MODULE: 4.1

Q.2 • What is OOP? List OOP concepts.

Ans-. Object-Oriented Programming (OOP) is a programming paradigm based on the concept of "objects." Objects are instances of classes, which can be thought of as blueprints for creating specific instances with defined attributes (data) and methods (functions or procedures).

**Key Concepts of OOP:**

1. **Classes and Objects**:

**Class**: A blueprint or template for creating objects. It defines a set of attributes and methods that the created objects will have.

**Object**: An instance of a class. It contains actual values for the attributes defined by the class.

1. **Encapsulation**:

The bundling of data (attributes) and methods (functions) that operate on the data into a single unit or class. It restricts direct access to some of the object's components, which can prevent the accidental modification of data.

1. **Inheritance**:

A mechanism where a new class (child or derived class) is derived from an existing class (parent or base class). The child class inherits attributes and methods from the parent class, allowing for code reuse and the creation of more complex hierarchies.

1. **Polymorphism**:

The ability to present the same interface for different data types. In OOP, it often refers to the ability of different objects to respond to the same method call in a way that is appropriate to their type.

1. **Abstraction**:

The concept of hiding the complex implementation details and showing only the essential features of the object. It helps in reducing programming complexity and effort.

Q.3 What is the difference between OOP and POP?

Ans. Object-Oriented Programming (OOP) and Procedural-Oriented Programming (POP) are two different programming paradigms with distinct approaches to program design and development. Here are the key differences between OOP and POP:

**Procedural-Oriented Programming (POP):**

1. **Program Structure**:

Programs are structured into procedures or routines (also known as functions or subroutines).

Emphasizes a step-by-step sequence of instructions to perform tasks.

1. **Data and Function Separation**:

Data and functions are separate entities.

Functions operate on data, but data and operations on data are not bundled together.

1. **Modularity**:

Modularity is achieved through the use of functions.

Functions can be reused but often need careful handling of global data.

1. **Data Handling**:

Data is often shared across functions through global variables.

Global data can be accessed and modified by any function, potentially leading to unintended side effects.

1. **Examples**:

Commonly used in languages like C, Pascal, and early versions of BASIC.

1. **Focus**:

Focuses on procedures (functions) that act on data.

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

1. **Program Structure**:

Programs are structured into objects.

Objects are instances of classes that bundle data and functions (methods) together.

1. **Data and Function Encapsulation**:

Encapsulation bundles data and the methods that operate on that data into a single unit (class).

Promotes data hiding and abstraction.

1. **Modularity**:

Modularity is achieved through classes and objects.

Classes can be reused and extended through inheritance.

1. **Data Handling**:

Data is usually private to the object and accessed through methods, enhancing data security and integrity.

Reduces the risk of unintended side effects since data manipulation is controlled.

1. **Examples**:

Commonly used in languages like Java, C++, Python, and C#.

1. **Focus**:

Focuses on objects that represent real-world entities and their interactions.

Key Differences:

1. **Abstraction Level**:

POP: Lower level of abstraction. Focuses on functions and sequence of actions.

OOP: Higher level of abstraction. Focuses on modeling real-world entities through objects.

1. **Modularity and Reusability**:

POP: Reusability is limited to functions. Modularity is less flexible.

OOP: Reusability is enhanced through inheritance and polymorphism. Modularity is more robust due to encapsulation.

1. **Ease of Maintenance**:

POP: Harder to maintain and extend, especially for large programs.

OOP: Easier to maintain and extend due to the organized structure and encapsulation.

1. **Code Readability and Complexity Management**:

POP: Code can become complex and difficult to manage as the program grows.

OOP: Code is generally more readable and easier to manage, especially for complex applications.

1. **Data Security**:

POP: Data is less secure due to global variables and the lack of data hiding.

OOP: Data is more secure due to encapsulation and access control.