2.What is OOP? List OOP concepts

OOP stands for Object Oriented Programming. This is the type used in programming languages. If we understand it in simple language, then a language which requires objects to create a program or software is called object-oriented programming language. Like C++, Java and Python, these are object-oriented programming languages.

These languages work on the concept of object in which the object belongs to a class. Variables or properties of that class are called data members of that class and methods or functions of that class are called member functions of that class.

So, in OOP we see things as objects. An object consists of two things:

Data - This is the information that tells about that thing, say the speed of the vehicle or the balance of the bank account.
Methods - These are the things that the thing can do, say increasing the speed of the car or withdrawing money from the bank account.

In this way it becomes easier to write programs and also helps in reusing the code.

Well, let's understand all the important concepts of OOP in detail one by one:

1. Class:

You can consider a class as a blueprint of something, just like a blueprint is prepared before constructing a building. Similarly, in programming, a class is a kind of blueprint that decides what kind of objects can be created.

The class defines two things:

Data: This is the information that the object stores within itself, for example the color, model of the car or the bank account balance.

Functions: These are actions that the object knows how to perform, for example, starting the car, stopping it, or withdrawing money from a bank account.

Multiple objects can be created from the same class. All these objects will be similar because they are built from the same blueprint, but each object may have its own unique data. For example, from the same vehicle class you can create a red Swift Dezire and a blue Honda City.

2. Object:

The real-world use of a class is the object.

You can think of it as a particular example of something, let's say the red Swift Desire car parked in your garage is an object.

An object contains the same data that was defined in its class, in our example the car's color, model, and a few more.

Also, the object can use the same methods (functions) that were described in its class, that is, this vehicle can start, stop and move.

3. Encapsulation:

Encapsulation is a way to keep data secure.

Data in a class cannot be directly changed from other parts of the program.

Data can be changed only through special methods (functions) of that class.

These methods control what data can be changed and in what manner.
For example, setting the speed of the car directly in the code as car.speed = 100 will do nothing. Instead, we will have to use a function like vehicle.Speed_increase (10).

Only this method will decide whether the speed can be increased to 100 or not.
Encapsulation makes code robust and prevents mistakes.

4. Inheritance:

Sometimes two things have some similarities. By taking advantage of inheritance in OOP, we can reuse this similarity in the code.

In inheritance, one class adopts the properties of another class.

Suppose both car and sports car are objects. There is a lot of similarity between these two, both have data like color, model and speed and both can start, stop and run.

Using inheritance, we can create a Car class and then create a Sports Car class from it.

The Sports Car class will inherit all the properties from the Car class, such as color, model, speed, starting, stopping and Walking.

But the sports car class can also retain some of its special qualities, such as the way it increases speed.

Inheritance avoids writing double code and makes the code easier to manage.

5. Polymorphism:

Polymorphism means "many". That means one thing can appear in different forms. In OOP this means that the same method (function) can work in different ways for objects of different classes.

For example, suppose we have a Shape class and create subclasses from it like Square and Circle. Both inherit from the Shape class.

Now class Shape can have a method calculateArea() which tells how to calculate the area. But the method of calculating the area of square and circle will be different.

In polymorphism, this method (calculateArea()) is overridden in every subclass (Square and Circle) so that each object can calculate the area accordingly.

The developer just has to call the calculateArea() method, this method automatically understands for which object it has to work.

6. Abstraction:

Abstraction is a way of simplifying complex things in the real world.
In programming, this means that we see only those things which are important to us, and hide the rest.

For example, while driving a car, we do not need to know the details of how the engine works, we just need to know that the car starts, stops and moves.

Similarly, in a class we see only those methods (functions) that we want to use, we do not see how these methods manipulate the data inside.

Abstraction makes code easier to understand and allows developers to focus only on what's important.