Exercise 1.5: Object-Oriented Programming in Python

1. **In your own words, what is object-oriented programming? What are the benefits of OOP?**

   Object-oriented programming is a type of coding that focuses on creating complex objects that can be reused throughout your code. By loading these objects with the data and methods needed you can keep your code non-repetitive and more efficient. OOP also lets you define your own classes, attributes, and methods allowing for a wide range of flexibility.

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

   Classes are the overarching structure that an object is based on, while an object is a specific instance of that class. If your class was a bridge for example, then draw-bridges, suspension bridges, and stone bridges would all be objects. They are all different, but still have the same attributes of crossing a gap, and a weight limit that the bridge “class” designates.

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 the ability to “inherit” or pass methods and properties from one class to another. This passing can only be done one way however, from the main/parent class to the subclass.
Polymorphism	Polymorphism is the ability for a certain method or data attribute that has the same name across multiple different classes to still have different operations. In other words, even if you have several different operations called check() for example, if they are in different classes they can all perform uniquely different operations independent of each other.
Operator Overloading	Operator Overloading is a way to use certain special operators Python has reserved. Because these certain operators don’t function in custom classes, we have to define them separately so Python knows what we want to do. An example would be replacing the + operator with __add__()