

Министерство науки высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет информационных технологий и программирования

Лабораторная работа №4
По дисциплине «Инструментальные средства разработки ПО»

Тестирование

Выполнил студент группы №М3103:
Заречнев Алексей Олегович

Проверил:



Санкт-Петербург
2023

Цели и задачи тестирования: Протестировать компоненты нашего проекта на корректную работу, и исправить недочёты обнаруженные в ходе тестирования.

Описание тестируемого продукта: библиотека для python способная считать площадь и периметр простейших геометрических фигур с целочисленными сторонами.

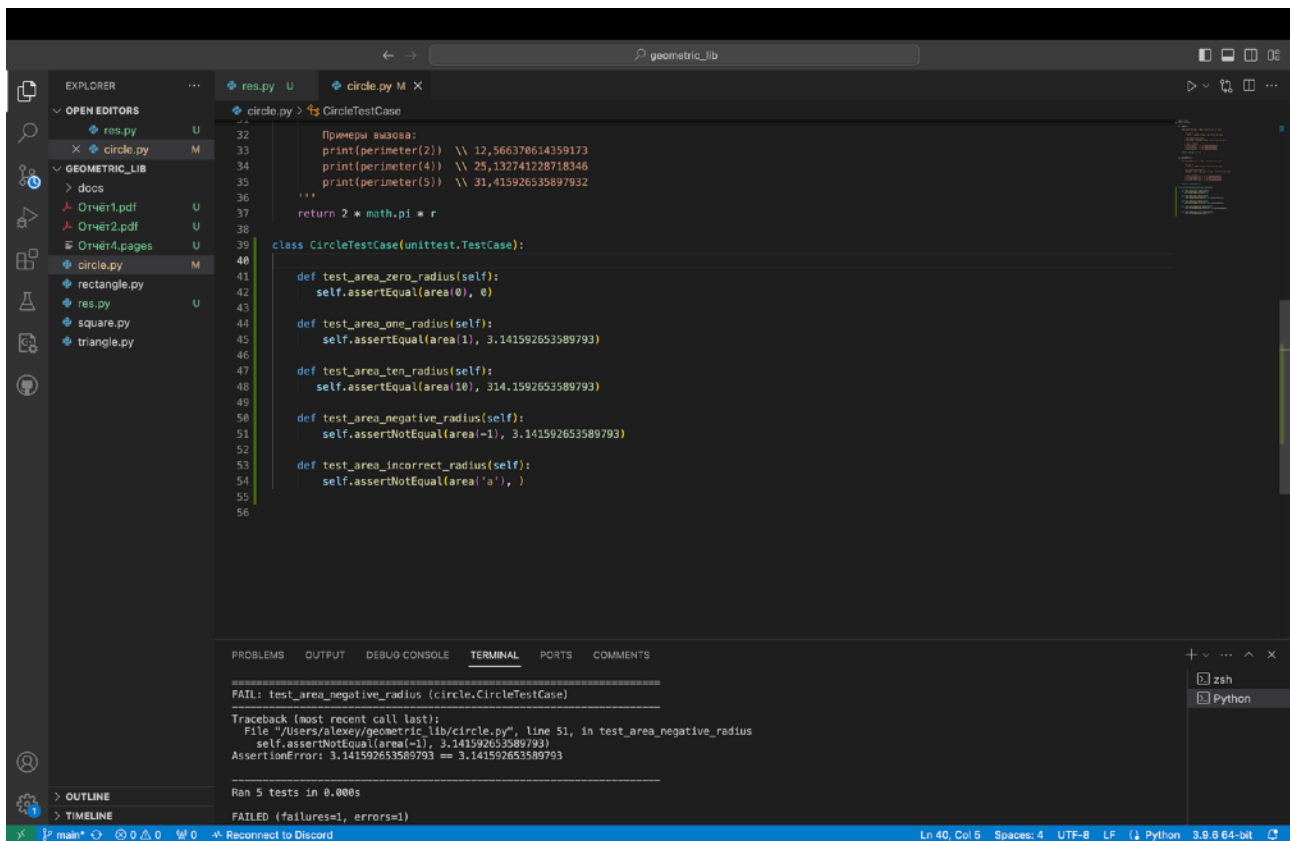
Область тестирования: Тестироваться будут функции `area()` и `perimeter()` в файлах `circle.py`, `rectangle.py`, `square.py`, `triangle.py`

Стратегия тестирования: Тестироваться продукт будет с помощью unit-тестов, будет производиться функциональное тестирование каждого компонента в отдельности. Для простоты написания тестов воспользуемся библиотекой `unittest` для python опишем в ней крайние случаи и немного случайных тестов, чтобы удостовериться, что модули библиотеки работают верно.

Критерии приёмки: Все тесты, у всех модулей пройдены.

Ожидаемые результаты: При корректных значениях программы должны выдавать верные ответы, при некорректных значениях соответствующую ошибку.

1. Напишем несколько конкретных тестов для файла circle.py. Протестируем результаты периметра и площади при значениях радиуса равных -1, 0, 1, 10 и 'a'.



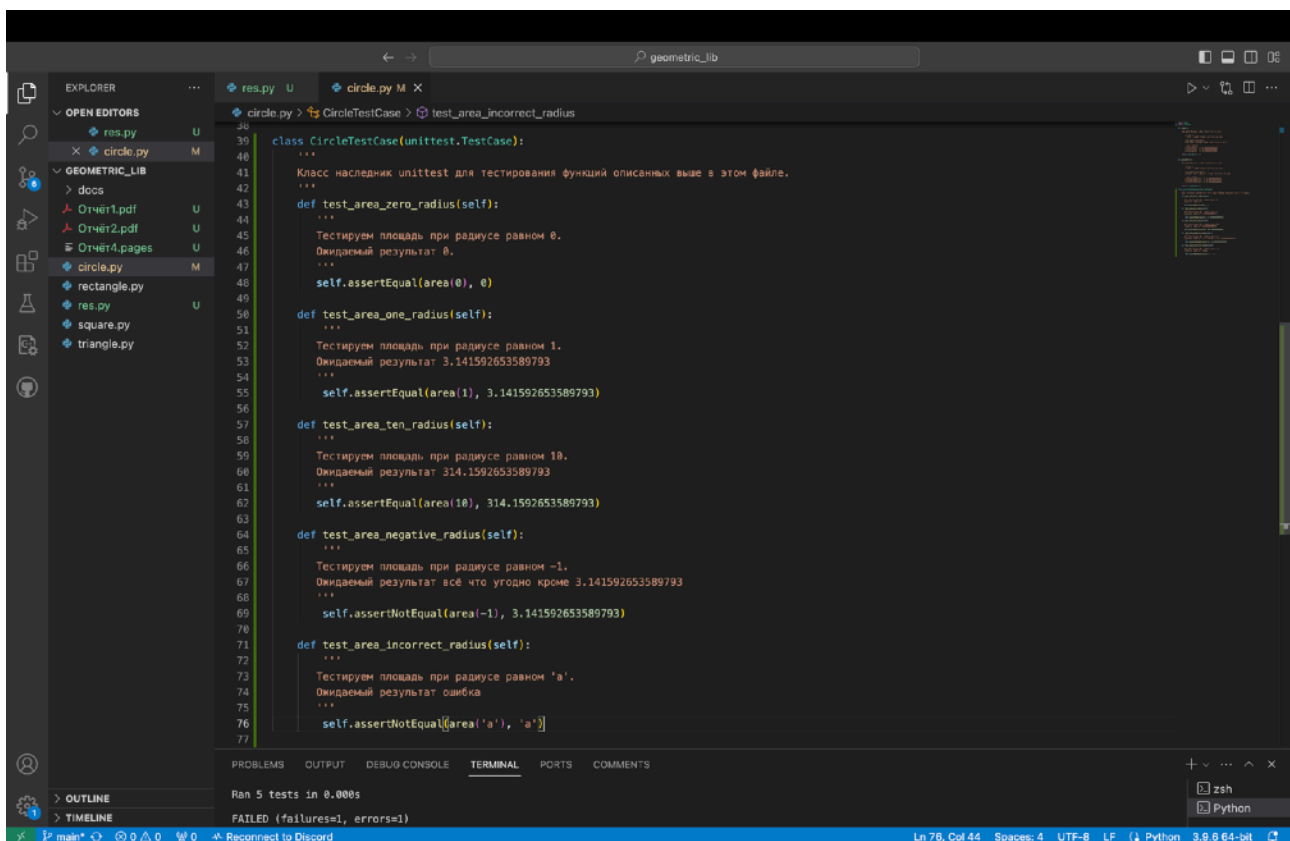
```
32  Примеры вызова:
33  print(perimeter(2)) \\ 12,566370614359173
34  print(perimeter(4)) \\ 25,132741228718346
35  print(perimeter(5)) \\ 31,415926535897932
36  ...
37  return 2 * math.pi * r
38
39  class CircleTestCase(unittest.TestCase):
40
41  def test_area_zero_radius(self):
42      self.assertEqual(area(0), 0)
43
44  def test_area_one_radius(self):
45      self.assertEqual(area(1), 3.141592653589793)
46
47  def test_area_ten_radius(self):
48      self.assertEqual(area(10), 314.1592653589793)
49
50  def test_area_negative_radius(self):
51      self.assertNotEqual(area(-1), 3.141592653589793)
52
53  def test_area_incorrect_radius(self):
54      self.assertNotEqual(area('a'), )
55
56
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
FAIL: test_area_negative_radius (circle.CircleTestCase)
Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/circle.py", line 51, in test_area_negative_radius
    self.assertNotEqual(area(-1), 3.141592653589793)
AssertionError: 3.141592653589793 == 3.141592653589793

Ran 5 tests in 0.000s
FAILED (failures=1, errors=1)
```

2. Добавим комментарии к новым функциям



```
39  class CircleTestCase(unittest.TestCase):
40  ...
41  ...
42  ...
43  def test_area_zero_radius(self):
44  ...
45  ...
46  ...
47  self.assertEqual(area(0), 0)
48
49  def test_area_one_radius(self):
50  ...
51  ...
52  ...
53  self.assertEqual(area(1), 3.141592653589793)
54
55  def test_area_ten_radius(self):
56  ...
57  ...
58  ...
59  self.assertEqual(area(10), 314.1592653589793)
60
61  def test_area_negative_radius(self):
62  ...
63  ...
64  ...
65  self.assertNotEqual(area(-1), 3.141592653589793)
66
67  def test_area_incorrect_radius(self):
68  ...
69  ...
70  ...
71  self.assertNotEqual(area('a'), 'a')
72
73
74
75
76
77
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
Ran 5 tests in 0.000s
FAILED (failures=1, errors=1)
```

3. Наконец запускаем тесты и находим, что один из них вылетает с ошибкой, а другой выдаёт неверный ответ.

```
> /usr/bin/python3 -m unittest circle.py
EF...
=====
ERROR: test_area_incorrect_radius (circle.CircleTestCase)
Тестируем площадь при радиусе равном 'a'. Ожидаемый результат ошибка
=====
Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/circle.py", line 71, in test_area_incorrect_radius
    self.assertNotEqual(area('a'), 'a')
  File "/Users/alexey/geometric_lib/circle.py", line 19, in area
    return math.pi * r * r
TypeError: can't multiply sequence by non-int of type 'float'
=====
FAIL: test_area_negative_radius (circle.CircleTestCase)
Тестируем площадь при радиусе равном -1. Ожидаемый результат всё что угодно кроме 3.141592653589793
=====
Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/circle.py", line 65, in test_area_negative_radius
    self.assertNotEqual(area(-1), 3.141592653589793)
AssertionError: 3.141592653589793 == 3.141592653589793
=====

Ran 5 tests in 0.000s

FAILED (failures=1, errors=1)
```

4. Исправляем `assertEqual`, на `assertRaises`, чтобы исправить ошибку во время тестирования

The screenshot shows the VS Code editor with the `circle.py` file open. The file contains the following code:

```
57     Тестируем площадь при радиусе равном 10. Ожидаемый результат 314.1592653589793
58     ...
59     self.assertEqual(area(10), 314.1592653589793)
60
61     def test_area_negative_radius(self):
62     ...
63     Тестируем площадь при радиусе равном -1. Ожидаемый результат всё что угодно кроме 3.141592653589793
64     ...
65     self.assertNotEqual(area(-1), 3.141592653589793)
66
67     def test_area_incorrect_radius(self):
68     ...
69     Тестируем площадь при радиусе равном 'a'. Ожидаемый результат ошибка
70     ...
71     with self.assertRaises(TypeError):
72         area('a')
73
74
```

The terminal output shows the test results:

```
=====
FAIL: test_area_negative_radius (circle.CircleTestCase)
Тестируем площадь при радиусе равном -1. Ожидаемый результат всё что угодно кроме 3.141592653589793
=====
Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/circle.py", line 65, in test_area_negative_radius
    self.assertNotEqual(area(-1), 3.141592653589793)
AssertionError: 3.141592653589793 == 3.141592653589793
=====

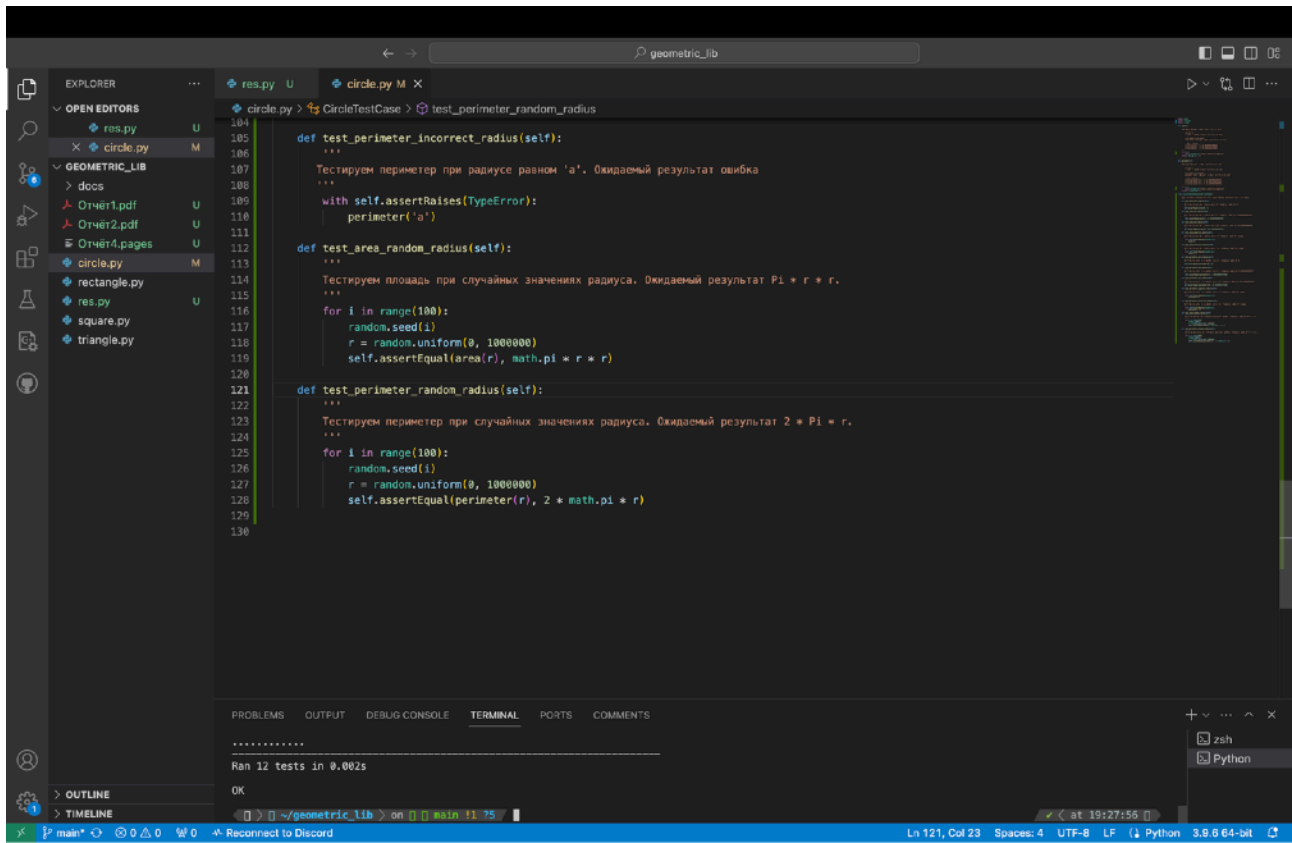
Ran 5 tests in 0.000s

FAILED (failures=1, errors=1)
> /usr/bin/python3 -m unittest circle.py
.F...
=====
FAIL: test_area_negative_radius (circle.CircleTestCase)
Тестируем площадь при радиусе равном -1. Ожидаемый результат всё что угодно кроме 3.141592653589793
=====
Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/circle.py", line 65, in test_area_negative_radius
    self.assertNotEqual(area(-1), 3.141592653589793)
AssertionError: 3.141592653589793 == 3.141592653589793
=====

Ran 5 tests in 0.000s

FAILED (failures=1)
```

6. Добавим немного случайных тестов, да бы точно знать, что компоненты работают верно.



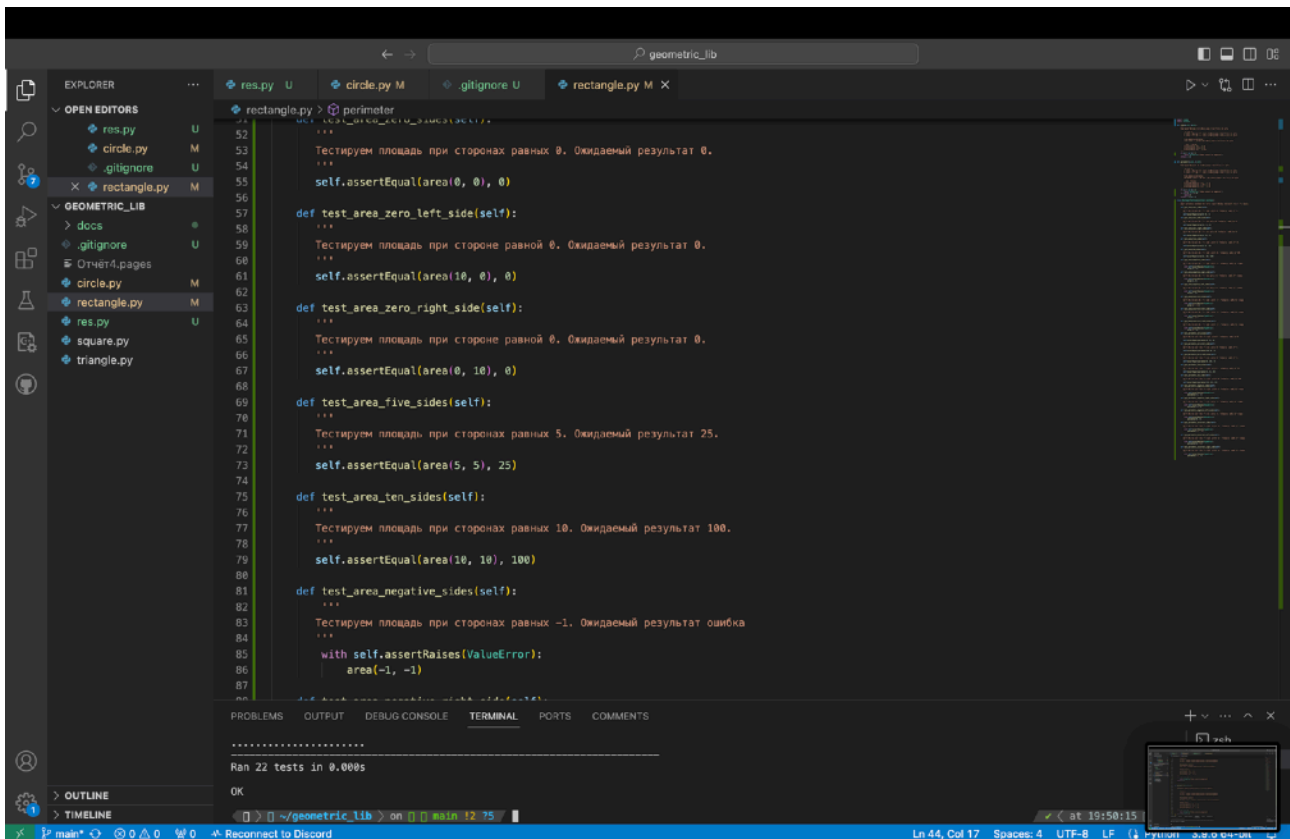
```
def test_perimeter_incorrect_radius(self):
    """
    Тестируем периметр при радиусе равном 'a'. Ожидаемый результат ошибка
    """
    with self.assertRaises(TypeError):
        perimeter('a')

def test_area_random_radius(self):
    """
    Тестируем площадь при случайных значениях радиуса. Ожидаемый результат  $P1 = r * r$ .
    """
    for i in range(100):
        random.seed(i)
        r = random.uniform(0, 1000000)
        self.assertEqual(area(r), math.pi * r * r)

def test_perimeter_random_radius(self):
    """
    Тестируем периметр при случайных значениях радиуса. Ожидаемый результат  $2 * \pi * r$ .
    """
    for i in range(100):
        random.seed(i)
        r = random.uniform(0, 1000000)
        self.assertEqual(perimeter(r), 2 * math.pi * r)
```

Ran 12 tests in 0.002s
OK

7. Повторяем тоже самое для rectangle.py



```
def test_perimeter(self):
    """
    Тестируем периметр при сторонах равных 0. Ожидаемый результат 0.
    """
    self.assertEqual(perimeter(0, 0), 0)

def test_area_zero_left_side(self):
    """
    Тестируем площадь при стороне равной 0. Ожидаемый результат 0.
    """
    self.assertEqual(area(10, 0), 0)

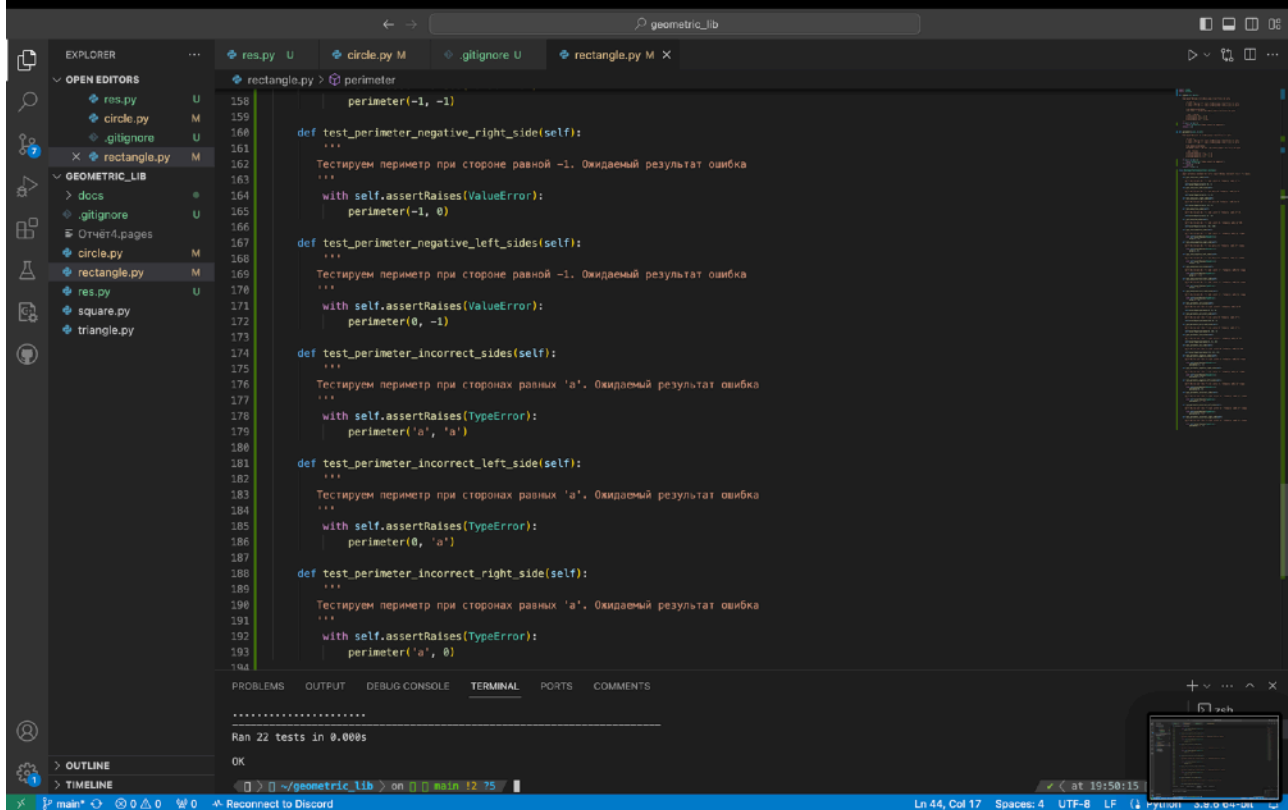
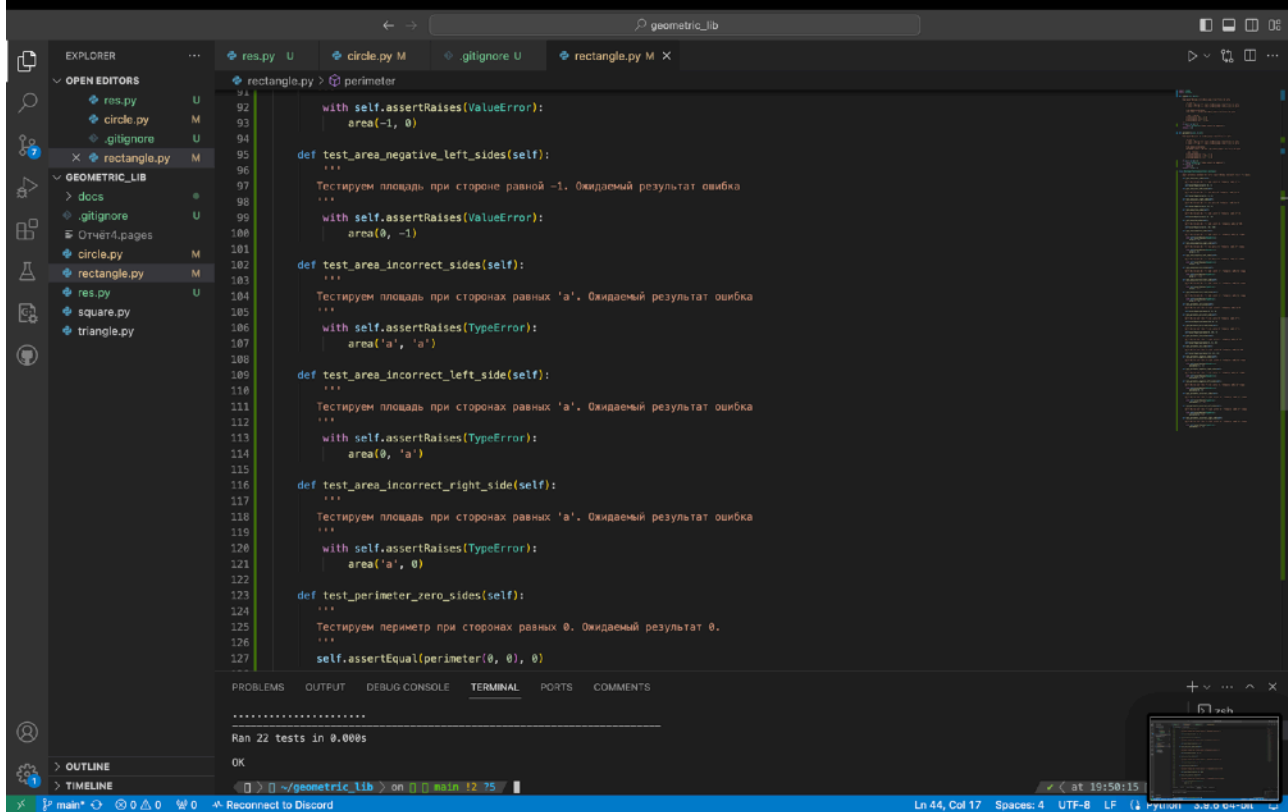
def test_area_zero_right_side(self):
    """
    Тестируем площадь при стороне равной 0. Ожидаемый результат 0.
    """
    self.assertEqual(area(0, 10), 0)

def test_area_five_sides(self):
    """
    Тестируем площадь при сторонах равных 5. Ожидаемый результат 25.
    """
    self.assertEqual(area(5, 5), 25)

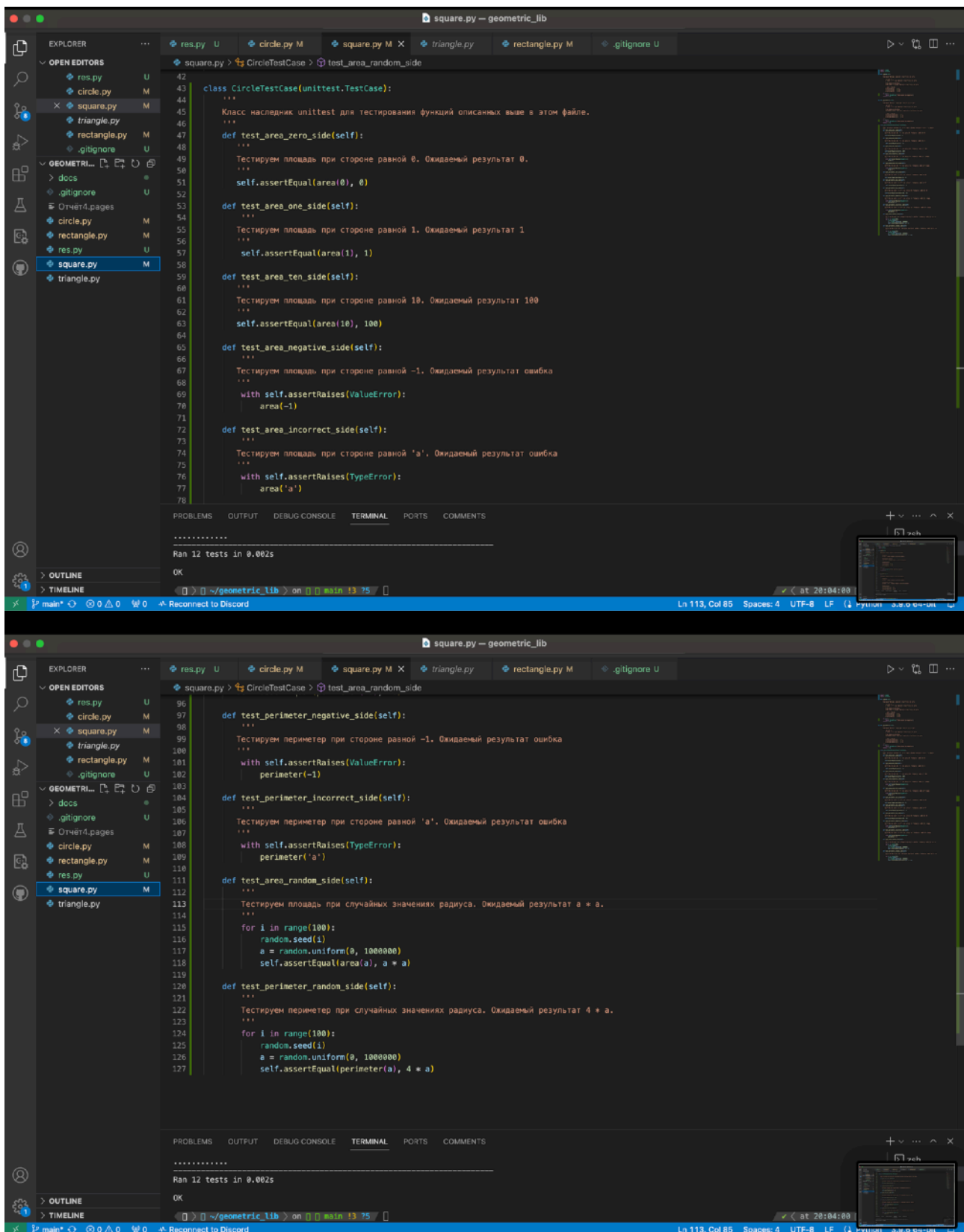
def test_area_ten_sides(self):
    """
    Тестируем площадь при сторонах равных 10. Ожидаемый результат 100.
    """
    self.assertEqual(area(10, 10), 100)

def test_area_negative_sides(self):
    """
    Тестируем площадь при сторонах равных -1. Ожидаемый результат ошибка
    """
    with self.assertRaises(ValueError):
        area(-1, -1)
```

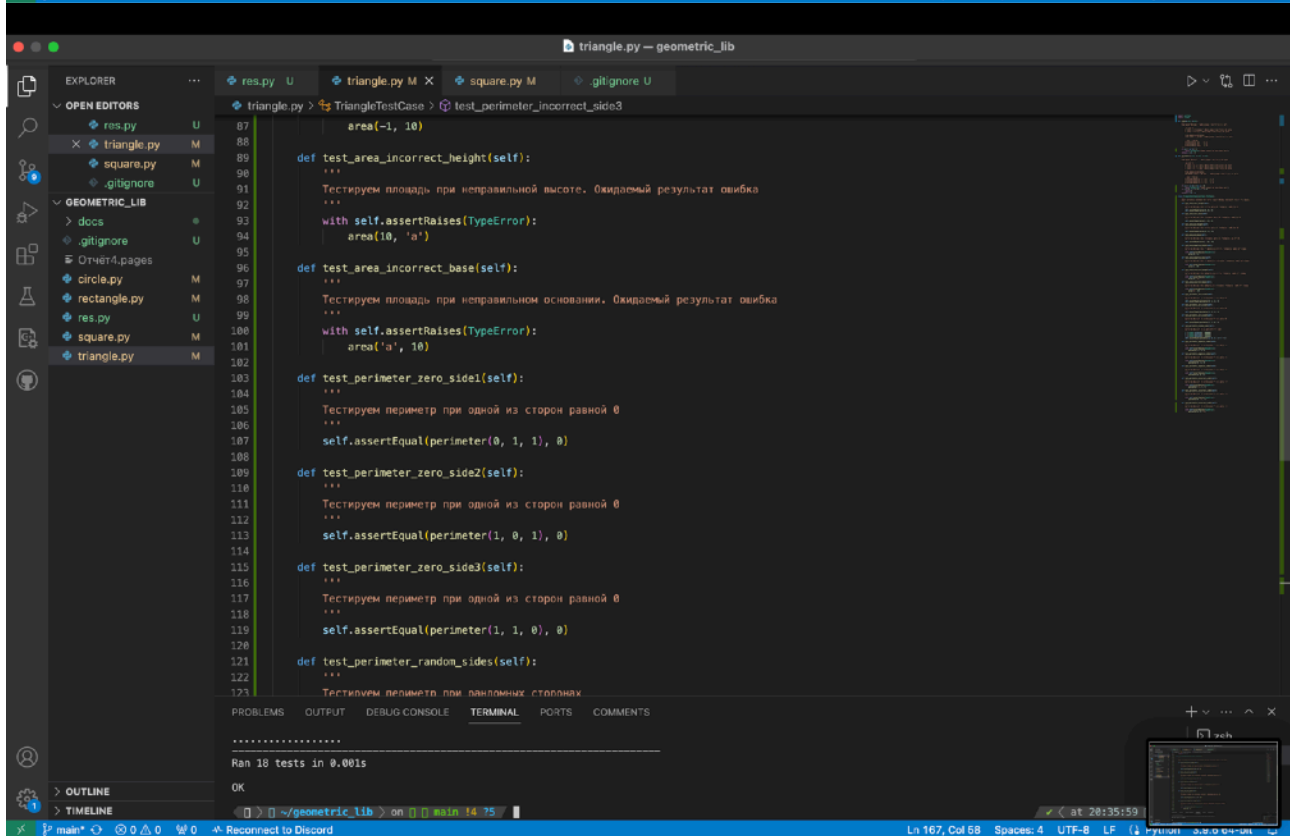
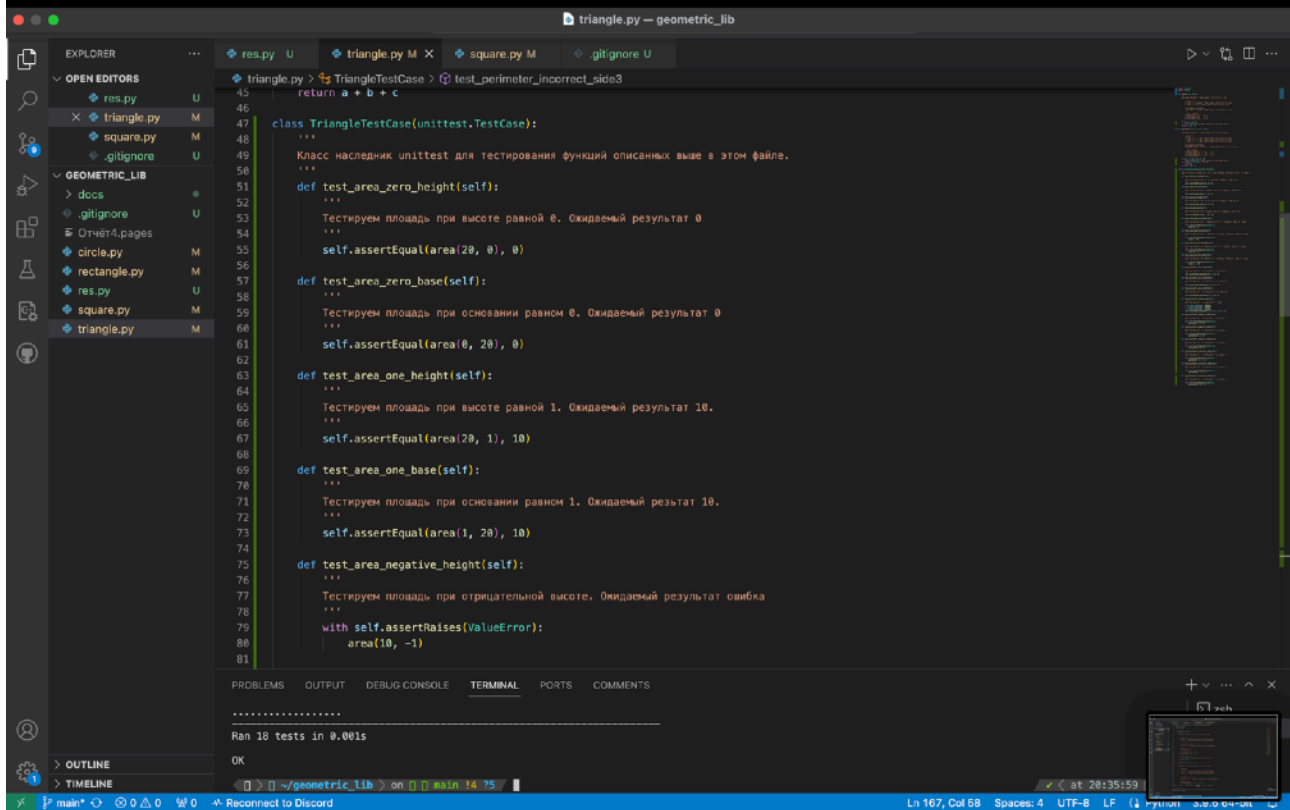
Ran 22 tests in 0.000s
OK

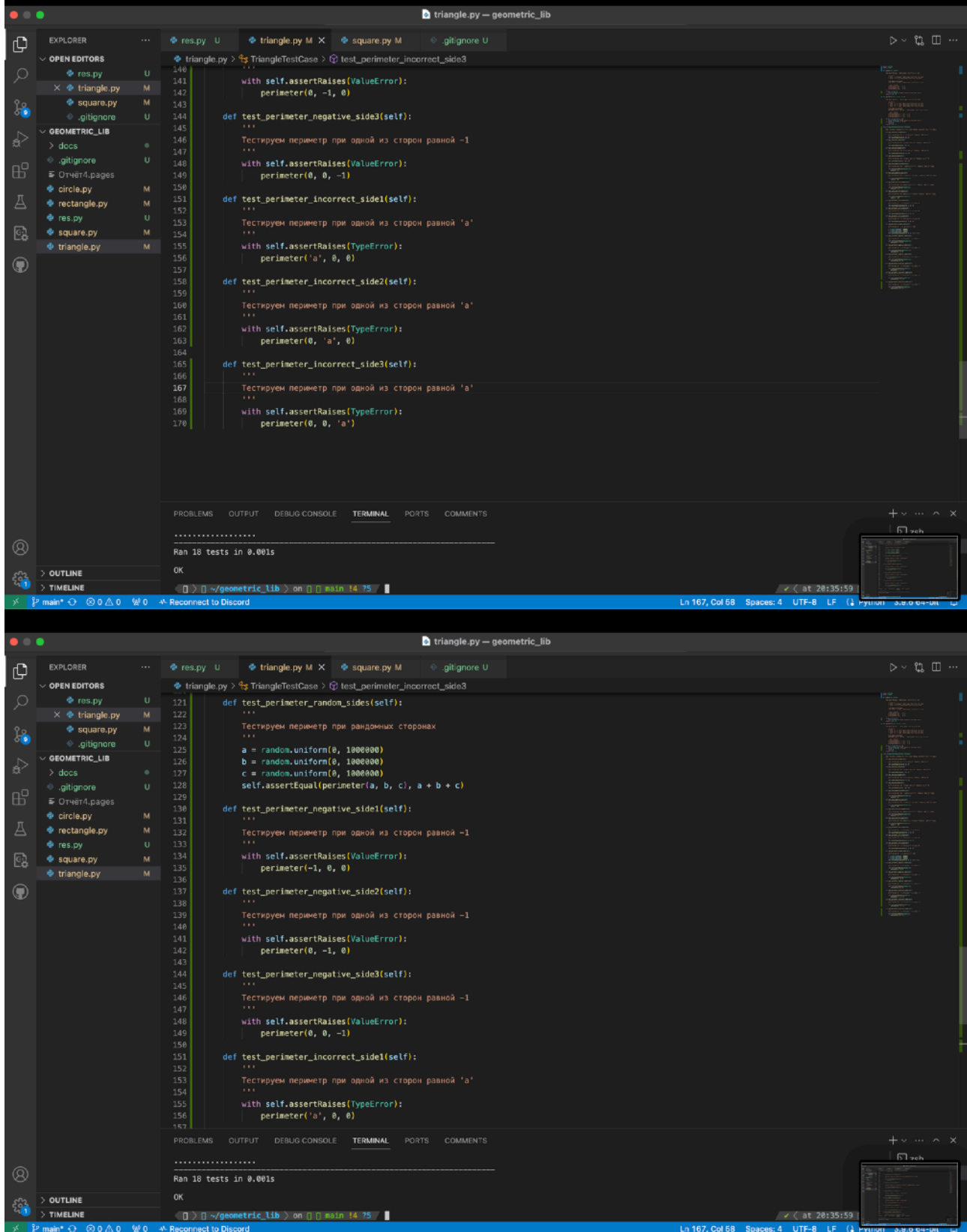


8. Теперь для square.py

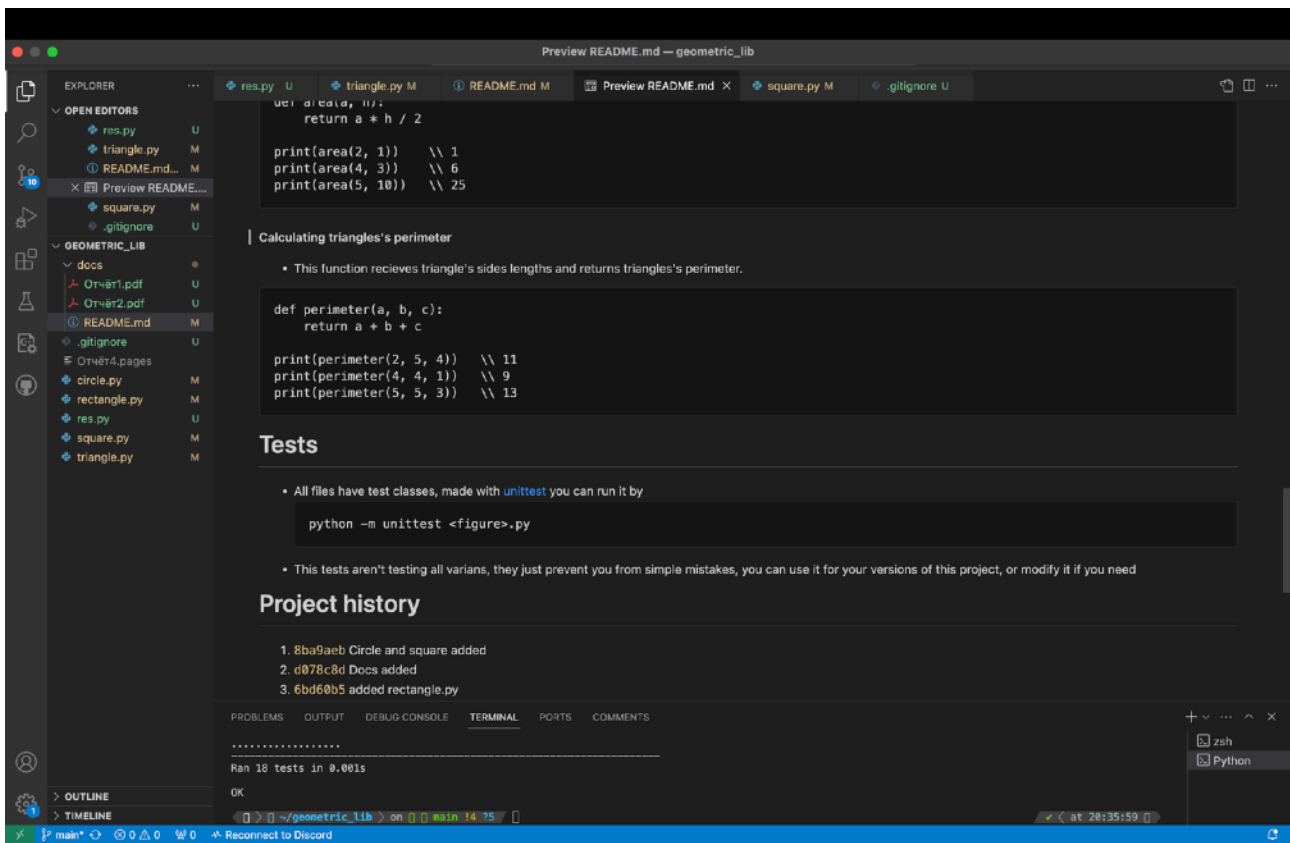


9. Теперь для triangle.py





10. Теперь пропишем в документации о добавлении тестов



Результаты прохождения написанных тестов:
circle.py

```
> /usr/bin/python3 -m unittest circle.py
.F.....F....

=====
FAIL: test_area_negative_radius (circle.CircleTestCase)
Тестируем площадь при радиусе равном -1. Ожидаемый результат ошибка

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/circle.py", line 67, in test_area_negative_radius
    area(-1)
AssertionError: ValueError not raised

=====
FAIL: test_perimeter_negative_radius (circle.CircleTestCase)
Тестируем периметр при радиусе равном -1. Ожидаемый результат ошибка

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/circle.py", line 99, in test_perimeter_negative_radius
    perimeter(-1)
AssertionError: ValueError not raised

=====
Ran 12 tests in 0.002s
FAILED (failures=2)
```

square.py

```
> /usr/bin/python3 -m unittest square.py
.F....FF....

=====
FAIL: test_area_negative_side (square.SquareTestCase)
Тестируем площадь при стороне равной -1. Ожидаемый результат ошибка

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/square.py", line 66, in test_area_negative_side
    area(-1)
AssertionError: ValueError not raised

=====
FAIL: test_perimeter_incorrect_side (square.SquareTestCase)
Тестируем периметр при стороне равной 'a'. Ожидаемый результат ошибка

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/square.py", line 105, in test_perimeter_incorrect_side
    perimeter('a')
AssertionError: TypeError not raised

=====
FAIL: test_perimeter_negative_side (square.SquareTestCase)
Тестируем периметр при стороне равной -1. Ожидаемый результат ошибка

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/square.py", line 98, in test_perimeter_negative_side
    perimeter(-1)
AssertionError: ValueError not raised

=====
Ran 12 tests in 0.002s
FAILED (failures=3)
```

rectangle.py

```
=====
FAIL: test_perimeter_negative_left_sides (rectangle.RectangleTestCase)
Тестируем периметр при стороне равной -1. Ожидаемый результат ошибка

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/rectangle.py", line 166, in test_perimeter_negative_left_sides
    perimeter(0, -1)
AssertionError: ValueError not raised

=====
FAIL: test_perimeter_negative_right_side (rectangle.RectangleTestCase)
Тестируем периметр при стороне равной -1. Ожидаемый результат ошибка

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/rectangle.py", line 159, in test_perimeter_negative_right_side
    perimeter(-1, 0)
AssertionError: ValueError not raised

=====
FAIL: test_perimeter_negative_sides (rectangle.RectangleTestCase)
Тестируем периметр при сторонах равных -1. Ожидаемый результат ошибка

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/rectangle.py", line 152, in test_perimeter_negative_sides
    perimeter(-1, -1)
AssertionError: ValueError not raised

=====
FAIL: test_perimeter_zero_left_side (rectangle.RectangleTestCase)
Тестируем периметр при стороне равной 0. Ожидаемый результат 0.

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/rectangle.py", line 127, in test_perimeter_zero_left_side
    self.assertEqual(perimeter(10, 0), 0)
AssertionError: 20 != 0

=====
FAIL: test_perimeter_zero_right_side (rectangle.RectangleTestCase)
Тестируем периметр при стороне равной 0. Ожидаемый результат 0.

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/rectangle.py", line 133, in test_perimeter_zero_right_side
    self.assertEqual(perimeter(0, 10), 0)
AssertionError: 20 != 0

=====
Ran 22 tests in 0.001s

FAILED (failures=11)
```

triangle.py

```
=====
FAIL: test_perimeter_negative_side2 (triangle.TriangleTestCase)
Тестируем периметр при одной из сторон равной -1

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/triangle.py", line 136, in test_perimeter_negative_side2
    perimeter(0, -1, 0)
AssertionError: ValueError not raised

=====
FAIL: test_perimeter_negative_side3 (triangle.TriangleTestCase)
Тестируем периметр при одной из сторон равной -1

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/triangle.py", line 143, in test_perimeter_negative_side3
    perimeter(0, 0, -1)
AssertionError: ValueError not raised

=====
FAIL: test_perimeter_zero_side1 (triangle.TriangleTestCase)
Тестируем периметр при одной из сторон равной 0

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/triangle.py", line 101, in test_perimeter_zero_side1
    self.assertEqual(perimeter(0, 1, 1), 0)
AssertionError: 2 != 0

=====
FAIL: test_perimeter_zero_side2 (triangle.TriangleTestCase)
Тестируем периметр при одной из сторон равной 0

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/triangle.py", line 107, in test_perimeter_zero_side2
    self.assertEqual(perimeter(1, 0, 1), 0)
AssertionError: 2 != 0

=====
FAIL: test_perimeter_zero_side3 (triangle.TriangleTestCase)
Тестируем периметр при одной из сторон равной 0

Traceback (most recent call last):
  File "/Users/alexey/geometric_lib/triangle.py", line 113, in test_perimeter_zero_side3
    self.assertEqual(perimeter(1, 1, 0), 0)
AssertionError: 2 != 0

=====
Ran 18 tests in 0.001s

FAILED (failures=8)
```