Exercise 1.5: Object-Oriented Programming in Python

Learning Goals

- Apply object-oriented programming concepts to your Recipe app

Reflection Questions

1. In your own words, what is object-oriented programming? What are the benefits of OOP?
   - Object-oriented programming is a method that uses objects and classes. The goal of using OOP is to bind data and functions together as a single unit.
   - The benefits of OOP are:
     - Modularity: Code can be organized into classes and objects
     - Reusability: Classes can be reused throughout the code reducing redundancy
     - Scalability: Makes expanding the project easier without having drastically modify existing code
     - Maintainability: Easier to maintain due to modularity
     - Encapsulation: Data inside classes is less susceptible to unexpected manipulation.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
   - Objects are variables that contain data and functions that can manipulate that data. Types of objects include: strings, lists, tuples, dictionaries, and more.
   - Classes are like templates used for creating objects in Python. Classes use the __intit__() function to assign values to the object.

   Example: A "Recipe" class for a recipe management app. The class might have attributes like name, ingredients, and instructions. Methods might include add_ingredient(), remove_ingredient(), and display_recipe(). An object of the Recipe class might be a specific recipe like "Pancake Recipe," with its unique ingredients and instructions.

3. In your own words, write brief explanations of the following OOP concepts; 100 to 200 words per method is fine.

Method	Description
Inheritance	In OOP when a new class, called a subclass, inherits existing attributes and methods from the superclass. This creates a hierarchical relationship between the classes. Subclasses can use or overwrite the superclass attributes.
Polymorphism	Different classes can be treated as different instances of the same class. It allows methods to perform differently based on the object that it is acting on while still sharing the same name. This is commonly done through overriding in a subclass.
Operator Overloading	Allows predefined operators to have additional and customized meanings when applied to objects. You can define how operators

	behave in your object. This allows for complex manipulations of objects.