Day 12:

1. **Inheritance**:
   * **Inheritance** allows you to create class hierarchies, where a base class (also known as a parent class) provides its behavior and attributes to a derived class (also known as a child class).
   * Common properties and methods can be defined in the base class, and specific features can be added or modified in the derived classes.
   * For example, consider a Bird class with common properties like color and legs. [When creating a specific bird (e.g., a Pigeon), we can inherit the Bird class and add or override specific behaviors (e.g., customizing the fly() method for pigeons) 1](https://www.geeksforgeeks.org/understanding-encapsulation-inheritance-polymorphism-abstraction-in-oops/).
2. **Encapsulation**:
   * **Encapsulation** involves hiding the internal details of a class from other objects.
   * By enclosing properties (attributes) within methods (getters and setters), we control access to the data. This prevents direct modification of attributes and ensures proper initialization and retrieval.
   * [For instance, if a pigeon is born with a gray color, encapsulation ensures that the color remains unchanged unless explicitly modified](https://www.geeksforgeeks.org/understanding-encapsulation-inheritance-polymorphism-abstraction-in-oops/).
3. **Polymorphism**:
   * **Polymorphism** allows us to use a common operation (method) in different ways.
   * It enables flexibility by allowing objects of different classes to respond to the same method call.
   * [For example, both a Pigeon and an Eagle can have a fly() method, but their implementations may differ](https://www.geeksforgeeks.org/understanding-encapsulation-inheritance-polymorphism-abstraction-in-oops/)
4. **Abstract Base Classes (ABCs)**:
   * ABCs provide a way to define a common interface for a group of related classes.
   * They allow you to specify methods that must be implemented by any class inheriting from the ABC.
   * In Python, you can create an ABC using the abc module. [Abstract methods within an ABC serve as placeholders for methods that subclasses must override](https://www.geeksforgeeks.org/understanding-encapsulation-inheritance-polymorphism-abstraction-in-oops/),
5. **Interfaces**:
   * An **interface** defines a contract that classes must adhere to.
   * Unlike abstract classes, interfaces do not provide any implementation; they only declare method signatures.
   * [In Python, interfaces are not explicitly defined, but you can achieve similar behavior using ABCs with abstract methods 4](https://community.sap.com/t5/application-development-blog-posts/abap-oops-inheritance-encapsulation-polymorphism/ba-p/13574205).