Magnus Dahl-Hansen

Part 2

Questions

1. **What is the difference between a class and an object?**
   Answer:
   -A class is a blueprint/script which describes an object and its behaviours.

   -An object is a representation of a class containing all its values and methods.

2. **What is inheritance? What is the Python syntax for inheritance?**
   Answer:
   -Inheritance is a tool that enables a programmer to create a class that inherits all/parts of the functionality from another existing class. A class that inherits from another class is often called the child class, whilst the one from which it inherits is called a parent class.

   -Syntax for inheritance is using a parent class as an argument in a child class. Lets say we have a class "Parent" and a class "Child", then an example of inheritance would be: child = Child(Parent)

3. **What is the difference between a "has-a" and an "is-a" relationship?**
   Answer:
   -A "has-a" relationship is the attributes, values and methods that defines a class. For example, a class "*Person*" has a value stating its name.

   -A "is-a" relationship are values that another class inherits from an existing class. Let's say we have a class "*Worker*" that inherits attributes from a class "*Person*". This means that a *Worker* "is-a" *Person*, and that the *Worker* class is based on the *Person* class.

4. **What is encapsulation? How is encapsulation handled in Python?**
   Answer:
   -Encapsulation means to restrict access to variables and methods from a class. This tool is used to prevent a user from modifying data by accident but can be modified intentionally.

   -Encapsulation is handled by typing a single "_" or double "__" underscore in front of a method or variable.

5. **What is polymorphism? Give examples of polymorphism from the precode and the Mayhem implementation.**
   Answer:
   -Polymorphism means that something can have many forms and in Python this means that a method name can used more than once by different classes or types.

-The player classes are one example of polymorphism from my code. I have two different classes for the players, each containing the same method names. So, when I create instances of each class with different names, I can access methods inside these classes with the same name. Let's say each class has a method "move", so when I create instances I can write "*player1.move*" and "*player2.*move" and know which class the method belongs to.