

**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 № М3307

Старцева Арина Михайловна

Подпись:

Проверил:

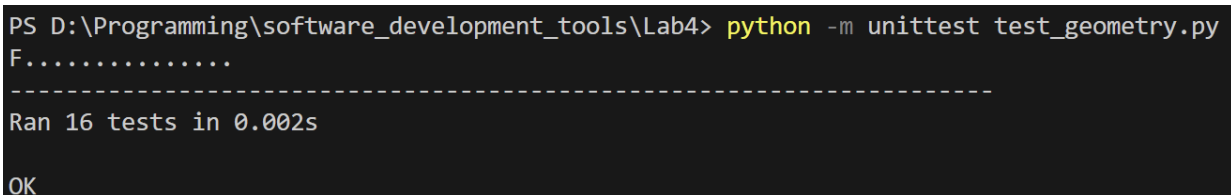
Повышев Владислав Вячеславович

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)
2. Описание тестируемого продукта:
Набор функций для расчёта площади и периметра различных фигур: круга, прямоугольника, квадрата и треугольника
3. 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)
4. Стратегия тестирования:
Для каждой функции будут созданы тесты с корректными значениями, граничными случаями и некорректными данными. Все тесты будут выполнены с использованием библиотеки unittest
5. Критерии приемки:
Тесты считаются пройденными, если они возвращают ожидаемые результаты без ошибок и исключений. Для каждого модуля должны быть пройдены все предназначенные тесты
6. Ожидаемые результаты:
Все тесты должны успешно выполнить расчёты площади и периметра для корректных данных, а также корректно обрабатывать граничные и некорректные входные данные

Приложение



```
PS D:\Programming\software_development_tools\Lab4> python -m unittest test_geometry.py
F.....
-----
Ran 16 tests in 0.002s

OK
```

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

```
PS D:\Programming\software_development_tools\Lab4> python -m unittest test_geometry.py
F.....
=====
FAIL: test_circle_area_positive (test_geometry.CircleTestCase)
-----
Traceback (most recent call last):
  File "D:\Programming\software_development_tools\Lab4\test_geometry.py", line 16, in test_circle_area_positive
    self.assertEqual(res, 28.274333882308138) # Pi * 3^2
AssertionError: 9.42477796076938 != 28.274333882308138 within 7 places (18.84955592153876 difference)
-----
Ran 16 tests in 0.002s
```

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.assertEqual(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.assertEqual(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()
```