2. What is OOP? List OOP concepts

Object-Oriented Programming (OOP) is a programming paradigm that uses "objects" to design applications and computer programs. Objects can be defined as instances of classes, which can encapsulate data and functions that operate on the data. The goal of OOP is to create reusable code that is more manageable, scalable, and easier to understand.

OOP Concepts:

1. ***Class:***
   - ***A class is a blueprint or template for creating objects. It defines a datatype by bundling data and methods that work on the data into one single unit.***
   - ***Example: A class Car might have attributes like color, make, and model, and methods like drive () and brake ().***
2. ***Object:***
   - ***An object is an instance of a class. It is a self-contained component that contains attributes and methods to manipulate those attributes.***
   - ***Example: An instance of the Car class could be a red Tesla Model S.***
3. ***Encapsulation:***
   - ***Encapsulation is the concept of wrapping the data (attributes) and the code (methods) together as a single unit. It restricts direct access to some of an object's components, which can prevent the accidental modification of data.***
   - ***Example: Using private variables in a class and providing public getter and setter methods.***
4. ***Abstraction:***
   - ***Abstraction is the concept of hiding the complex implementation details and showing only the necessary features of an object. It helps in reducing programming complexity and effort.***

- ***Example: A car's interface includes methods like start (), stop (), but the user does not need to understand the complex mechanics of how these methods work internally.***

5. ***Inheritance:***

- ***Inheritance is the mechanism by which one class can inherit attributes and methods from another class. This helps in code reusability and the creation of a hierarchical relationship between classes.***
- ***Example: A Sedan class could inherit from a Car class and add specific attributes or methods unique to sedans.***

6. ***Polymorphism:***

- ***Polymorphism means "many shapes" and it allows methods to do different things based on the object it is acting upon, even though they share the same name.***
- ***There are two types of polymorphism: compile-time (method overloading) and runtime (method overriding).***
- ***Example: A draw () method in a Shape class could be overridden by derived classes like Circle, Rectangle, etc., each implementing the draw () method differently.***