1. OOP: Programming paradigm based on objects combining data and behavior.
2. Class: Blueprint for creating objects.
3. Object: Instance of a class.
4. Abstraction vs Encapsulation: Abstraction hides implementation; encapsulation hides data using access control.
5. Dunder methods: Special methods with __name__ for built-in behavior.
6. Inheritance: Child class acquires properties and methods of parent class.
7. Polymorphism: Same interface, different behavior.
8. Encapsulation in Python: Using private/protected attributes and methods.
9. Constructor: __init__ initializes object state.
10. Class vs Static methods: Class method works on class; static method is utility-like.
11. Method overloading: Achieved via default/variable arguments.
12. Method overriding: Child redefines parent method.
13. Property decorator: Controls attribute access via methods.
14. Importance of polymorphism: Flexibility and extensibility.

15.Abstract class: Class with abstract methods, cannot be instantiated.

16.Advantages of OOP: Reusability, modularity, maintainability.

17.Class vs Instance variable: Shared vs object-specific.

18.Multiple inheritance: Class inherits from multiple parents.

19.__str__ vs __repr__: User-friendly vs developer-friendly string.

20.super(): Calls parent class methods.

21.__del__: Destructor called on object deletion.

22.@staticmethod vs @classmethod: No class reference vs class reference.

23.Polymorphism with inheritance: Child overrides parent behavior.

24.Method chaining: Calling methods sequentially on same object.

25.__call__: Makes object callable like a function.