Reflection 1.5

1) Object Oriented Programming is when you code structure and flow creating objects with your data. Each object can be used to represent a block or section of data. This makes large amounts of data easily readable and accessible.

2) Classes are the more general term and hold basic information that all objects will need. Objects are pieces of data that fit in specific classes that bring more information that is unique to that specific object. An example of this could be for restaurants. The class would be called ‘restaurants’ and hold basic information like placeholder names, address, type of food. These can be identified when the actual restaurant object is inserted to have its own specific information. Any additional data on the restaurant can be added to the object.

3) Inheritance - Inheritance is similar to my answer in question 2. Classes can inherit specific traits or methods from each other. There can become a hierarchy of classes where each one becomes more detailed with more in depth information. The easiest example of this would be the animal/human class. There would be a larger class named ‘animal’ which would have basic information that all animals could relate to. Any animal could be filled in for this class, like a deer or an elephant. A smaller class can be made that would be called ‘humans’. This is a completely different class but it can be passed through ‘animal’ class to get all the methods there and then add on its own.

   Polymorphism - This is a super useful feature that allows different attributes or methods to share the same name. This is used when different classes use the same name but will give different responses. This is really good because it would be terrible if the same name always got confused with which class was being called. An example of how this could be helpful could be with restaurant classes. If two different restaurants had an attribute directions(), it would only be useful if they both gave their own unique response.

   Operator Overloading - This is when you are able to redefine how specific attributes work in code. This is helpful when you want a specific operator to behave exactly how you want it to. If you have a group of data with heights of individuals you can use operator overloading to change how the addition operator works. You can use **__add__** to change the heights and add it to inches before adding the heights together. This is one example of how you can change the way that it works.