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. Object-oriented programming is focused around creating and working with objects as opposed to functional programming that works with functions. Objects assemble real-world objects as a human would normally understand, such as a person, a book, a room, etc. Each object has its set of methods (like activity it can perform) and data (attributes that describe the object and its state). The benefits of OOP encapsulation of methods and data within classes, better organisation of code, it can be more understandable than functional programming due to its focus on objects resembling real world, reuse of code through inheritance, and modularity.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.

    a. Classes can be viewed as templates that prescribe what kind of data and methods are available to all objects of that particular class, what is the object's structure. Objects are created as instances of their class with their specific data values and with the methods defined in the class. For example, if we think of a bicycle, a class could be something like a template/blueprint of how a bicycle's structure should look like and what behaviors/functions it has, such as it has 2 wheels, a saddle, breaks, a color, and it can go straight, turn to the left, right, go faster, go slower, etc. It would be the basic description of any bicycle. Then a specific bicycle would have a saddle and breaks of a certain make, its own color, etc - that would be an object.

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

Method	Description
Inheritance	Taking over data and methods from a parent class so that they can be reused in a new subclass. A subclass can also implement additional data and methods but is able to utilize all of its parent class.
Polymorphism	When a method or a data attribute has the same name on different classes or data types but performs different actions. For example, the len() method on lists vs. strings.
Operator Overloading	Defining what a known operator should do with objects of a custom class. Such as, what the + operation should result into.