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?
    a. OOP is the use of four key paradigms that make up the philosophy of that style of coding. Encapsulation, Abstraction, Inheritance and Polymorphism are the methods by which OOP is defined. Encapsulation is the bundling of data and methods into objects which makes code modular and easier to understand and maintain. Abstraction is defining abstract classes that define the behavior of properties of objects, which makes code more flexible and more adaptable. Inheritance allows you to create subclasses that inherit the properties of their parental classes and Polymorphism, which allows you to create objects that can take on multiple forms and behave in different ways depending on the context. The benefits are that OP makes your code more modular, flexible, maintainable, and easier to understand.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
    a. Objects are specific instances within a class. They represent entities that have attributes and behaviors which are defined in the class. When creating an object from a class, you are creating a unique copy of that class with its own data and functions. A class is the template or blueprints that defines the attributes and methods that will describe the behavior of an object. It encapsulates data and functions related to that data.
    b. An example is a class called "Humans" and an object being a specific, unique human. The object, for example, named Bob, has his own age, gender, hobbies, etc, but falls under the parent class of human, which shares certain attributes with all other humans.

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

Method	Description
Inheritance	See Q.1
Polymorphism	See Q.1
Operator Overloading	Operator overloading is a feature of OOP which allows operators to be redefined to work with custom data types or classes, by using __ __ on either side of the operator. Ex.

	__init__, __add__.