ЗАдача 1. Создайте класс Airplane, который будет представлять самолет и поддерживать следующие операции:

1. **Сравнение типов самолетов** – реализуйте операцию ==, позволяющую проверять, относятся ли два самолета к одному типу.
2. **Изменение количества пассажиров** – реализуйте операторы +, -, +=, -=, чтобы можно было увеличивать или уменьшать число пассажиров на борту (при этом не превышая максимальную вместимость и не опускаясь ниже нуля).
3. **Сравнение самолетов по вместимости** – реализуйте операторы >, <, >=, <=, позволяющие сравнивать два самолета по их максимальному количеству пассажиров.

Задача 2. Создайте систему для управления заказами в ресторане, используя классы для моделирования блюд и заказов.

1. Реализуйте классы **Dish (блюдо)** и **Order (заказ)**.
Класс Dish должен содержать информацию о названии и цене блюда.

2. Класс **Order** должен хранить информацию о нескольких блюдах, которые входят в заказ.

3. Создайте несколько классов-наследников от Dish:
MainDish (основное блюдо), **Dessert (десерт)** и **Drink (напиток)**. Пусть у каждого класса будут специфические свойства (например, наличие алкоголя у напитков или вегетарианские блюда).

4. Класс Order должен уметь работать с любыми типами блюд, поддерживая добавление и удаление блюд различных типов в заказ.

5. Перегрузите оператор **+**, чтобы можно было объединять два объекта Order (суммировать заказы).

6. Перегрузите оператор **>,** чтобы сравнивать заказы по общей стоимости.

Задача 3. Создайте систему, моделирующей работу больницы.

В больнице есть врачи и пациенты.. Врачи имеют специализацию (терапевт, хирург, кардиолог) и могут выполнять процедуры для пациентов. Пациенты имеют заболевания, которые лечатся с помощью соответствующих процедур.

Необходимо реализовать систему, в которой:

1. Врачи и пациенты наследуются от одного базового класса **Person**.
2. Врачи разных специализаций выполняют уникальные процедуры для пациентов.
3. Реализована перегрузка операторов:
    - Оператор **>** для сравнения врачей по количеству выполненных процедур.
    - Оператор **+** для объединения пациентов в группу.
4. Вывод информации о врачах, пациентах и выполненных процедурах.

1. **Базовый класс Person:**
    - Атрибуты: name (имя), age (возраст).
    - Метод __str__ для вывода информации о человеке.
2. **Класс Patient (наследуется от Person):**
    - Дополнительный атрибут: disease (заболевание).
    - Перегрузка оператора + для объединения пациентов в группу.
3. **Класс Doctor (наследуется от Person):**
    - Дополнительный атрибут: specialization (специализация).
    - Атрибут procedures_performed (количество выполненных процедур).
    - Метод perform_procedure(patient) для выполнения процедуры.
    - Перегрузка оператора > для сравнения врачей по количеству выполненных процедур.
4. **Классы-наследники для врачей:**
    - Therapist (терапевт).
    - Surgeon (хирург).
    - Cardiologist (кардиолог).
    - Каждый класс должен переопределять метод perform_procedure() для выполнения уникальной процедуры.
5. **Класс PatientGroup:**
    - Представляет группу пациентов.
    - Атрибут patients (список пациентов).
    - Метод __str__ для вывода информации о группе.
6. **Класс Procedure:**
    - Атрибуты: name (название процедуры), doctor (врач), patient (пациент).
    - Метод __str__ для вывода информации о процедуре.