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?
    - OOP is a way of organizing code using objects, which are instances of classes. These classes contain objects that use attributes and methods and allow for concepts like inheritance, encapsulation, polymorphism, and abstractions. The use of OOP in coding and web development lends itself to make code more reusable, modular, and easier to maintain.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
    - A class is an overarching template describing an internal structure, while an object is an instance of a class. Objects are basically everything in Python and can be broken down into the data they contain.
    - To illustrate a real-world example, you could think of a quiver of skis. The quiver would be considered the class and everything in this quiver would be the objects. Examples being, how many sets of skis, their price, length, width, type, and even their purpose.

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

Method	Description
Inheritance	Inheritance is like passing down traits from parents to children. In programming, it means a class (the "child") can automatically use the properties and actions of another class (the "parent"). This saves you from repeating code and helps organize things better. For example, if you have a parent class called Animal that has a method called speak(), you can create a child class Dog that automatically gets this speak() method. You can also change or add new methods specific to Dog.
Polymorphism	Polymorphism is a fancy word that means "many shapes" or "forms." In programming, it allows different classes to have methods with the same name, but those methods can do different things depending on the object that's using them. It's a way to make sure that different objects can be used in similar ways, even if they behave a little differently.

	For example, you could have a Dog class and a Cat class, both of which have a speak() method. Even though the method name is the same, the dog will bark, and the cat will meow when you call it.
Operator Overloading	Operator overloading lets you define how operators like +, -, *, and others work for custom objects. Normally, operators are used for basic data types like numbers, but with operator overloading, you can teach Python how to handle them for your own objects. For example, the + operator usually adds numbers, but you can "overload" it to work with objects of a class, like adding two custom Point objects together by combining their coordinates.