

**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 № М3101

Мутаева Олеся Богдановна

Проверил:

Прядкин Александр Олегович

Санкт-Петербург

2023

Ход выполнения работы

https://github.com/l3sssia/geometric_lib/

План тестирования:

1. Цели и задачи тестирования: Определение основных целей тестирования и задач, которые необходимо достичь в процессе тестирования.
 - Проверка правильности расчета площади и периметра для каждой фигуры.
 - Выявление и исправление ошибок в расчетах.
 - Убедиться, что функции работают корректно при различных входных данных.
2. Описание тестируемого продукта: Обзор функциональности, особенностей и требований к продукту, которые должны быть протестированы.
 - `rectangle.py`, `square.py`, `circle.py`, `triangle.py` содержат функции для расчета площади и периметра соответствующих фигур.
 - Функции `area()` и `perimeter()` принимают входные значения и возвращают рассчитанную площадь или периметр.
 - Требования к продукту: корректность работы каждой функции. При использовании `float` допускается погрешность вычислений, при входных данных, не являющимися возможными в реальном мире (круг с радиусом <0 и подобные) – `undefined behavior`.
3. Область тестирования: Определение конкретных функций, модулей или компонентов продукта, которые будут исследованы в тестирование.
 - Проверка на корректность обработки различных входных данных, включая положительные, отрицательные значения, нули.

4. Стратегия тестирования: Описание общего подхода к тестированию, включая методы, техники и типы тестирования, которые будут использоваться, например, функциональное тестирование, тестирование производительности, тестирование безопасности и т. д.

В данной задаче используется функциональное тестирование.

5. Критерии приемки: Определение условий и критериев, которые должны быть выполнены для успешного завершения тестирования и приемки продукта.

- Тесты: Наличие комплекса тестов, покрывающих все функции библиотеки и проверяющих их работоспособность с различными входными данными.
- Обработка ошибок: Библиотека должна корректно обрабатывать некорректные входные данные, такие как отрицательные значения сторон или радиусов, а также некорректные типы данных.
- Документация: Библиотека должна быть хорошо задокументирована, включая описание каждой функции, их входных параметров, выходных значений и возможных ошибок.

6. Ожидаемые результаты: Указание ожидаемых результатов тестирования, таких как отчеты о дефектах, статусы тестирования, метрики качества и другие соответствующие данные.

Ручное тестирование

Долго, неудобно, ненадежно, при любых изменениях в исходном коде есть необходимость заново вручную проверять все тесты и не забывать про это.

Unittest

Для тестирования `geometric_lib`, воспользуемся `unittest`

Для тестирования каждого отдельного модуля будем использовать отдельный файл тестов, создадим папку tests для них.

Файл теста для прямоугольника будет выглядеть следующим образом:

```
tests > rectangle_test.py > ...
1  import unittest
2  import rectangle
3
4  class RectangleTestCase(unittest.TestCase):
5      def test_zero_mul(self):
6          res = rectangle.area(10, 0)
7          self.assertEqual(res, 0)
8          res = rectangle.area(0, 10)
9          self.assertEqual(res, 0)
10
11     def test_square_mul(self):
12         res = rectangle.area(10, 10)
13         self.assertEqual(res, 100)
14
15     def test_sum(self):
16         res = rectangle.perimeter(10, 2)
17         self.assertEqual(res, 24)
18
19     def test_negative(self):
20         with self.assertRaises(ValueError):
21             rectangle.area(-10, 1)
22         with self.assertRaises(ValueError):
23             rectangle.area(10, -1)
24         with self.assertRaises(ValueError):
25             rectangle.perimeter(10, -1)
26         with self.assertRaises(ValueError):
27             rectangle.perimeter(-10, 1)
28
```

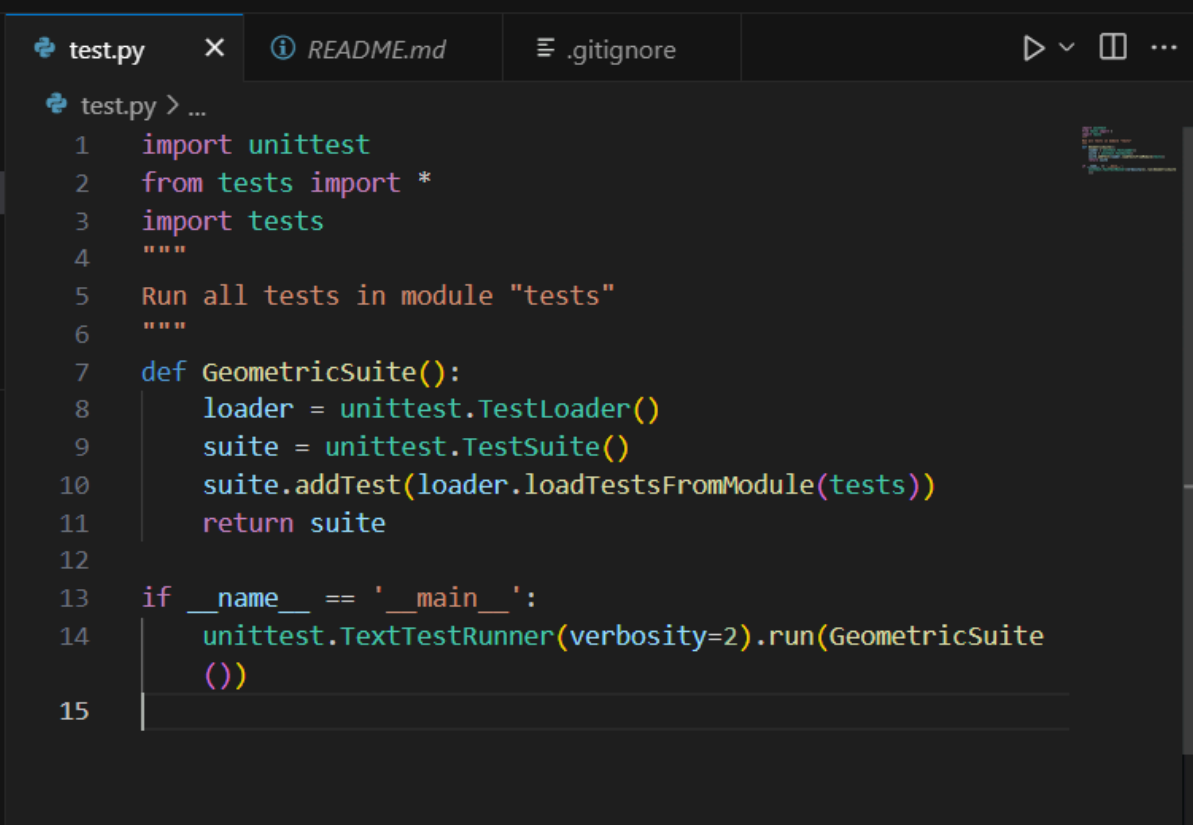
В тесте должны быть учтены проверки на отрицательные стороны и радиусы.

Аналогичным образом сделаем тесты для остальных фигур, добавим их в репозиторий:

```
C:\Users\lessia\ITMO\s1\devtools\lab4\geometric_lib>git status
On branch main
Your branch is ahead of 'origin/main' by 4 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   tests/circle_test.py
        new file:   tests/rectangle_test.py
        new file:   tests/square_test.py
        new file:   tests/triangle_test.py
```

Теперь можно отдельно запускать тесты для каждого модуля, но можно и объединить их.



The screenshot shows a code editor with a dark theme. The top bar has tabs for 'test.py', 'README.md', and '.gitignore'. The 'test.py' tab is active, showing a Python script. The script imports unittest and the 'tests' module, then defines a 'GeometricSuite()' function that uses unittest's TestLoader and TestSuite to load tests from the 'tests' module. Finally, it has a main block that runs the suite using TextTestRunner with verbosity=2. Line numbers 1 through 15 are visible on the left side of the code.

```
test.py > ...
1  import unittest
2  from tests import *
3  import tests
4  """
5  Run all tests in module "tests"
6  """
7  def GeometricSuite():
8      loader = unittest.TestLoader()
9      suite = unittest.TestSuite()
10     suite.addTest(loader.loadTestsFromModule(tests))
11     return suite
12
13 if __name__ == '__main__':
14     unittest.TextTestRunner(verbosity=2).run(GeometricSuite
15     ())
```

Можно запустить сразу все тесты:

```
PS C:\Users\lessia\ITMO\s1\devtools\lab4\geometric_lib> python3 -m unittest -v test.py
test_negative (tests.test_circle.CircleTestCase) ... ok
test_zero_mul (tests.test_circle.CircleTestCase) ... ok
test_negative (tests.test_rectangle.RectangleTestCase) ... ok
test_square_mul (tests.test_rectangle.RectangleTestCase) ... ok
test_sum (tests.test_rectangle.RectangleTestCase) ... ok
test_zero_mul (tests.test_rectangle.RectangleTestCase) ... ok
test_mul (tests.test_square.SquareTestCase) ... ok
test_negative (tests.test_square.SquareTestCase) ... ok
test_sum (tests.test_square.SquareTestCase) ... ok
test_zero_mul (tests.test_square.SquareTestCase) ... ok
test_negative (tests.test_triangle.TriangleTestCase) ... ok
test_square_mul (tests.test_triangle.TriangleTestCase) ... ok
test_sum (tests.test_triangle.TriangleTestCase) ... ok
test_zero_mul (tests.test_triangle.TriangleTestCase) ... ok

-----
Ran 14 tests in 0.005s

OK
```

Осталось залить изменения в репозиторий на github и отметить
соответствующие изменения в документации.