Assignment Module: 4

Basic concepts of OOP.

2. What is OOP? List OOP concepts.

Ans. The full form of OOP is Object-Oriented Programming. It’s a programming framework that revolves around the concept of objects, which can contain data in the form of attributes or properties and code in the form of procedures(methods or functions).

Fundamental concepts of OOP:

1. Class: A blueprint for creating objects. It defines the attributes and methods common to all objects of a kind.
2. Object: An instance of a class. It’s concrete entity created based on the blueprint provided by the class.
3. Encapsulation: The bundling of data (attributes) and methods (functions) that operate on the data into a single unit (class). It hides the internal state of an object and requires all interactions to be performed through well-defined interfaces (methods).
4. Inheritance: A mechanism that allows a class (subclass or derived class) to inherit properties and behavior from another class (superclass or 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 have multiple behaviors based on their context. It enables functions or methods to be written to operate on objects of a superclass and then be redefined in subclasses to provide specialized implementations.
6. Abstraction: The process of simplifying complex reality by modeling classes appropriate to the problem, while hiding unnecessary details from the user. It allows programmers to focus on relevant aspects of an object and ignore irrelevant details.
7. Association: A relationship between two or more classes where objects of one class are connected with objects of another class. It can be one-to-one, one-to-many, or many-to-many.
8. Composition: A form of association where one class contains objects of another class as part of its state. The lifetime of the contained objects is controlled by the container object.
9. Aggregation: A specialized form of association where one class contains objects of another class, but the contained objects may exist independently of the container object.

3. What is the difference between OOP and POP?

Ans. OOP (Object-Oriented Programming) and POP (Procedural-Oriented Programming) are two different programming paradigms, each with its own approach to structuring and organizing code. Here are the main differences between the two:

1. Data and Code Organization:

* OOP: Organizes code around objects, which encapsulate data (attributes) and behavior (methods). Objects interact with each other by sending messages.
* POP: Organizes code around procedures or functions. Data and functions are kept separate and operate on data using sequential execution.

1. Abstraction:

* OOP: Emphasizes abstraction through classes and objects, allowing complex systems to be modeled by defining objects that represent real-world entities.
* POP: Uses procedures or functions to break down tasks into smaller, more manageable components. Abstraction is achieved through the decomposition of tasks into smaller functions.

1. Encapsulation:

* OOP: Encapsulates data and behavior within objects, hiding the internal state of objects and exposing only the necessary interfaces for interacting with them.
* POP: Does not inherently support encapsulation. Data and functions are typically separate, and there's less emphasis on hiding data.

1. Inheritance:

* OOP: Supports inheritance, allowing classes to inherit properties and behavior from other classes, promoting code reuse and establishing hierarchical relationships.
* POP: Does not support inheritance as a built-in feature. Code reuse is achieved through procedural decomposition, where functions can be reused across different parts of the program.

1. Polymorphism:

* OOP: Supports polymorphism, enabling objects to exhibit different behaviors based on their types or classes. This allows for flexible and extensible code.
* POP: Achieves polymorphism through function overloading and parameter passing techniques but doesn't have built-in support for polymorphism as in OOP.

1. Example Languages:

* OOP: Languages like Java, C++, Python, and Ruby are popular for OOP.
* POP: Languages like C, Fortran, and Pascal are often used for procedural programming, although they may support other paradigms as well.

In summary, OOP focuses on organizing code around objects, promoting encapsulation, inheritance, and polymorphism, while POP organizes code around procedures or functions, emphasizing sequential execution and procedural decomposition of tasks. Each paradigm has its strengths and weaknesses, and the choice between them depends on the requirements of the project and the preferences of the programmer.