**Why would you want to write your code in an object-oriented programming language?**

After delving into some research and taking a look at the uploaded PowerPoint slides, it shows that there are quite a few perks to using an object-oriented programming language (OOP). Here are some:

* It helps keep things organized and structured by using objects to bundle up data and functions.
* It makes handling big parts of code much easier thanks to its modular nature (in real-life objects).
* With support for inheritance, you can recycle code and scale up your projects by representing real-world stuff as objects.

**What's the difference between a class and an object?**

In the OOP language, a class acts as a sort of blueprint or template for creating objects. It lays out the traits (like fields, think of it as the stats in this Pokémon game assignment - nickname, strength, weakness) and behaviors (methods) that all objects of that class will have. Or to put it simply, a class is like a blueprint that sketches out how a specific type of object should look and act, without actually making any particular one.

An object, on the other hand, is like the actual information. It's an actual instance of a class, embodying all the characteristics and behaviors set out by that blueprint. Each object has its own set of data (the field values) and does its own thing when it comes to executing methods.

So, to summarize: a class sets the groundwork for what a type of object should be like, while an object is the real-life version of that type, complete with its own traits and actions. Think of a class as the plan for building something, and an object as the thing you've built based on that plan.