Reflection Questions Exercise 1.5

1. **In your own words, what is object-oriented programming? What are the benefits of OOP?**
   Object-oriented programming is structured on objects which are built through classes. It's a method of programming aimed at reducing repetitive code and maximizing efficiency. Classes are re-useable pieces of code that hold data and methods which can be applied to initialized objects as needed; this helps centralize data storage and re-use custom methods without re-writing them repeatedly.
2. **What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.**
   Objects are the building blocks of the code, each having its own data type and methods of interaction. Classes are the bigger template which may cover several objects of different data types. A real-world analogy could be a bedroom closet. The closet itself is the class, while the items in it are the data attributes. The attributes can be different kinds of "data type" - clothes, shoes, suitcases, etc. There are also various kinds of methods to interact with one class – taking items out of the closet, putting them in, making a list of the contents, etc.
3. **In your own words, write brief explanations of the following OOP concepts; 100 to 200 words per method is fine.**
   Inheritance
   Inheritance is a way to pass down properties or methods, from one class to another. By listing the "parent class" when initializing the "sub-class," the developer can access all methods and data attributes available in the original class. It only works in one direction, and if a method of the same name is listed in both classes, the one in the subclass will supersede the original.
   Polymorphism
   Polymorphism is when a data attribute or a method has the same name in different (not connected) classes or data types but performs a different function in each without conflict. For example, two classes can have a `speak()` method which will yield different results based on how it's defined in each class. While this can get confusing if the developer is not organized, it can also cut down on method name variations, resulting in cleaner, more readable code.
   Operator Overloading
   Operator overloading refers to the process of defining common operator functions for a custom class. Using the + operator on a custom class will result in an error message. First, the developer must define a function with a (pre-reserved by Python) name that will illustrate what that operator will do. Example: `__add__()` for addition, `__sub()__` for subtraction.