Question 1 [5]:

What is Object-Oriented Programming (OOP)?

Answer1: Object-Oriented Programming (OOP) is a programming paradigm that organizes code into objects, which are instances of classes. It emphasizes the concept of "objects" that encapsulate data and behavior. OOP principles include encapsulation, inheritance, and polymorphism, providing a modular and organized approach to software development.

Question 2 [5]:

Explain Encapsulation in OOP.

Answer2: Encapsulation is the OOP principle that involves bundling data (attributes) and methods (functions) that operate on the data into a single unit known as an object. It helps in hiding the internal details of an object and restricting direct access to its implementation. Encapsulation enhances code organization, reduces complexity, and promotes data integrity by controlling access to the internal state of objects.

Question 3 [2]:

What is Inheritance in OOP?

Answer3: Inheritance is a fundamental OOP concept that allows a new class (subclass or derived class) to inherit attributes and behaviors from an existing class (superclass or base class). This promotes code reusability and establishes a hierarchy of classes. The subclass can extend or override the functionality of the superclass while inheriting its characteristics.

Question 4 [2]:

Describe the concept of Polymorphism in OOP.

Answer4: Polymorphism in OOP refers to the ability of objects to take on multiple forms or the ability of a method to perform different actions based on the object it is acting upon. This can be achieved through method overloading (same method name, different parameters) and method overriding (same method name and parameters, different implementation). Polymorphism enhances flexibility and adaptability in code design.

Question 5 [2]:

What is Abstraction in the context of Object-Oriented Programming?

Answer5: Abstraction is the process of simplifying complex systems by modeling classes based on the essential properties and behaviors relevant to the problem at hand, while ignoring unnecessary details. It involves creating abstract classes or interfaces that define a common structure without specifying the implementation. Abstraction helps in managing complexity, improving code readability, and facilitating code maintenance.