# Object-Oriented Programming (OOP) - Basic Concepts

Object-Oriented Programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of attributes, and code in the form of methods. It helps in structuring programs so that properties and behaviors are bundled into individual objects.

## The Four Main Pillars of OOP

**1. Encapsulation:** It is the process of wrapping data (variables) and methods (functions) together as a single unit. It helps to protect data from unauthorized access and modification.

Example in Python: class Person: def __init__(self, name): self.__name = name # private variable def get_name(self): return self.__name p = Person("Rajat") print(p.get_name())

**2. Inheritance:** It allows one class to acquire the properties and methods of another class. It promotes code reusability and establishes a relationship between classes.

Example in Python: class Parent: def show(self): print("This is Parent class") class Child(Parent): pass c = Child() c.show()

**3. Polymorphism:** It means 'many forms'. It allows methods to have the same name but behave differently based on the object calling them.

Example in Python: class Dog: def sound(self): print("Bark") class Cat: def sound(self): print("Meow") for animal in (Dog(), Cat()): animal.sound()

**4. Abstraction:** It means hiding complex implementation details and showing only the necessary features of an object.

Example in Python: from abc import ABC, abstractmethod class Shape(ABC): @abstractmethod def area(self): pass class Circle(Shape): def area(self): return "Area = $\pi r^2$" c = Circle() print(c.area())

Object-Oriented Programming simplifies software development and maintenance by providing concepts that model real-world entities effectively.