1. What is Object-Oriented Programming (OOP)

OOP is a programming paradigm based on the concept of objects that contain both data and methods. It promotes reusability and modularity.

2. What is a class in OOP

A class is a blueprint for creating objects. It defines attributes (data) and methods (functions).

3. What is an object in OOP

An object is an instance of a class. It represents a specific entity with its own state and behavior.

4. What is the difference between abstraction and encapsulation

Abstraction hides implementation details, focusing on what an object does. Encapsulation hides data by restricting direct access and exposing methods.

5. What are dunder methods in Python

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

6. Explain the concept of inheritance in OOP

Inheritance allows a class to acquire attributes and methods from another class (parent). It promotes code reusability.

7. What is polymorphism in OOP

Polymorphism allows objects to take many forms. The same method can behave differently depending on the object.

8. How is encapsulation achieved in Python

Encapsulation is achieved using private/protected variables and getter/setter methods.

9. What is a constructor in Python

A constructor is the __init__ method in Python, automatically called when an object is created.

10. What are class and static methods in Python

Class methods use @classmethod and work with class variables. Static methods use @staticmethod and don’t access class or instance variables.

11. What is method overloading in Python

Python does not support traditional method overloading but can achieve similar effects using default arguments or *args.

12. What is method overriding in OOP

Method overriding occurs when a child class provides a specific implementation of a method already defined in the parent class.

13. What is a property decorator in Python

A property decorator (@property) allows defining methods that can be accessed like attributes.

14. Why is polymorphism important in OOP

Polymorphism is important because it allows flexibility and reusability of code.

15. What is an abstract class in Python

An abstract class is a class with abstract methods (declared but not implemented). It cannot be instantiated directly.

16. What are the advantages of OOP

Advantages of OOP: code reusability, modularity, scalability, maintainability.

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

A class variable is shared across all instances of a class, while an instance variable is unique to each object.

18. What is multiple inheritance in Python

Multiple inheritance means a class can inherit from more than one parent class.

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

__str__ provides a readable string representation for users, while __repr__ provides an unambiguous representation for developers.

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

super() is used to call methods from the parent class, commonly used in constructors.

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

__del__ is a destructor method, called when an object is about to be destroyed.

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

@staticmethod does not take self/cls; @classmethod takes cls as its parameter and can modify class variables.

23. How does polymorphism work in Python with inheritance

With inheritance, polymorphism allows overriding methods to have different behaviors depending on the child class.

24. What is method chaining in Python OOP

Method chaining means returning self from a method so multiple method calls can be linked together.

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

__call__ allows an object to be called like a function.