### Challis Regan

### 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?**
   1. Object-oriented programming is a programming paradigm based on the concept of building objects, containing data and code, and organizing them with classes. OOP binds together the data and functions that operate on them, ensuring no other part of the code can access this data except that function. OOP makes code more efficient by keeping it non-repetitive and non-redundant.
2. **What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.**
   1. Almost everything in Python is an object, consisting of data and methods used to interact with the data. Objects are built and then organized with classes. A class constructs objects, like a blueprint for creating objects.
   2. Object, a single Party
   3. Party Class
      1. Data attributes
         1. Building where party is hosted
         2. Date & time
         3. Host(s)
         4. Guests
         5. Food
         6. Drinks
         7. Music
      2. Procedural attributes
         1. Method to invite or uninvite guests
         2. Method to select music, food or drinks
         3. Method to send party reminder(s) to guests
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 means inheriting methods from one class to another. The main class is the parent or base class and the class that inherits the procedures and data attributes is the subclass or inherited class. This helps keep the code non-repetitive because you do not need to re-type the parent class information in the subclass(es). You can add additional properties to the subclass that are not in the parent class. The simplest hierarchy is a parent class with a subclass underneath but you can extend the hierarchy to include more features exclusive to each subclass. Subclasses can have their own initialization method, inheriting parent class attributes and adding unique ones. |
| Polymorphism | Polymorphism means “many forms” and in Python OOP, it refers to operators/methods/functions with the same name that can be executed on many classes or objects and perform different operations. For example, you could make a parent class, “meal” and the “cook” method could be applied differently depending on what subclass is being cooked. It could be fried, baked or microwaved, and they would all be under the “cook” method. Whoever is calling the object doesn’t have to know exactly which object it is, it just needs to know what method is available to make it work. Having different objects do different things in response to the same functions. |
| Operator Overloading | Operator overloading is defining your own methods for using build-in operators on custom classes. The built-in operators work differently depending on what data type they are used with. The + operator adds numbers, but joins strings. The \* operator is the product of two numbers and it also repeats a string. It is replacing the default built in operators with two underscores on each side to customize it. Operator Overloading is good for printing a user-friendly string rather than an object in class. For example, if you want to increase the serving size of a recipe, you could overload the multiply to change the portion sizes, but not the cooking time or ingredient names. |