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? Programming using objects, with data and instructions, like components. What are the benefits of OOP? It keeps code clean and efficient and often speeds up creation of new code through use of inheritance.

2. What are objects and classes in Python? Objects are organized groupings of data and the operations that can be performed on that data. Everything in Python is an object. Classes are names given to a template for creating instances of an object that will use the same structure applied to its data, methods and consstructors. Come up with a real-world example to illustrate how objects and classes work.

   You can create a class of Dog, which you can then apply to any number of new instances of Dog. The class Dog will have an attribute for Breed, Name, and temperment.

```
class Dog(object):
  def __init__(self, breed, name, temperment):
    self.breed = breed
    self.name = name
    self.temperment = temperment
```

you can create a
dog1 = Dog("Golden Retriever, "Doug", "friendly")

you can now create many dogs with the same structure.

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

Method	Description
Inheritance	You can assign properties of one class to another class, quickly and efficiently, through inheritance. Instead of including "object" in parentheses, you can use another class and it's attributes and methods will automatically transfer to your new class. Your new class is the

	subclass which has inherited from the parent class also called the base class or superclass. Inheritance also only flows in one direction. A Parent class will not inherit from a subclass.
Polymorphism	This is a feature in object oriented programming that allows data attributes or methods to have the same name, yet different functionality across different classes, depending on where it gets defined. For example two sublasses of a superclass with the method speak() can each use that method to perform different implementation. This allows for greater flexibility and efficiency in your code.
Operator Overloading	This is the process in object oriented programing, of defining your own custom methods for already existing operators such as +, - , >, <+, etc. in your owm classes. You can then use operators to treat your objects like you would built-in data types. In other words you can add the value of objects the way you would add two integers.