ДЗ к 6.02: Классы. Наследование

Необходимо создать иерархию классов для различных типов геометрических фигур, включая круги, прямоугольники и треугольники, с использованием механизма наследования. Каждый класс должен содержать общие и специфические для фигуры атрибуты и методы. Также следует реализовать подклассы для отдельных вариаций фигур.

Требования к выполнению задания

1. **Создать базовый класс** `Shape`:
    - Включить общие атрибуты для всех фигур, такие как:
        - `color` (цвет фигуры)
        - `area` (площадь фигуры, вычисляемая в подклассах)
    - Реализовать общий метод `describe()`, который выводит информацию о фигуре.
2. **Реализовать подклассы для конкретных типов фигур**:
    - `Circle` с дополнительными атрибутами:
        - `radius` (радиус)
        - Метод `calculate_area()`, вычисляющий площадь круга.
    - `Rectangle` с атрибутами:
        - `width` (ширина)
        - `height` (высота)
        - Метод `calculate_area()`, вычисляющий площадь.
    - `Triangle` с атрибутами:
        - `base` (основание)

- `height` (высота)
- Метод `calculate_area()`, вычисляющий площадь по формуле:

3. **Создать подклассы для вариаций фигур**:
   - `Square` как подкласс `Rectangle`, у которого `width == height`.
   - `Parallelogram` как подкласс `Rectangle` с дополнительным углом наклона и своей формулой вычисления площади.
   - `IsoscelesTriangle` как подкласс `Triangle` с равными боковыми сторонами.
   - `EquilateralTriangle` как подкласс `IsoscelesTriangle`, у которого все стороны равны.
4. **Реализовать демонстрацию наследования**:
   - В родительских классах `Shape`, `Rectangle` и `Triangle` задать общие методы и свойства, которые переопределяются в дочерних классах.
   - Продемонстрировать возможность использования общих методов (например, `calculate_area()`) в подклассах.
5. **Создать и протестировать экземпляры классов**:
   - Создать несколько объектов различных классов и вызвать их методы.
   - Проверить, корректно ли работает наследование и переопределение методов.

Дополнительные требования

- Реализовать метод `__str__()` для вывода информации о каждом объекте.
- В случае некорректных значений (например, отрицательный радиус) выбрасывать исключения.
- Использовать аннотации типов (`type hints`) для всех методов.
- Установить black и flake8. Отформатировать black и избавиться от всех ошибок flake8.

Крутая статья про ООП

https://medium.com/data-bistrot/inheritance-in-python-object-oriented-programming-63bd93d7490c