```
Не все тесты пройдены, есть ошибки :(


Количество затраченных попыток: 22

Время выполнения: 1.867186 сек
```

Общая статистика

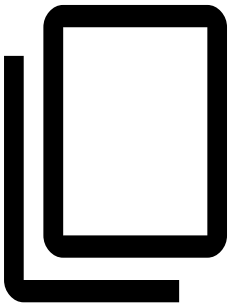
```
Всего тестов: 1. Пройдено: 0. Не пройдено: 1.

Подробную информацию по каждому тесту смотрите ниже.
```

Тест 1

Тест не пройден ✗

Формулировка:

** Итоговый код для проверки.*



```python
import warnings

warnings.filterwarnings('ignore')

# Введите ваше решение ниже
import doctest

class NegativeValueError(ValueError):
    pass

class Rectangle:
    """
    Класс, представляющий прямоугольник.

    Атрибуты:
    - width (int): ширина прямоугольника
    - height (int): высота прямоугольника

    Методы:
    - perimeter(): вычисляет периметр прямоугольника
    - area(): вычисляет площадь прямоугольника
    - __add__(other): определяет операцию сложения двух прямоугольников
    - __sub__(other): определяет операцию вычитания одного прямоугольника из другого
    - __lt__(other): определяет операцию "меньше" для двух прямоугольников
    - __eq__(other): определяет операцию "равно" для двух прямоугольников
    - __le__(other): определяет операцию "меньше или равно" для двух прямоугольников
    - __str__(): возвращает строковое представление прямоугольника
    - __repr__(): возвращает строковое представление прямоугольника, которое может быть
```

```
    """

    def __init__(self, width, height=None):
        if width <= 0:
            raise NegativeValueError(f'Ширина должна быть положительной, а не {width}')
        if height != None and height <= 0:
            raise NegativeValueError(f'Высота должна быть положительной, а не {height}')
        self._width = width
        if height is None:
            self._height = width
        else:
            self._height = height

    @property
    def width(self):
        return self._width

    @width.setter
    def width(self, value):
        if value > 0:
            self._width = value
        else:
            raise NegativeValueError(f'Ширина должна быть положительной, а не {value}')

    @property
    def height(self):
        return self._height

    @height.setter
    def height(self, value):
        if value > 0:
            self._height = value
        else:
            raise NegativeValueError(f'Высота должна быть положительной, а не {value}')

    def perimeter(self):
        """
        Вычисляет периметр прямоугольника.

        Возвращает:
        - int: периметр прямоугольника
        """
        return 2 * (self.width + self.height)

    def area(self):
        """
        Вычисляет площадь прямоугольника.

        Возвращает:
        - int: площадь прямоугольника
        """
        return self.width * self.height

    def __add__(self, other):
        """
        Определяет операцию сложения двух прямоугольников.

        Аргументы:
        - other (Rectangle): второй прямоугольник

        Возвращает:
        - Rectangle: новый прямоугольник, полученный путем сложения двух исходных прямоу
```

```
        """
        width = self.width + other.width
        perimeter = self.perimeter() + other.perimeter()
        height = perimeter / 2 - width
        return Rectangle(width, height)

    def __sub__(self, other):
        """
        Определяет операцию вычитания одного прямоугольника из другого.

        Аргументы:
        - other (Rectangle): вычитаемый прямоугольник

        Возвращает:
        - Rectangle: новый прямоугольник, полученный путем вычитания вычитаемого прямоуг
        """
        if self.perimeter() < other.perimeter():
            self, other = other, self
        width = abs(self.width - other.width)
        perimeter = self.perimeter() - other.perimeter()
        height = perimeter / 2 - width
        return Rectangle(width, height)

    def __lt__(self, other):
        """
        Определяет операцию "меньше" для двух прямоугольников.

        Аргументы:
        - other (Rectangle): второй прямоугольник

        Возвращает:
        - bool: True, если площадь первого прямоугольника меньше площади второго, иначе
        """
        return self.area() < other.area()

    def __eq__(self, other):
        """
        Определяет операцию "равно" для двух прямоугольников.

        Аргументы:
        - other (Rectangle): второй прямоугольник

        Возвращает:
        - bool: True, если площади равны, иначе False
        """
        return self.area() == other.area()

    def __le__(self, other):
        """
        Определяет операцию "меньше или равно" для двух прямоугольников.

        Аргументы:
        - other (Rectangle): второй прямоугольник

        Возвращает:
        - bool: True, если площадь первого прямоугольника меньше или равна площади второ
        """
        return self.area() <= other.area()

    def __str__(self):
        """
        Возвращает строковое представление прямоугольника.
```

```
        Возвращает:
        - str: строковое представление прямоугольника
        """
        return f"Прямоугольник со сторонами {self.width} и {self.height}"

    def __repr__(self):
        """
        Возвращает строковое представление прямоугольника, которое может быть использова

        Возвращает:
        - str: строковое представление прямоугольника
        """
        return f"Rectangle({self.width}, {self.height})"

    # Тесты:

    def test_width(self):
        '''
        Тестирование инициализации ширины. Созданы прямоугольники r1 с шириной 5 и r4
        с отрицательной шириной (-2). Убедимся, что r1.width корректно установлен на 5,
        а создание r4 вызывает исключение NegativeValueError с текстом Ширина должна быт
        а не -2

        >>> r1 = Rectangle(5)
        >>> r1.width
        5
        >>> # with NegativeValueError:
        >>> r4 = Rectangle(-2)  # r4 = Rectangle(2)   # Ширина должна быть положительной
        ...
        Ширина должна быть положительной, а не -2
        '''
        pass

    def test_height(self):
        '''
        Тестирование инициализации ширины и высоты. Созданы прямоугольники r2 с
        шириной 3 и высотой 4.
        Проверяем, что r2.width равно 3 и r2.height равно 4.
        При необходимости выбрасывать исклчение NegativeValueError с текстом
        Высота должна быть положительной, а не {value}
        >>> r2= Rectangle(3, 4)
        >>> r2.width
        3
        >>> r2.height
        4
        '''
        pass


    def test_perimeter(self):
        '''
        Тестирование вычисления периметра. Создан прямоугольник r1 с шириной 5 и
        проверяем, что r1.perimeter() возвращает 20. Также создан прямоугольник r2 с шир
        проверяем, что r2.perimeter() возвращает 14.
        >>> r1 = Rectangle(5)
        >>> r1.perimeter()
        20

        >>> r2 = Rectangle(3, 4)
        >>> r2.perimeter()
        14
        '''
```

```
        pass

    def test_area(self):
        '''
        Тестирование вычисления площади. Создан прямоугольник r1 с шириной 5 и
        проверяем, что r1.area() возвращает 25. Также создан прямоугольник r2 с шириной
        проверяем, что r2.area() возвращает 12.
        >>> r1 = Rectangle(5)
        >>> r1.area()
        25
        >>> r2 = Rectangle(3, 4)
        >>> r2.area()
        12
        '''
        pass


    def test_addition(self):
        '''
        Тестирование операции сложения. Созданы прямоугольники r1 с шириной 5 и r2 с
        шириной 3 и высотой 4.
        Выполняем операцию сложения r1 + r2 и проверяем, что полученный прямоугольник r3
        правильные значения ширины и высоты (8 и 6.0 соответственно).   ,, 6 9

        >>> r1 = Rectangle(5)
        >>> r2 = Rectangle(3, 4)
        >>> r31 = r1 + r2  #  r1.__add__(r2)
        >>> r31.width
        8
        >>> r31.height
        9.0
        '''
        pass


    def test_subtraction(self):
        '''
        Тестирование операции вычитания. Созданы прямоугольники r1 с шириной 5 и
        r2 с шириной 3 и высотой 4.
        Выполняем операцию вычитания r1 - r2 и проверяем, что полученный прямоугольник r
        правильные значения ширины и высоты (2 и 2.0 соответственно).

        >>> r1 = Rectangle(5)
        >>> r2 = Rectangle(3, 4)
        >>> r32 = r1 - r2      #   r1.__sub__(r2)
        >>> r32.width
        2
        >>> r32.height
        1.0
        '''
        pass



# Запускать тесты не надо, автотест это сделает сам:
# __file__ = None
# doctest.testmod(extraglobs={'__file__': __file__})

# __file__ = None
# doctest.testmod(extraglobs={'__file__': __file__})
```

```
import sys

# Открываем файл для записи
with open('pytest_output.txt', 'w') as file:
    # Перенаправляем stdout в файл
    sys.stdout = file

    # Запускаем pytest.main() с нужными параметрами
    __file__ = None

    doctest.testmod(extraglobs={'__file__': __file__})

# Возвращаем stdout в исходное состояние
sys.stdout = sys.__stdout__
# Считываем содержимое файла
with open('pytest_output.txt', 'r') as file:
    lines = file.readlines()
    #first_line = file.readline()
    #first_five_lines = lines[:1]


import re

file_name = "pytest_output.txt.txt"

# Открываем файл на чтение
with open('pytest_output.txt', "r") as file:
    # Считываем содержимое файла
    file_content = file.read()

# Используем регулярное выражение для удаления "line" и чисел после него
cleaned_content = re.sub(r'File "__main__", line \d+', '', file_content)

# Записываем обновленное содержимое обратно в файл
with open(file_name, "w") as file:
    file.write(cleaned_content)

with open(file_name, 'r') as new_file:
    file_contents = new_file.read()
    # Выводим содержимое файла на экран
    print(file_contents)
```

Ожидаемый ответ:

```
**********************************************************************
, in __main__.Rectangle.__add__
Failed example:
    r3.height
Expected:
    6.0
Got:
    9.0
**********************************************************************
, in __main__.Rectangle.__sub__
Failed example:
```

```
    r3.height
Expected:
    2.0
Got:
    1.0
**********************************************************************
2 items had failures:
   1 of   5 in __main__.Rectangle.__add__
   1 of   5 in __main__.Rectangle.__sub__
***Test Failed*** 2 failures.

Ваш ответ:

**********************************************************************
, in __main__.Rectangle.test_width
Failed example:
    r4 = Rectangle(-2)  # r4 = Rectangle(2)   # Ширина должна быть положительной, а не -2
Exception raised:
    Traceback (most recent call last):
      File "/usr/local/lib/python3.8/doctest.py", line 1336, in __run
        exec(compile(example.source, filename, "single",
      File "<doctest __main__.Rectangle.test_width[2]>", line 1, in <module>
        r4 = Rectangle(-2)  # r4 = Rectangle(2)   # Ширина должна быть положительной, а не -2
      File "1AS5QBLUJXSEO0L9FGQJ.py", line 35, in __init__
        raise NegativeValueError(f'Ширина должна быть положительной, а не {width}')
    NegativeValueError: Ширина должна быть положительной, а не -2
**********************************************************************
1 items had failures:
   1 of   3 in __main__.Rectangle.test_width
***Test Failed*** 1 failures.
```