OOPS ANSWERS

1. What is Object-Oriented Programming (OOP)?

OOP is a programming model based on objects, which encapsulate data and behavior. It emphasizes modularity, code reuse, and real-world modeling.

2. What is a class in OOP?

A class is a blueprint for creating objects. It defines attributes and methods that the objects (instances) will have.

3. What is an object in OOP?

An object is an instance of a class containing data and behavior defined by that class.

4. What is the difference between abstraction and encapsulation?

Abstraction hides implementation details; encapsulation restricts access to internal object state for protection.

5. What are dunder methods in Python?

Dunder (double underscore) methods like __init__, __str__, __repr__ are special methods used for operator overloading and customization.

6. Explain the concept of inheritance in OOP.

Inheritance allows one class (child) to acquire properties and behaviors of another class (parent), promoting code reuse.

7. What is polymorphism in OOP?

Polymorphism allows the same method to perform different behaviors based on the object calling it.

8. How is encapsulation achieved in Python?

Encapsulation is achieved using private attributes (prefix with __) and providing access via getter/setter methods.

9. What is a constructor in Python?

A constructor is a special method (__init__) automatically called when a new object is created.

10. What are class and static methods in Python?

@classmethod takes cls and accesses class-level data. @staticmethod doesn't take self or cls and behaves like a regular function inside a class.

11. What is method overloading in Python?

Python does not support traditional overloading. It can be mimicked using default or variable-length arguments.

12. What is method overriding in OOP?

Overriding means redefining a method of a parent class in its child class to provide specific behavior.

13. What is a property decorator in Python?

@property turns a method into a read-only attribute.

14. Why is polymorphism important in OOP?

It enables flexibility and the ability to use different object types through a common interface.

15. What is an abstract class in Python?

An abstract class has at least one abstract method and cannot be instantiated directly. It's defined using abc module.

16. What are the advantages of OOP?

Modularity, code reuse, encapsulation, abstraction, and easier maintenance.

17. What is the difference between a class variable and an instance variable?

Class variables are shared across all instances. Instance variables are specific to each object.

18. What is multiple inheritance in Python?

It means a class can inherit from more than one parent class.

19. Explain the purpose of __str__ and __repr__ methods in Python.

__str__ returns a user-friendly string; __repr__ returns a detailed developer-friendly string.

20. What is the significance of the super() function in Python?

super() allows access to methods of the superclass from the subclass.

21. What is the significance of the __del__ method in Python?

__del__ is a destructor method called when an object is deleted.

22. What is the difference between @staticmethod and @classmethod in Python?

@staticmethod doesn't access class or instance; @classmethod accesses the class via cls.

23. How does polymorphism work in Python with inheritance?

Through method overriding, the subclass can define a method with the same name as in the parent, allowing different behaviors.

24. What is method chaining in Python OOP?

Returning self in methods allows chaining multiple method calls in a single statement.

25. What is the purpose of the __call__ method in Python?

__call__ lets an object be called like a function.