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

Object-oriented Programming (OOP) has several advantages that makes code reusable, easier to maintain and expandable. By organizing code into objects, OOP allows for better flexibility, making it easier to understand and modify. Data Protection (encapsulation) ensures that data is protected and only accessible through defined interfaces, reducing unintended interations. Inheritance promotes code reuse by allowing new classes to get functionality from existing ones, reducing repetition. Polymorphism (Versatility) enhances flexibility by enabling different objects to be treated equally. Additionally, OOP helps manage large projects efficiently by encouraging a structured approach to design, improving collaboration among developers, and supporting best practices in software engineering.

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

A **class** is a blueprint or template that defines the structure and behavior of objects. It specifies properties (attributes) and methods (functions) that it’s objects will have. A class itself doesn’t store data; it outlines what an object should contain.

An **object** is an instance of a class, meaning it is a concrete entity created based on the class definition. Objects have their own individual states and data but follow the rules set by the class. For example, if Car is a class, then myCar = Car("Red", "Toyota") is an object with specific attributes like color and brand.