Object-Oriented Programming in Python

Presented by : Pandya Shaunak Rajnikant

What is OOP?

- A programming paradigm based on the concept of "objects"
- Objects contain both data and methods that operate on that data
- Helps structure complex programs into simple, reusable pieces
- Commonly used in many modern programming languages

Key Concepts in OOP

- **Class**: A blueprint for creating objects
- **Object**: An instance of a class
- **Encapsulation**: Bundling data with methods that operate on that data
- **Abstraction**: Hiding internal details and showing only essential features
- **Inheritance**: A way to form new classes using classes that have already been defined
- **Polymorphism**: The ability to present the same interface for different data types

Classes and Objects

- **Class**: A blueprint or template for creating objects
- **Object**: An instance of a class with its own data and behavior
- Objects interact with each other through methods
- Classes define the structure and behavior that the objects will have

Inheritance and Polymorphism

- **Inheritance** promotes code reusability by allowing a new class to take on the properties of an existing class
- **Polymorphism** allows objects of different classes to be treated as objects of a common superclass
- These features support scalability and flexibility in program design

Advantages of Using OOP

- Improves code organization and readability
- Encourages modular design for complex systems
- Makes maintenance and debugging easier
- Enhances code reusability through inheritance
- Supports abstraction and data protection

Real-World Applications of OOP

- Used in GUI-based applications, web frameworks, and games
- Common in design of enterprise software and databases
- Python's OOP is widely used in data science, automation, and AI projects
- Frameworks like Django and Flask follow OOP principles

Thank You