MODULE 4: OOP Concepts

Q2. What is OOP ? List of concepts.

Ans : Object-oriented programming (OOP) is defined as a programming paradigm (and not a specific language) built on the concept of objects, i.e., a set of data contained in fields, and code, indicating procedures – instead of the usual logic-based system.

Key Concepts of OOP

1. **Classes and Objects**:

    - **Class**: A blueprint for creating objects. It defines a datatype by bundling data and methods that work on the data into one single unit.
    - **Object**: An instance of a class. Objects can contain data in the form of fields (often known as attributes or properties) and code, in the form of procedures (often known as methods).

2. **Encapsulation**:

    - The concept of bundling the data (variables) and code (methods) together as a single unit. Encapsulation hides the internal state of the object from the outside world, only exposing a controlled interface. This is achieved by making some of the object's fields private while providing public methods to access and modify those fields.

3. **Inheritance**:

    - A mechanism by which one class can inherit the attributes and methods from another class. This allows for code reuse and the creation of a hierarchical relationship between classes. The class that inherits is called the child or derived class, and the class being inherited from is the parent or base class.
    - There are 5 types:

        1. Single.
        2. Multiple.
        3. Multilevel.
        4. Hierarchical.
        5. Hybrid.

4. **Polymorphism**:

    - The ability of different objects to respond to the same message (or method call) in different ways. It allows for one interface to be used for a general class of actions, with the specific action determined by the exact nature of the situation. Polymorphism is typically achieved through method overriding (a subclass provides a specific implementation of a method already defined in its superclass) and method overloading (methods have the same name but differ in the type or number of parameters).

5. **Abstraction**:

    - The concept of hiding the complex implementation details and showing only the necessary features of an object. Abstraction allows the programmer to focus on interactions at a high level, without needing to know the intricate details of the implementation. It can be achieved using abstract classes and interfaces.

Q2. What is difference between OOP and POP ?

Ans: Object-Oriented Programming (OOP) :-

1. **Focus**:

    - OOP focuses on objects that represent real-world entities and are organized into classes.
    - Emphasizes data encapsulation, inheritance, and polymorphism.

2. **Structure**:

    - Programs are structured around objects and their interactions.
    - Objects are instances of classes and contain both data and methods.

3. **Data and Functions**:

    - Data and functions are bundled into objects.
    - Data is typically hidden and protected, accessible only through methods (encapsulation).

4. **Code Reusability**:

    - Promotes code reusability through inheritance and polymorphism.
    - Classes can inherit attributes and methods from other classes, allowing for hierarchical relationships and code reuse.

5. **Abstraction**:

    - Emphasizes abstraction by allowing complex details to be hidden behind simpler interfaces.
    - Uses abstract classes and interfaces to define common behaviors.

6. **Design**:

    - Encourages design patterns and principles that support modular, scalable, and maintainable code.
    - Examples include MVC (Model-View-Controller) and SOLID principles.

7. **Examples of OOP Languages**:

    - Java, C++, Python, Ruby, C#, etc.

Procedural-Oriented Programming (POP) :-

1. **Focus**:

    - POP focuses on procedures or routines (functions) that operate on data.
    - Emphasizes the sequence of actions to be performed.

2. **Structure**:

    - Programs are structured as a sequence of instructions or function calls.
    - Functions are the primary building blocks and are called to perform operations on data.

3. **Data and Functions**:

    - Data and functions are separate entities.
    - Data is often exposed and can be accessed directly by functions.

4. **Code Reusability**:

    - Encourages code reusability through function calls and libraries.
    - Functions can be reused, but there's no inherent mechanism like inheritance.

5. **Abstraction**:

    - Abstraction is achieved through function calls.
    - Does not inherently support complex abstraction mechanisms like classes and objects.

6. **Design**:

    - Encourages a top-down approach to design, breaking down tasks into smaller functions.
    - Focuses on step-by-step instructions and flow control.

7. **Examples of POP Languages**:

    - C, Pascal, Fortran, Basic, etc.

.