Theoretical Questions

1) OOP

Programming paradigm with objects bundling data+methods.

2) Class

Blueprint for creating objects.

3) Object

Instance of a class.

4) Abstraction vs Encapsulation

Abstraction hides *what*, encapsulation hides *how*.

5) Dunder methods

Special methods like **init**, **str**, **add**.

6) Inheritance

Subclass inherits methods/attrs from parent class.

7) Polymorphism

Same method name behaves differently in subclasses.

8) Encapsulation in Python

Use private vars (__var) and getters/setters.

9) Constructor

init initializes new object.

10) Class vs Static methods

Class methods take cls, static methods take neither.

11) Method overloading

Not native in Python; simulated via defaults/*args.

12) Method overriding

Subclass redefines parent's method.

13) Property decorator

@property lets methods act like attributes.

14) Importance of polymorphism

Increases flexibility, reuse, common interface.

15) Abstract class

Class with abstract methods via abc.ABC.

16) Advantages of OOP

Reusability, modularity, scalability.

17) Class vs Instance variable

Class vars shared, instance vars unique per object.

18) Multiple inheritance

Subclass inherits from multiple parents.

19) **str** vs **repr**

str: user-friendly, **repr**: unambiguous.

20) super()

Calls parent methods/constructors.

21) **del**

Destructor called on object deletion.

22) staticmethod vs classmethod

Staticmethod no cls/self, classmethod gets cls.

23) Polymorphism with inheritance

Subclass overrides; base ref can call overridden method.

24) Method chaining

Return self from methods for chained calls.

25) **call**

Lets object be invoked like a function.