Basic Concepts of OOP

2. What is OOP? List OOP concepts

OOP stands for Object-Oriented Programming. It is a programming paradigm based on the concept of "objects," which can contain data (attributes) and code (methods). OOP aims to structure software in a way that models real-world entities and their interactions.

Here are some key OOP concepts:

1. **Class**: A blueprint for creating objects. It defines the attributes (data) and methods (functions) that all objects of that class will have.
2. **Object**: An instance of a class. It represents a specific entity with its own unique data and behaviour.
3. **Encapsulation**: The bundling of data (attributes) and methods (functions) that operate on the data into a single unit (class). Encapsulation helps in hiding the internal workings of an object and only exposing the necessary interfaces.
4. **Inheritance**: A mechanism where one class (subclass/derived class) inherits properties and behaviour from another class (superclass/base class). It promotes code reusability and establishes an "is-a" relationship between classes.
5. **Polymorphism**: The ability of objects to take on multiple forms or behave differently based on the context. Polymorphism can be achieved through method overloading (having multiple methods with the same name but different parameters) and method overriding (redefining a method in a subclass).
6. **Abstraction**: The process of hiding complex implementation details and exposing only the essential features of an object. Abstraction allows developers to focus on what an object does rather than how it does it.

These OOP concepts form the foundation of many modern programming languages such as C++, Java, Python, and more. They help in organizing code, improving code reusability, and creating modular and maintainable software systems.

3. What is the difference between OOP and POP?

OOP (Object-Oriented Programming) and POP (Procedural-Oriented Programming) are two different programming paradigms that guide how software is structured and organized. Here are the key differences between OOP and POP:

1. **Organization**:
   - OOP organizes code around objects, which are instances of classes containing both data (attributes) and behaviour (methods/functions).
   - POP organizes code around procedures or functions, where the program is structured as a series of sequential steps that manipulate data.
2. **Data Handling**:
   - OOP emphasizes data encapsulation, where data and methods that operate on the data are bundled together within objects. Objects communicate with each other by sending messages.
   - POP relies on data being shared between procedures/functions through parameters and global variables. Data and functions are separate, and functions can modify data directly.
3. **Code Reusability**:
   - OOP promotes code reusability through inheritance and polymorphism. Inheritance allows new classes to inherit properties and behaviour from existing classes, while polymorphism enables objects to take on multiple forms.
   - POP typically involves writing procedures or functions specific to a task, and code reusability may be achieved through modularization (breaking code into reusable modules) but without the same level of inherent reusability as OOP.
4. **Focus**:
   - OOP focuses on modelling real-world entities using objects, fostering a more intuitive and modular approach to software design.
   - POP focuses on breaking down a problem into a series of steps or procedures, often used in more straightforward tasks or when performance optimization is a priority.
5. **Complexity**:
   - OOP can handle complex systems more effectively by providing mechanisms like abstraction, encapsulation, inheritance, and polymorphism.
   - POP is generally simpler and may be preferred for smaller projects or tasks where the procedural nature aligns well with the problem domain.

In summary, OOP is cantered around objects, data encapsulation, inheritance, and polymorphism, making it suitable for modelling complex systems and promoting code reusability. On the other hand, POP revolves around procedures/functions, sequential code execution, and may be favoured for simpler tasks or when performance is critical.