25)

    def test\_triangle\_perimeter(self):

        res = triangle\_perimeter(3, 4, 5)

        self.assertEqual(res, 12)

if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

Лист.1 – листинг test\_geometry.py