**Basic Concepts of OOP.**

**1. What is OOP? List OOP concepts.**

OOP stands for Object-Oriented Programming. It's a programming paradigm that revolves around the concept of "objects", which are instances of classes, and the interactions between these objects. OOP focuses on organizing code into reusable and modular components, making it easier to manage and maintain large-scale software systems.

In C++, some of the key OOP concepts include:

* Class
* Object
* Encapsulation
* Data Abstraction
* Inheritance
* Polymorphism
* Data binding

**1**. **Class:** A blueprint for creating objects. It defines the properties (data members) and behaviours (member functions) that all objects of that class will have.

**2**. **Object:** An instance of a class. It encapsulates data for the object and provides methods to interact with that data.

**3**. **Encapsulation**: Encapsulation is the bundling of data (attributes) and methods (functions) that operate on the data into a single unit (i.e., class). It hides the internal state of an object from the outside world and only exposes the necessary functionalities.

**4**. **Inheritance**: Inheritance allows a class (subclass/derived class) to inherit properties and behaviours from another class (superclass/base class). This promotes code reuse and establishes a hierarchical relationship between classes.

**5**. **Polymorphism**: Polymorphism means "many forms". It allows objects of different classes to be treated as objects of a common superclass. Polymorphism can be achieved through function overloading, function overriding, and abstract classes.

**6**. **Abstraction:** Abstraction refers to the process of simplifying complex systems by hiding unnecessary details while exposing only the essential features of an object. It helps in managing complexity by focusing on the relevant aspects of objects.

1. **What is the difference between OOP and POP?**

| **OOP** | **POP** |
| --- | --- |
| Object oriented. | Structure oriented. |
| Program is divided into objects. | Program is divided into functions. |
| Bottom-up approach. | Top-down approach. |
| Inheritance property is used. | Inheritance is not allowed. |
| Encapsulation is used to hide the data. | No data hiding. |
| Concept of virtual function. | No virtual function. |
| Adding new data and functions is easy | Expanding new data and functions is not easy. |
| The existing code can be reused. | No code reusability. |
| use for solving big problems. | Not suitable for solving big problems. |
| C++, Java. | C, Pascal. |