What is OOP? List OOP concepts

**Ans**. OOP, or Object-Oriented Programming, is a programming paradigm that revolves around the concept of objects, which can contain data (attributes or properties) and code (methods or procedures). It aims to model real-world entities as software objects that interact with each other to solve problems. Here are the core concepts of Object-Oriented Programming:

1. **Class**: A blueprint or template for creating objects. It defines the attributes and methods that will be associated with objects created from that class.
2. **Object**: An instance of a class. It encapsulates data (attributes) and behavior (methods) as defined by its class.
3. **Encapsulation**: Encapsulation is the bundling of data (attributes) and methods that operate on the data into a single unit (class). It restricts direct access to some of an object's components and hides the internal state.
4. **Abstraction**: Abstraction involves hiding the complex implementation details and showing only the essential features of the object. It focuses on what an object does rather than how it does it.
5. **Inheritance**: Inheritance is a mechanism where a new class (derived class or subclass) is derived from an existing class (base class or superclass). The subclass inherits attributes and methods from its superclass and can also add new features.
6. **Polymorphism**: Polymorphism means the ability to take on multiple forms. In OOP, it allows objects of different classes to be treated as objects of a common superclass. Polymorphism can be achieved through method overriding (runtime polymorphism) or method overloading (compile-time polymorphism).

What is the difference between OOP and POP?

**Ans.**

The main difference between Object-Oriented Programming (OOP) and Procedural Programming (POP) lies in their fundamental approach to organizing and structuring code:

**Object-Oriented Programming (OOP):**

1. **Focus**: OOP focuses on modeling real-world entities as objects, which have attributes (data) and methods (functions or procedures).
2. **Data Abstraction**: Encapsulation hides the internal state of objects and provides access only through methods. Abstraction focuses on the essential features of an object, hiding implementation details.
3. **Code Reusability**: OOP promotes reuse through inheritance, where classes (blueprints for objects) can inherit attributes and behaviors from other classes.
4. **Polymorphism**: Objects of different classes can be treated as objects of a common superclass, allowing for flexibility and extensibility.
5. **Examples**: Java, C++, Python are languages that support OOP principles extensively. Classes and objects are fundamental concepts, and relationships between objects are crucial.

**Procedural Programming (POP):**

1. **Focus**: POP emphasizes procedures or functions that operate on data. It follows a linear top-down approach where the program is divided into a sequence of functions.
2. **Data and Functions**: Data and functions are separate entities, and functions may operate on shared data stored in global variables.
3. **Code Reusability**: Code reuse is achieved through functions, but it typically involves copying and pasting code rather than through mechanisms like inheritance.
4. **Sequential Execution**: Programs are executed sequentially, and control flow is managed through function calls and jumps.
5. **Examples**: C, Fortran, and Pascal are examples of languages that follow procedural programming paradigms. Functions are central, and data is typically managed using structures and arrays