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

OOP is the practice of writing code focused on the creation and manipulation of objects, rather than focusing on other mediums, such as functions. This practice uses classes in order to create new objects. The practice of creating objects from classes is inherently reusable code, as each object simply needs to refer to the class, as opposed to having to define each unit/ object individually. This also produces code that is generally more self-contained than other coding practices, resulting in code that is easy to organize, understand, and maintain.

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

A class is a template that defines the structure, data attributes, and methods available to objects that are created using the class template. An object is a unit of code that contains data values passed through via the creation of the object from the class, or via methods designed to add data to the object. These 'setter' methods, along with other methods defined in the class, are available to objected initialized by the class.

A game in which users create characters could be a useful example of classes and objects. Characters would have an array of attributes which would be defined in the class, such as their name, character class, character race, ability scores, etc., as well as methods to change attributes or define character actions. When a user creates a character attributes would be passed into the object via the class initialization, and the result would be individual character objects based off of the general character class template.

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

Inheritance - Inheritance is the practice of classes assuming data attributes and methods from another class. When created, classes have an optional parameter where an existing class can be entered. This existing class is the 'parent class' and the new class is the 'sub class'. Objects initialized from the sub class can access data attributes and methods from the parent class without those attributes/methods being defined in the sub class. Attributes defined in the sub class would not be available to object initialized by the parent class, however, as inheritance only works in one direction.

Polymorphism - Polymorphism is where a data attribute or method has the same name in different situations, such as in different classes, or methods used on different data types, but produces different results. For example, a method defined in class A could have the same name as a method defined in class B. If both classes call their respective methods, different operations would be performed, and the result/outcome would be different for each class, despite each method having the same name.

Operator Overloading - Operator overloading is the process of defining specific functions within custom classes so that operators (both for arithmetic and logical comparison) can be used on custom classes. Without these functions operators such as '+' or '<' would produce a TyperError is used on two custom classes. In order to define one of these functions, a specific syntax has to be used. For example, to add the results of two custom classes together the __add__( ) function must be defined within the class. Each operator has its own corresponding function that must be defined in order to for the operator to be compatible with the class.