

**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'.

The screenshot shows the VS Code editor with the file `circle.py` open. The file contains a `CircleTestCase` class with several test methods. The `test_area_negative_radius` method is failing, as indicated by the error message in the terminal.

```
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
```

The terminal output shows the following error:

```
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

The screenshot shows the VS Code editor with the file `circle.py` open. The file contains a `CircleTestCase` class with several test methods. Each method now includes a docstring with a description of the test and the expected result.

```
39 class CircleTestCase(unittest.TestCase):
40     """
41     Класс наследник unittest для тестирования функций описанных выше в этом файле.
42     """
43
44     def test_area_zero_radius(self):
45         """
46         Тестируем площадь при радиусе равном 0.
47         Ожидаемый результат 0.
48         """
49         self.assertEqual(area(0), 0)
50
51     def test_area_one_radius(self):
52         """
53         Тестируем площадь при радиусе равном 1.
54         Ожидаемый результат 3.141592653589793
55         """
56         self.assertEqual(area(1), 3.141592653589793)
57
58     def test_area_ten_radius(self):
59         """
60         Тестируем площадь при радиусе равном 10.
61         Ожидаемый результат 314.1592653589793
62         """
63         self.assertEqual(area(10), 314.1592653589793)
64
65     def test_area_negative_radius(self):
66         """
67         Тестируем площадь при радиусе равном -1.
68         Ожидаемый результат все что угодно кроме 3.141592653589793
69         """
70         self.assertNotEqual(area(-1), 3.141592653589793)
71
72     def test_area_incorrect_radius(self):
73         """
74         Тестируем площадь при радиусе равном 'a'.
75         Ожидаемый результат ошибка
76         """
77         self.assertNotEqual(area('a'), 'a')
```

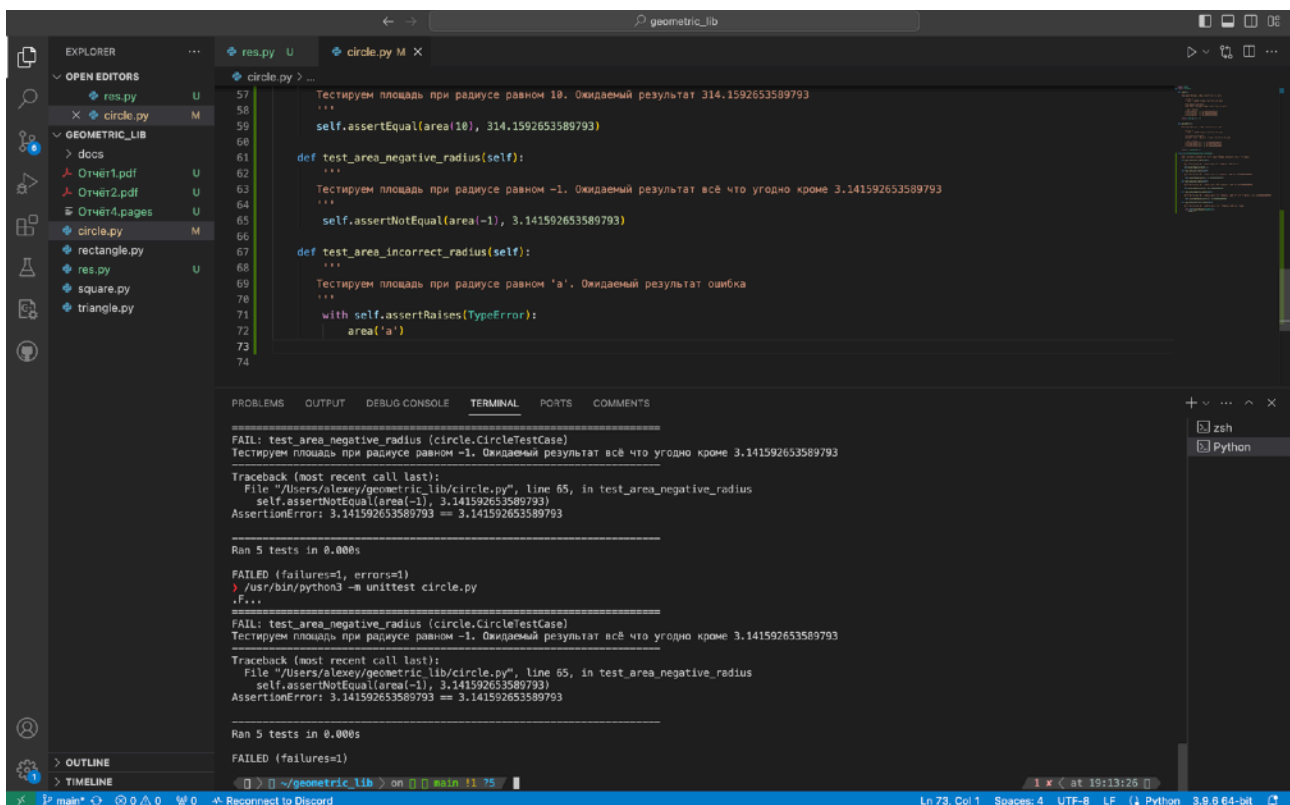
3. Наконец запускаем тесты и находим, что один из них вылетает с ошибкой, а другой выдаёт неверный ответ.

```
> /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
=====

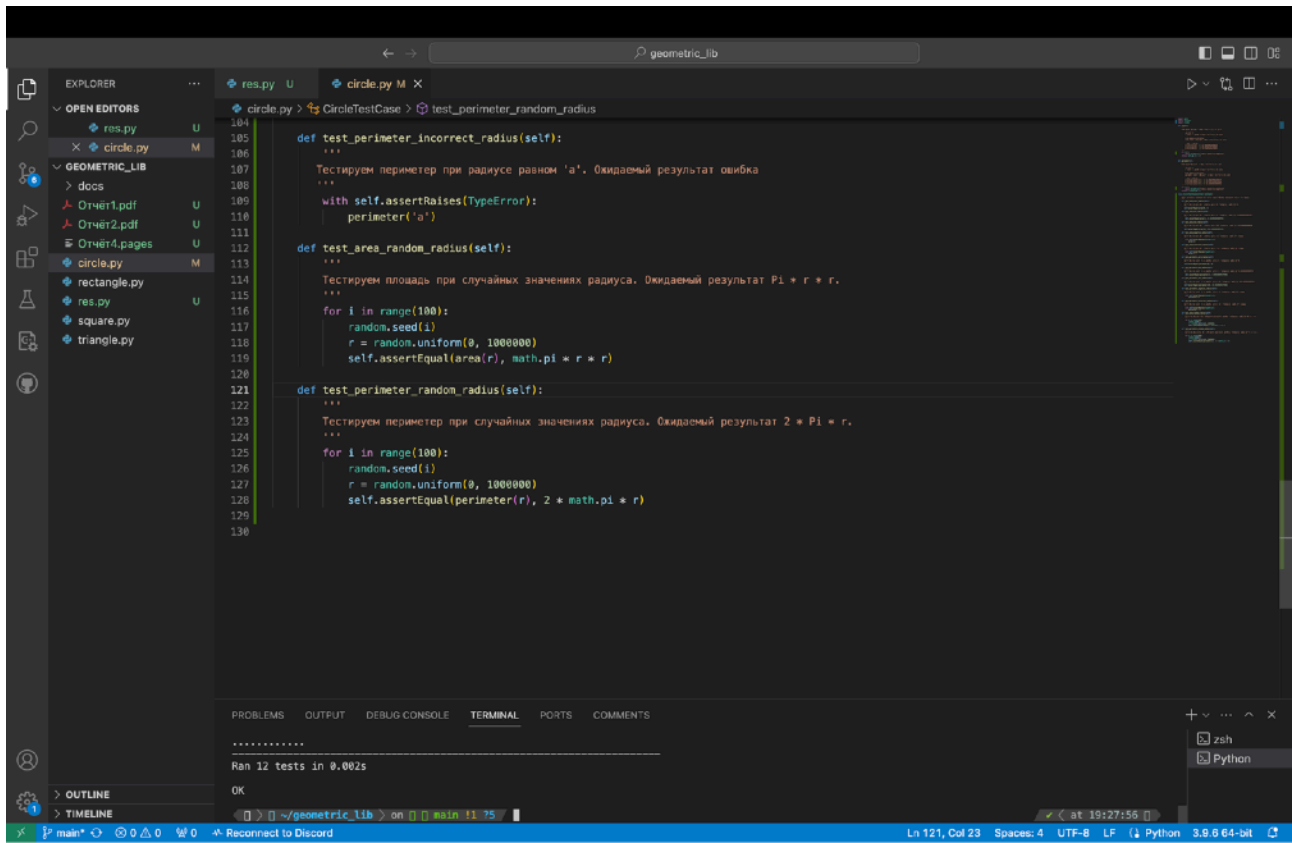
Ran 5 tests in 0.000s

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.



```
104
105
106 def test_perimeter_incorrect_radius(self):
107     """
108     Тестируем периметр при радиусе равном 'a'. Ожидаемый результат ошибка
109     """
110     with self.assertRaises(TypeError):
111         perimeter('a')
112
113 def test_area_random_radius(self):
114     """
115     Тестируем площадь при случайных значениях радиуса. Ожидаемый результат  $P1 = r * r$ .
116     """
117     for i in range(100):
118         random.seed(i)
119         r = random.uniform(0, 1000000)
120         self.assertEqual(area(r), math.pi * r * r)
121
122 def test_perimeter_random_radius(self):
123     """
124     Тестируем периметр при случайных значениях радиуса. Ожидаемый результат  $2 * \pi * r$ .
125     """
126     for i in range(100):
127         random.seed(i)
128         r = random.uniform(0, 1000000)
129         self.assertEqual(perimeter(r), 2 * math.pi * r)
130
```

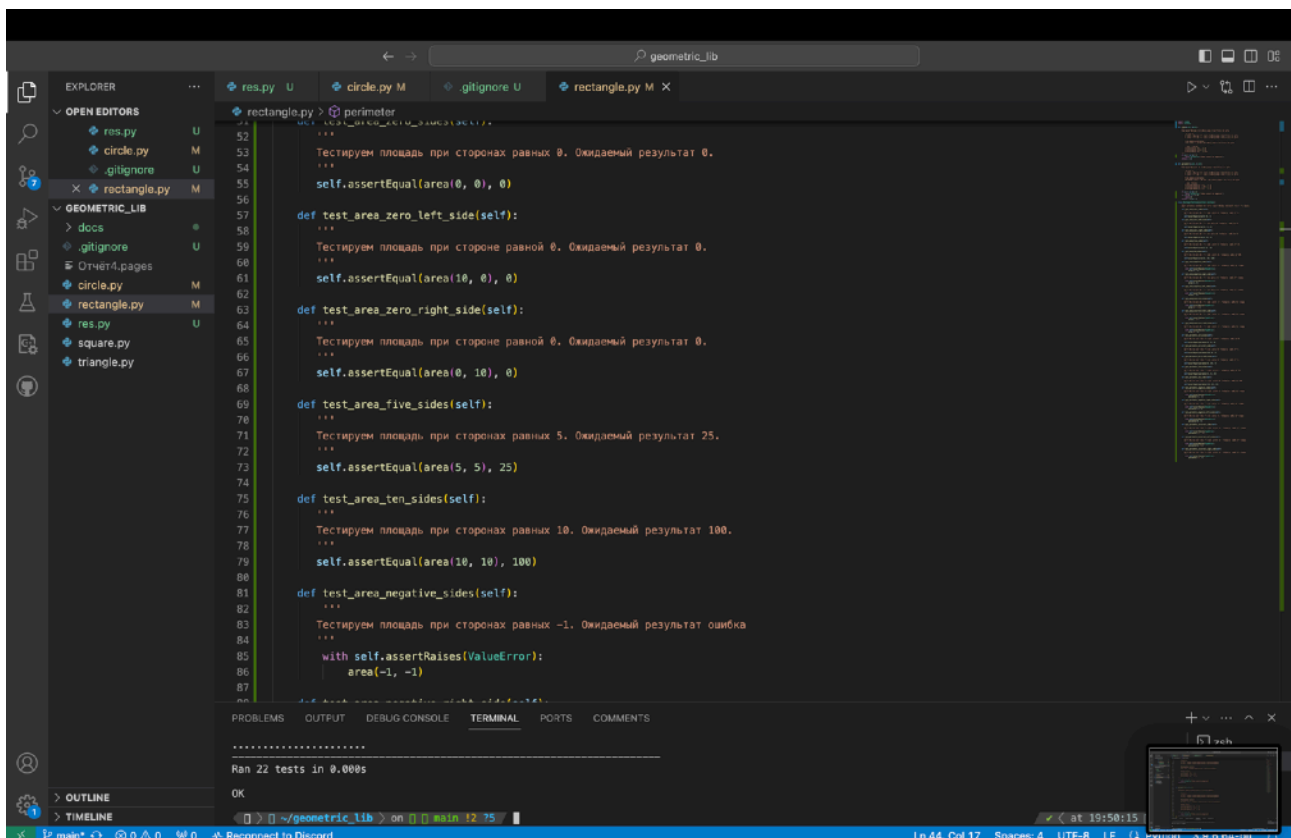
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

.....

Ran 12 tests in 0.002s

OK

7. Повторяем тоже самое для rectangle.py



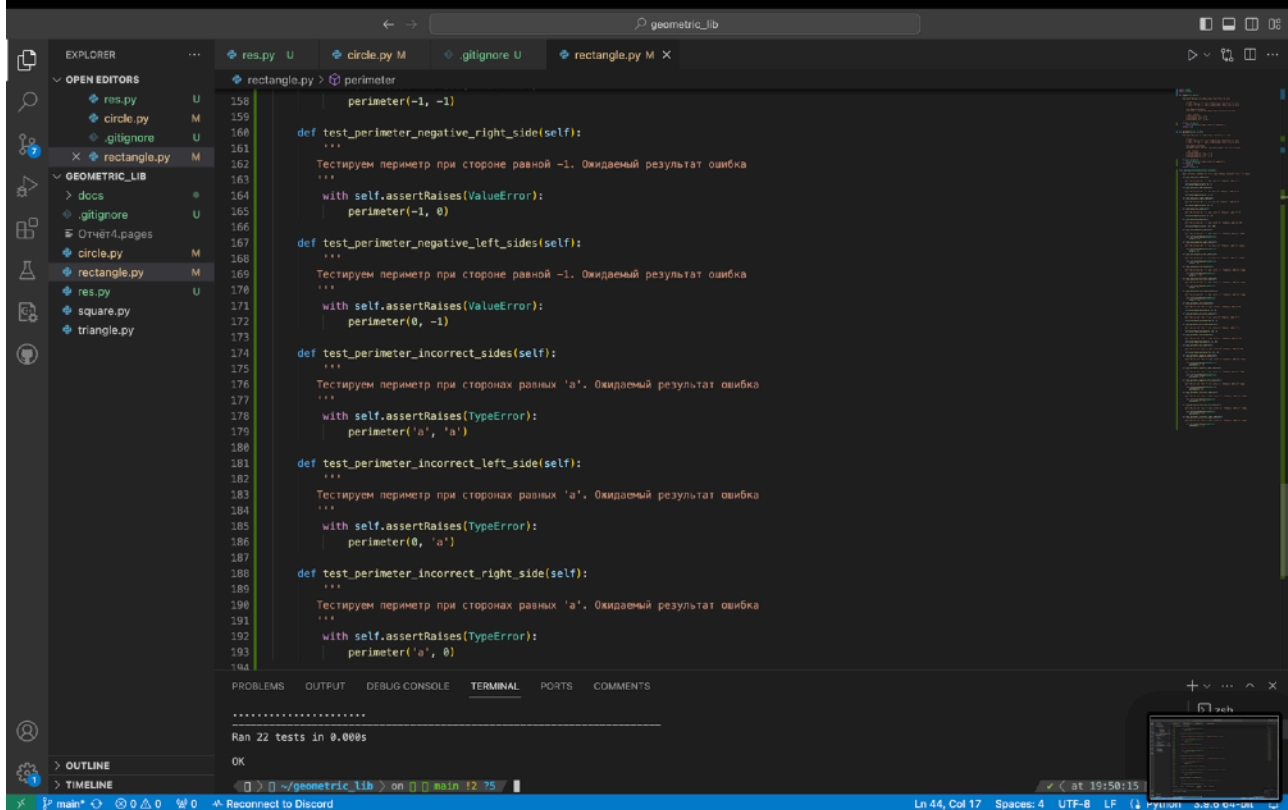
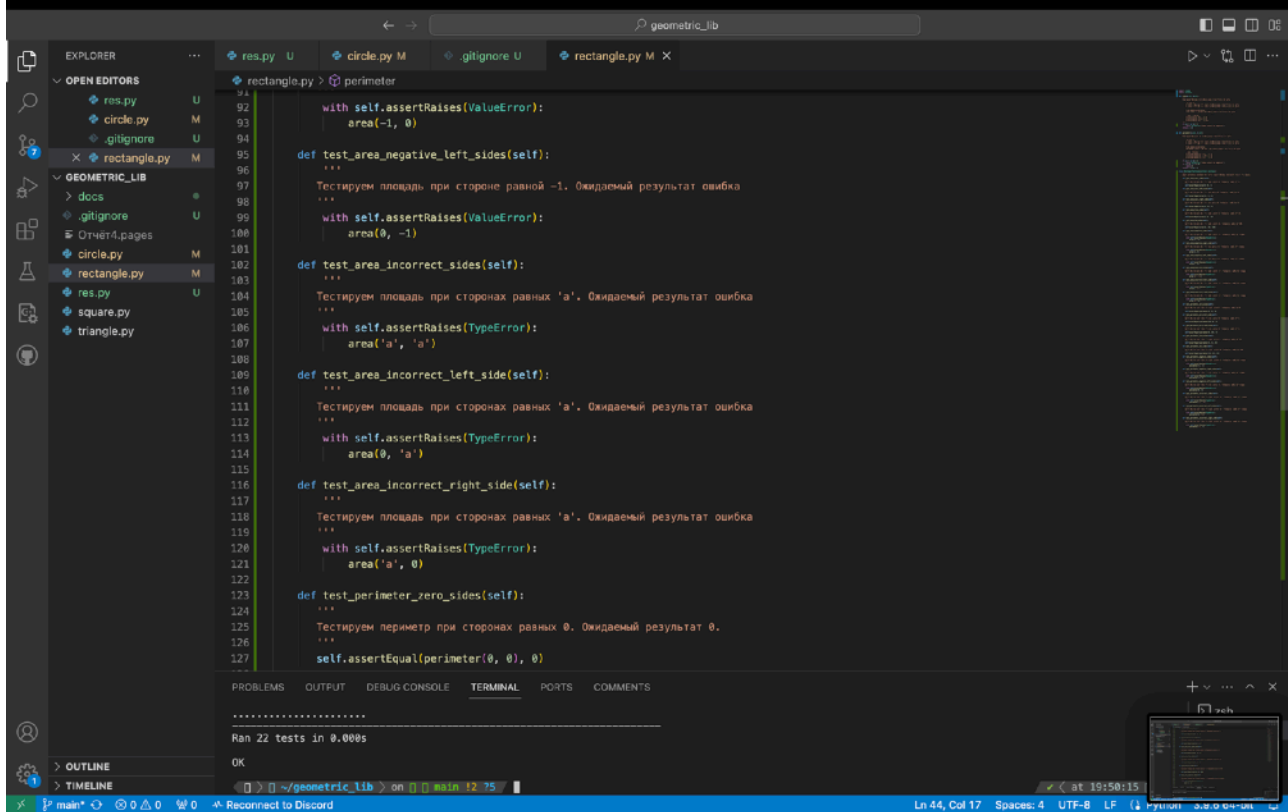
```
52 def test_perimeter(self):
53     """
54     Тестируем периметр при сторонах равных 0. Ожидаемый результат 0.
55     """
56     self.assertEqual(perimeter(0, 0), 0)
57
58 def test_area_zero_left_side(self):
59     """
60     Тестируем площадь при стороне равной 0. Ожидаемый результат 0.
61     """
62     self.assertEqual(area(10, 0), 0)
63
64 def test_area_zero_right_side(self):
65     """
66     Тестируем площадь при стороне равной 0. Ожидаемый результат 0.
67     """
68     self.assertEqual(area(0, 10), 0)
69
70 def test_area_five_sides(self):
71     """
72     Тестируем площадь при сторонах равных 5. Ожидаемый результат 25.
73     """
74     self.assertEqual(area(5, 5), 25)
75
76 def test_area_ten_sides(self):
77     """
78     Тестируем площадь при сторонах равных 10. Ожидаемый результат 100.
79     """
80     self.assertEqual(area(10, 10), 100)
81
82 def test_area_negative_sides(self):
83     """
84     Тестируем площадь при сторонах равных -1. Ожидаемый результат ошибка
85     """
86     with self.assertRaises(ValueError):
87         area(-1, -1)
88
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

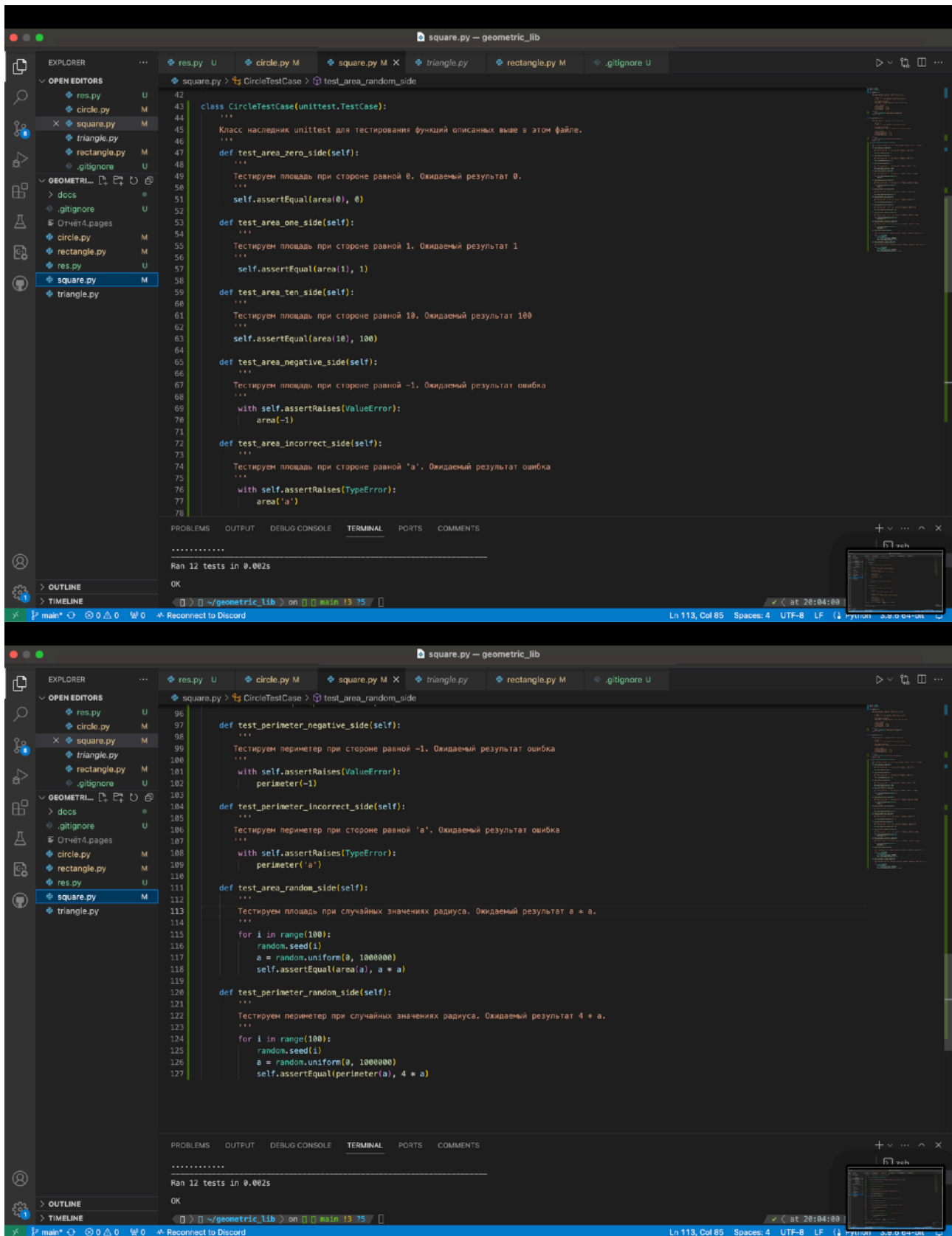
.....

Ran 22 tests in 0.000s

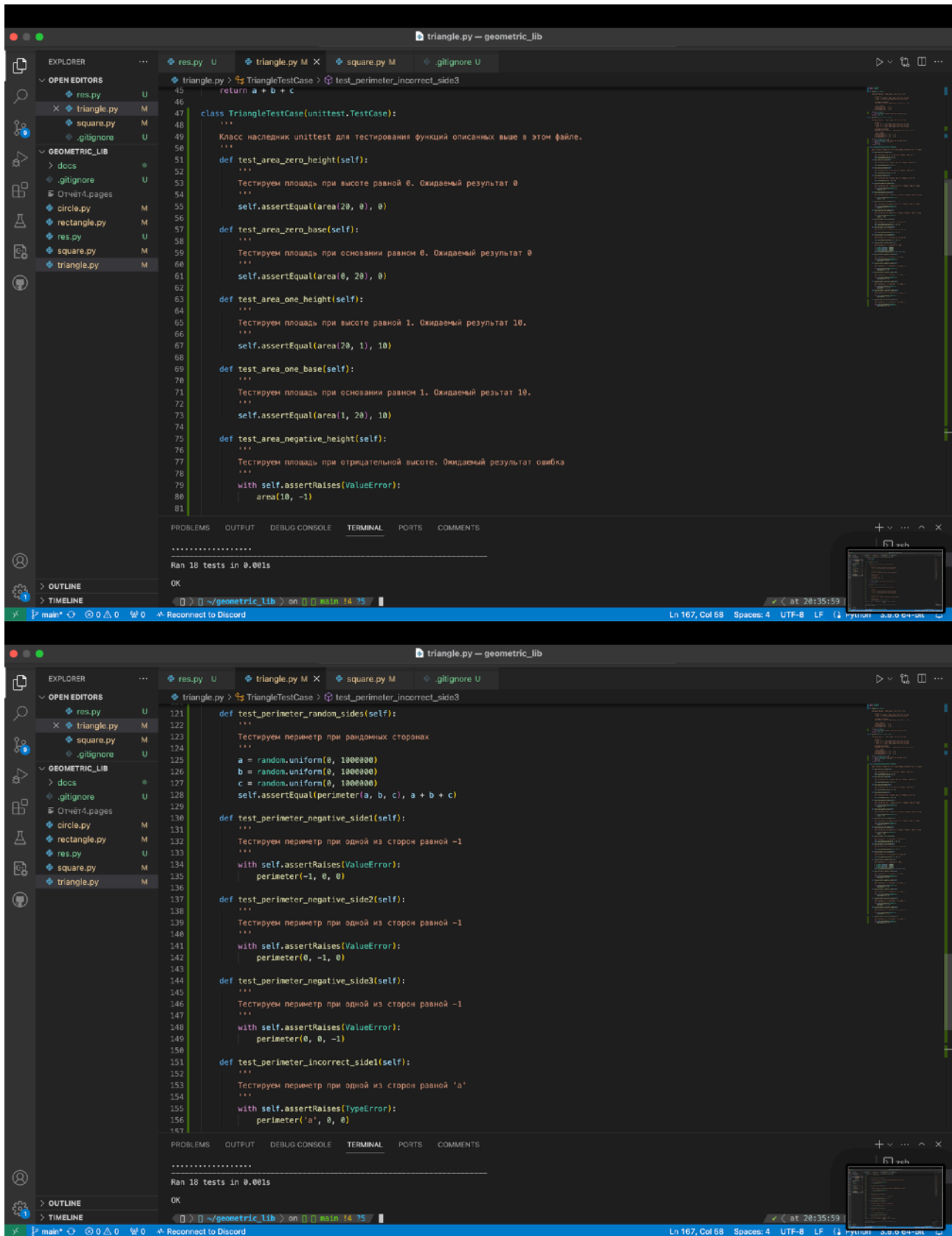
OK

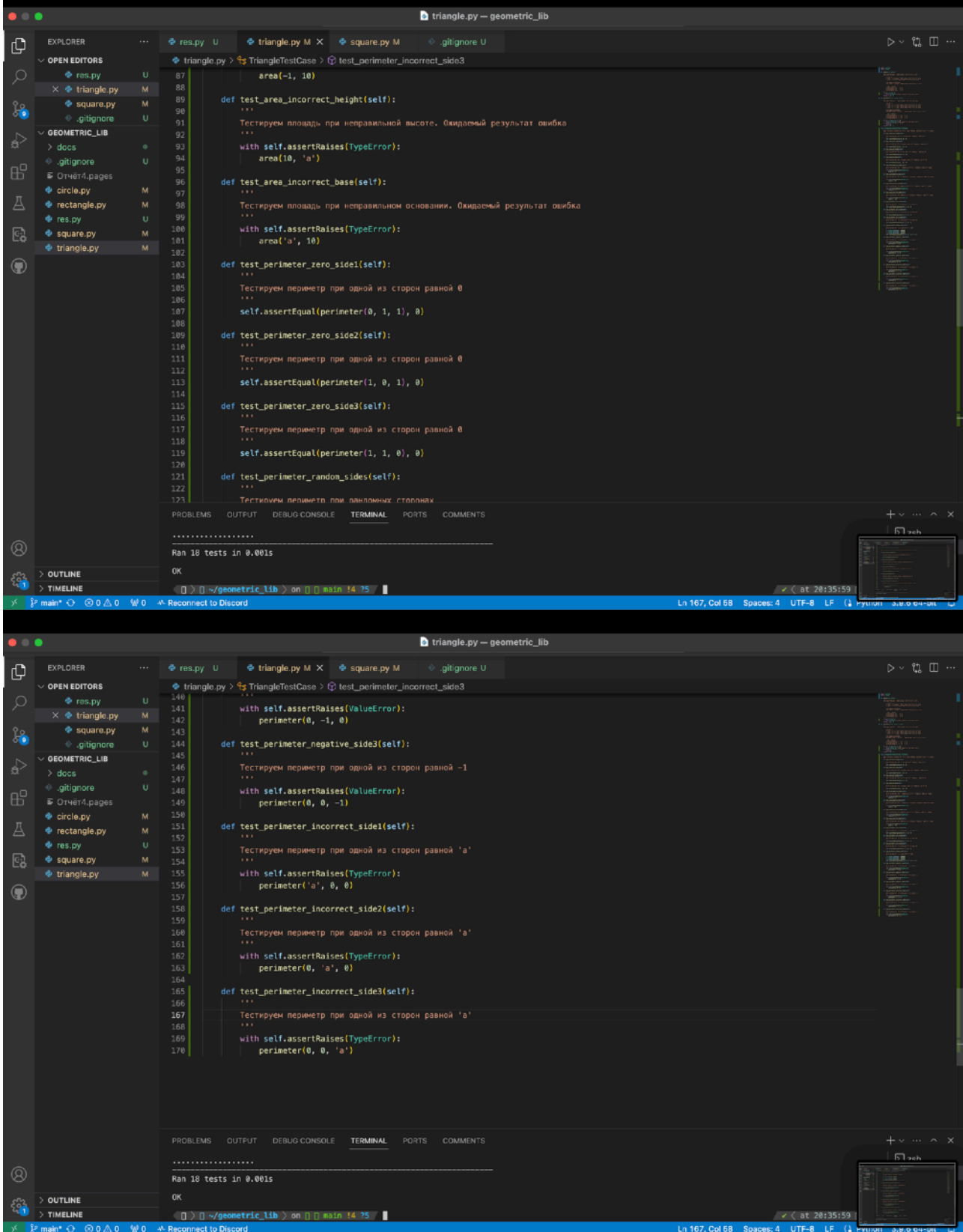


## 8. Теперь для square.py



## 9. Теперь для triangle.py





10. Теперь пропишем в документации о добавлении тестов



