Day 5: Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) is a programming approach that organizes software design around **objects** rather than functions and logic. It improves code readability, reusability, scalability, and maintenance by modeling real-world entities.

1. Classes & Objects

Class

A **class** is a blueprint or template used to define the structure and behavior of objects. It specifies what **attributes** and **methods** an object created from the class will have.

Key Characteristics of a Class:

- Acts as a logical framework
- Does not occupy memory until an object is created
- Helps group related data and functionality

Object

An **object** is a real-time instance of a class. It represents a specific entity with actual values for the attributes defined in the class.

Key Characteristics of an Object:

- Occupies memory
- Can access class attributes and methods
- Multiple objects can be created from a single class

2. Attributes & Methods

Attributes

Attributes are variables that store information about an object or a class.

Instance Variables

- Belong to individual objects
- Each object has its own copy
- Used to store object-specific data

Class Variables

- Shared among all objects of the class
- Only one copy exists regardless of the number of objects
- Used to store common data

Methods

Methods are functions defined within a class that describe the behavior of objects.

Types of Methods:

- Instance methods: Operate on object data
- Class methods: Operate on class-level data
- Static methods: Independent of class and object data

3. Constructor (__init__ Method)

A **constructor** is a special method that is automatically executed when an object is created.

Purpose of Constructor:

- Initializes object attributes
- Assigns initial values to instance variables
- Ensures the object starts in a valid state

self Parameter

- Refers to the current object
- Helps distinguish instance variables from local variables
- Allows access to object attributes and methods

4. Inheritance

Inheritance is a mechanism that allows one class (child class) to acquire the properties and behaviors of another class (parent class).

Advantages of Inheritance:

- Promotes code reusability
- Reduces redundancy
- Establishes a logical relationship between classes
- Simplifies code maintenance

Types of Inheritance:

- Single Inheritance
- Multiple Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

super() Function

The super() function is used to access methods and attributes of the parent class from the child class. It ensures proper initialization and method execution in inheritance hierarchies.

5. Encapsulation

Encapsulation is the process of wrapping data and methods together into a single unit (class).

Key Concepts of Encapsulation:

- Restricts direct access to data
- Protects object integrity
- Improves security and control

Private Attributes

- Intended to be accessed only within the class
- Prevent accidental modification from outside the class

Getters and Setters

- Getter methods retrieve private data
- Setter methods modify private data safely
- Provide controlled access to class attributes

6. Special Methods

Special methods are predefined methods that allow objects to interact with built-in functions and operators.

__str__ Method

- Provides a readable string representation of an object
- Used for user-friendly output

__repr__ Method

- Provides an official representation of an object
- Mainly used for debugging and development

__len__ Method

- Returns the length or size of an object
- Allows objects to work with length-based operations