Answers to part II of assignment 3

Håkon Silseth

UiT - Norges Arktiske Universitet

April 10, 2023

What is the difference between a class and an object?

The difference between an object and a class is that an object is an instance of a class. A class can be thought of as a blueprint for creating objects. To give an example from the Mayhem code Player is a class that describes the attributes and methods that an object of this class has. In the Game class, the Player class is instantiated to create player_1 which is an object. Player_1 has the attributes and methods that are layed out in the class Player.

What is inheritance? What is the python syntax for inheritance?

Inheritance is the process of borrowing attributes and methods from one class (parent class) to another class (child class). The reason we have for doing this is that we avoid repetition in our code. If we have multiple classes that have many similar attributes and methods, we can create a parent class for all of these, such that we avoid writing out all the shared attributes and methods for all the child classes. Also, if it becomes necessary to change an attribute or method, then we only need to do it once in the parent class, instead of doing it for each of the child classes separately. For simplicity lets say we have a class called Child that inherits from a class called Parent, the Python syntax for this becomes:

```
1 class Child(Parent):
2     def__init__(self):
3         super().__init__()
```

Listing 1: Example of inheritance in python

In this example we can see that we just include the name of the parent class in parentheses after the name of the child class. We also need to inherit the "__init__" function of the parent class, which is done in the 3rd line of the example.

What is the difference between a is-a and has-a relationship?

An is-a relationship is just the same as inheritance, to give an example imagine we have a class called Building and another named House. Logically House would be a child-class of Building since it is a type of building, therefore inheriting from the Building class. Also note that this is a uni-directional or a one-way relationship, a house is a building but a building is not necessarily a house. The easiest way to identify an is-a relationship is just to see if it makes sense to use the phrase "is a" when describing the relationship.
A has-a relationship is the same as composition, or using an instance of a class in another, this is also easiest to identify by seeing if we can use the phrase "has a" when describing the relationship. For an example of this we can again think of a house class and lets say a kitchen class. We can say that the house has a kitchen. In this example there is no inheritance here, only composition, where one object has references to another object. Unlike the is-a relationship a has-a relationship can be bi-directional, it goes both ways, this is not shown very good with the 'house' example, but we can show this quite simply with another example, think of a man and a dog, saying that the dog is the mans pet or the man is the dogs owner are equivalent statements.

What is encapsulation? How is encapsulation handled in Python?

Encapsulation is a fundamental concept in OOP, it is the process of containing data and the methods that work on the data in a separate unit. By 'encapsulating' the information like this we are both

keeping the information private and we avoid this data being altered by accident. Classes are used to encapsulate data in python. By creating a class we store the data and methods locally in the class and we can access them by calling on an object of the class. We can also create different levels of privacy in the class by adding one or two underscores before an attribute. By doing this we make it so that the attribute can't be accessed outside of the class. (one underscore make them protected: can be accessed in sub-classes) and two underscores make them private: can only be accessed in this specific class).

What is polymorphism? Give examples of polymorphism from the pre-code and the Mayhem implementation.

Polymorphism means that we have different functions with the same name. This is implemented when we have inheritance in a program, and both the child and the parent class have methods with the same name. Usually a child class will inherit all the methods of its parent class, but if we create a new method in the child class that shares a name with a class in the parent class, we will override the method from the parent class. This is very useful if we have a situation where the original method from the parent class is not ideal for the child class. Lets say we have a parent class for all moving objects, where all the objects will move in a straight line with constant speed. We then introduce a child class which moves with a constant acceleration and moves about in many directions. We can easily see here that the method used for moving the objects in the parent class won't work for the child class, so we need to redefine the method so it will work with nonlinear motion. In my Mayhem implementation polymorphism is not used, so theree are no examples to take from it.