Object-Oriented Programming (OOP) in Python

What is OOP?

- • A programming approach using objects
- • Helps structure code logically
- • Models real-world entities

Why Learn OOP?

- • Makes large code manageable
- • Promotes reusability
- • Prevents repetition
- • Common in software development

Class & Object: Concept

- Class → Blueprint/Template
- Object → Real instance created from the class
- Analogy: Class = Student Form, Object = Actual Student

Class Syntax

- class ClassName:
-     statements (attributes, methods)

- Example:
- class Student:
-     pass

Object Syntax

- object_name = ClassName()

- Example:
- s1 = Student()
- s2 = Student()

Attributes: Concept

- • Variables inside a class
- • Store object data

- Example: name, age, roll

Methods: Concept

- • Functions declared inside class
- • Define actions/behavior of object
- • Example: introduce(), study()

Syntax: Attributes & Methods

```
• class Student:
•     def __init__(self, name):
•         self.name = name  # attribute

•     def introduce(self):
•         print('Hello, I am', self.name)
```

__init__: Constructor

- • Special method
- • Runs automatically when object is created
- • Initializes attributes

Constructor Syntax

- class Student:
-     def __init__(self, name, age):
-         self.name = name
-         self.age = age

self Keyword

- • Refers to the current object
- • Used to access attributes & methods

- Analogy: self = 'I' in real life

Encapsulation: Concept

- • Protecting data inside a class
- • Controls who can access what
- • Achieved using _ and __ variables

Encapsulation Syntax

```
class Bank:
    def __init__(self):
        self.__balance = 0  # private

    def deposit(self, amt):
        self.__balance += amt
```

Abstraction: Concept

- • Hiding unnecessary internal details
- • Showing only important features
- • Example: Car's steering vs engine internals

Abstraction Syntax

```
• class Remote:
•     def turn_on(self):
•         print('TV ON')
```

Inheritance: Concept

- • One class (child) inherits from another (parent)
- • Child can use parent's attributes & methods

- Analogy: Child inherits traits from parents

Inheritance Syntax

- class Animal:
-     def sound(self): pass

- class Dog(Animal):
-     def bark(self):
-         print('Woof')

Polymorphism: Concept

- • Same function name, different behaviors
- • Achieved via method overriding

- Example: Dog.sound() vs Cat.sound()

Polymorphism Syntax

- class Animal:
-     def sound(self): print('Sound')

- class Cat(Animal):
-     def sound(self): print('Meow')

Summary of OOP Concepts

- • Class & Object
- • Attributes & Methods
- • Constructor (__init__)
- • self keyword
- • Encapsulation
- • Abstraction
- • Inheritance
- • Polymorphism