OOP viva Q/A

Q: What is Object-Oriented Programming (OOP)?

A: A programming paradigm based on objects and classes.

(Programming ka usool jo objects aur classes par mabni hota hai.)

Q: What is a class?

A: A blueprint for creating objects.

(Wo design jo objects bananay ke liye istemal hota hai.)

Q: What is an object?

A: An instance of a class.

(Class ka ek asal namoona jo kaam karta hai.)

Q: Define encapsulation.

A: Combining data and methods into one unit.

(Data aur functions ko ek jaga bandhna.)

Q: What is abstraction?

A: Hiding complexity and showing only essential features.

(Mushkil cheezein chhupa kar zaroori baatein dikhana.)

Q: Explain inheritance.

A: Acquiring properties and methods from a parent class.

(Parent class se properties aur methods lena.)

Q: What is polymorphism?

A: Ability of methods to behave differently based on objects.

(Methods ke mukhtalif objects par mukhtalif tor par kaam karna.)

Q: What is a constructor?

A: A special method to initialize objects.

(Object bananay wala khaas method.)

Q: What is method overriding?

A: Child class providing a new implementation of a parent method.

(Child class mein parent method ko dobara define karna.)

Q: What does the public access modifier do?

A: Allows access from anywhere.

(Kisi bhi jagah se access milta hai.)

Q: What does the private access modifier do?

A: Restricts access to within the class.

(Sirf class ke andar access hota hai.)

Q: Describe the protected access modifier.

A: Accessible within class and subclasses.

(Class aur uske derived classes ke liye accessible.)

Q: What is an interface?

A: A collection of method declarations without implementations.

(Methods ka ek group jisme implementation nahi hoti.)

Q: Define an abstract class.

A: A class that cannot be instantiated.

(Aisi class jiska object nahi banaya ja sakta.)

Q: What is multiple inheritance?

A: A class inheriting from more than one class.

(Ek class kai classes se properties lena.)

Q: Why use OOP?

A: For modularity, reusability, and maintainability.

(Code ko tod kar chhote hisso mein rakhne aur reuse karne ke liye.)

Q: What is modularity?

A: Dividing code into logical, independent parts.

(Code ko aise hisso mein baantna jo alag alag chal saken.)

Q: How does reusability benefit programming?

A: Write code once, use it multiple times.

(Code ek dafa likho aur bar bar istemal karo.)

Q: What does scalability mean?

A: The ease to extend and grow a program.

(Asani se program ko barhane aur badhane ki salahiyat.)

Q: What is real-world modeling in OOP?

A: Representing real objects and their relationships.

(Haqeeqat ke objects ko aur unke talluqaat ko dikhana.)

Q: What is method overloading?

A: Same method name with different parameters.

(Ek hi method naam lekin mukhtalif parameters.)

Q: What is compile-time polymorphism?

A: Polymorphism resolved during compilation (overloading).

(Jab compiler faisla karta hai kaunsa method call hoga.)

Q: What is runtime polymorphism?

A: Polymorphism resolved during program execution (overriding).

(Program chal raha hota hai tab method select karna.)

Q: Why hide data using encapsulation?

A: To protect data from unauthorized access.

(Data ko beja access se bachane ke liye.)

Q: What is the ‘super’ keyword used for?

A: To call parent class constructor or methods.

(Parent class ke constructor ya methods ko call karne ke liye.)

Q: What is a virtual function?

A: A function in base class that can be overridden.

(Base class ka function jo child class badal sakti hai.)

Q: Can abstract classes have method implementations?

A: Yes, some methods can be implemented and some abstract.

(Kuch methods define kar sakte hain, kuch nahi.)

Q: What is a default constructor?

A: Constructor automatically provided if none is defined.

(Aisa constructor jo compiler khud banata hai agar programmer na banaye.)

Q: Can we instantiate an interface?

A: No, interfaces cannot create objects directly.

(Interface se seedha object nahi banaya ja sakta.)

Q: What is a method signature?

A: The method's name and parameter list.

(Method ka naam aur uske arguments.)

CLASS:

Q: What is a class?
A: A blueprint or template for creating objects.
(Aik naksha ya template jo objects banane ke liye hota hai.)

Q: What does a class define?
A: It defines attributes (data) and methods (functions) for objects.
(Yeh attributes (data) aur methods (functions) ko define karta hai jo objects mein hote hain.)

Q: Can you give an example of a class?
A: A Vehicle class with attributes like color, speed, and model.
(Ek Vehicle class jo color, speed, aur model jaise attributes rakhti hai.)

Q: What is an object?
A: An instance of a class with unique data.
(Class ka ek instance jo apna unique data rakhta hai.)

Q: How do you create a class in Python?
A: Use the class keyword followed by the class name.
(Python mein class keyword istemal karke class banate hain.)

Q: What is the syntax to define a class?
A: class ClassName: pass
(Class define karne ka syntax: class ClassName: pass)

Q: What are attributes?
A: Variables that belong to an object, representing its state.
(Variables jo object se taluq rakhte hain, jo uski state dikhate hain.)

Q: What are methods?
A: Functions that belong to an object, defining its behavior.
(Functions jo object se taluq rakhte hain, jo uska behavior define karte hain.)

Q: How do you instantiate an object?
A: Call the class name like a function.
(Class ka naam function ki tarah call karke object banate hain.)

Q: What is instantiation?
A: The process of creating an object from a class.
(Class se object banane ka amal.)

Q: What does self refer to in a class?
A: It refers to the current instance of the class.
(Yeh class ka maujooda instance dikhata hai.)

Q: Why is self important?
A: It is used to access attributes and methods within the class.
(Yeh attributes aur methods tak access dene ke liye zaroori hai.)

Q: What is the purpose of the __init__ method?
A: It initializes the object when created.
(Yeh object banate waqt usay initialize karta hai.)

Q: Can you show an example of a class in Python?
A:

python4 lines

class Person:

def __init__(self, name):

self.name = name

self.state = "Idle"

(Yeh ek example hai Python mein class ka.)

Q: What does the running method do in the Person class?
A: It changes the state to "Running" and prints a message.
(Yeh state ko "Running" mein badalta hai aur message print karta hai.)

Q: How do you create an object of the Person class?
A: person1 = Person("Ali")
(Person class ka object banane ka tareeqa: person1 = Person("Ali"))

Q: What does the show_state method do?
A: It prints the current state of the person.
(Yeh insaan ki maujooda state print karta hai.)

Q: What will person1.show_state() output if person1 is initialized with "Ali"?
A: "Ali is currently in 'Idle' state."
(Agar person1 ko "Ali" se initialize kiya gaya ho to output hoga: "Ali is currently in 'Idle' state.")

Q: What happens when person1.walking() is called?
A: It changes the state to "Walking" and prints a message.
(Yeh state ko "Walking" mein badalta hai aur message print karta hai.)

Q: What will person1.sleeping() do?
A: It changes the state to "Sleeping"

03 constructor and destructor

Q: What is a constructor?
A: A special method called when an object of a class is created.
(Aik khaas method jo tab call hota hai jab class ka object banaya jata hai.)

Q: What is the purpose of a constructor?
A: To initialize the object's attributes or perform setup tasks.
(Object ke attributes ko initialize karna ya setup kaam karna.)

Q: What is the name of the constructor method in Python?
A: The constructor method is named __init__.
(Python mein constructor method ka naam __init__ hota hai.)

Q: What is a parameterized constructor?
A: A constructor that takes parameters to initialize attributes.
(Aisi constructor jo parameters leti hai attributes ko initialize karne ke liye.)

Q: Can you show an example of a parameterized constructor?
A:

class Person:

def __init__(self, name, age, address):

self.name = name

self.age = age

self.address = address

(Yeh ek example hai parameterized constructor ka.)

Q: How do you create an object using a parameterized constructor?
A: person1 = Person("Alice", 30, "81 Harvard Avenue New York, NY 1012")
(Parameterized constructor se object banane ka tareeqa: person1 = Person("Alice", 30, "81 Harvard Avenue New York, NY 1012"))

Q: What does the __del__ method do?
A: It is called when an object is about to be destroyed.
(Yeh tab call hota hai jab object khatam hone wala hota hai.)

Q: What is the purpose of a destructor?
A: To perform cleanup tasks, like releasing resources.
(Cleanup kaam karna, jaise resources ko release karna.)

Q: What is the name of the destructor method in Python?
A: The destructor method is named __del__.
(Python mein destructor method ka naam __del__ hota hai.)

Q: Can you show an example of a destructor?
A:

class Car:

def __init__(self, brand, model):

self.brand = brand

self.model = model

def __del__(self):

print(f"The {self.brand} {self.model} has been destroyed.")

(Yeh ek example hai destructor ka.)

Q: What happens when you delete an object explicitly?
A: It triggers the destructor method.
(Jab aap object ko explicitly delete karte hain, yeh destructor method ko trigger karta hai.)

Q: What will be the output when the destructor is called?
A: It prints a message indicating the object has been destroyed.
(Yeh ek message print karta hai jo batata hai ke object khatam ho gaya hai.)

Q: How do you explicitly delete an object in Python?
A: Use the del statement followed by the object name.
(Object ko explicitly delete karne ke liye del statement istemal karte hain.)

Q: Can you give an example of deleting an object?
A:

```
my_car = Car("Toyota", "Corolla")

del my_car  # Output: The Toyota Corolla has been destroyed.
```

Q: Why is it important to understand constructors and destructors?
A: They help manage the lifecycle of objects in programs.
(Yeh programs mein objects ki lifecycle ko manage karne mein madadgar hote hain.)

Access Modifiers: Public, Private, and Protected

1. **Q: What are access modifiers in Python?**
   A: They control the visibility of class attributes and methods.
   (Yeh class ke attributes aur methods ki visibility ko control karte hain.)

2. **Q: What is a public access modifier?**
   A: Attributes and methods accessible from anywhere.
   (Attributes aur methods jo kisi bhi jagah se access kiye ja sakte hain.)

3. **Q: What is a protected access modifier?**
   A: Attributes and methods intended for internal use, prefixed with a single underscore (_).
   (Attributes aur methods jo internal istemal ke liye hain, ek single underscore se shuru hote hain.)

4. **Q: What is a private access modifier?**
   A: Attributes and methods accessible only within the class, prefixed with double underscores (__).
   (Attributes aur methods jo sirf class ke andar access kiye ja sakte hain, double underscores se shuru hote hain.)

5. **Q: Can you give an example of a protected attribute?**
   A: **_secret_location** in the **Father** class.
   (Father class mein **_secret_location** ek protected attribute hai.)

6. **Q: What happens if a protected attribute is accessed from outside the class?**
   A: It is not recommended but still accessible.
   (Yeh recommend nahi kiya jata lekin phir bhi access kiya ja sakta hai.)

7. **Q: Can you give an example of a private attribute?**
   A: **__salary** in the **Employee** class.
   (Employee class mein **__salary** ek private attribute hai.)

8. **Q: How can you access a private attribute using name mangling?**
   A: By using **_ClassName__attribute**.
   (Name mangling ka istemal karke **_ClassName__attribute** se access kar sakte hain.)

9. **Q: What is single inheritance?**
   A: A subclass inherits from a single superclass.
   (Aik subclass ek hi superclass se inherit karta hai.)

10. **Q: What is multiple inheritance?**
    A: A subclass inherits from multiple superclasses.
    (Aik subclass kai superclasses se inherit karta hai.)

11. **Q: What is the purpose of the super() function?**
    A: To call methods from the parent class.
    (Parent class ke methods ko call karne ke liye.)

12. **Q: What is method overriding?**
    A: When a subclass provides a specific implementation of a method defined in its superclass.
    (Jab ek subclass apne superclass mein defined method ki specific implementation deta hai.)

13. **Q: Can you give an example of method overriding?**
    A: The **speak** method in the **Dog** class overrides the **speak** method in the **Animal** class.
    (Dog class ka **speak** method Animal class ke **speak** method ko override karta hai.)

14. **Q: What is polymorphism?**
    A: The ability to use a single interface for different data types.
    (Mukhtalif data types ke liye ek hi interface ka istemal karne ki salahiyat.)

15. **Q: Can you give an example of polymorphism?**
    A: Using an **Animal** reference to refer to a **Dog** object.
    (Animal reference ka istemal karke Dog object ko refer karna.)

16. **Q: What is operator overloading?**
    A: Customizing the behavior of operators for user-defined classes.
    (User -defined classes ke liye operators ka behavior customize karna.)

17. **Q: Can you give an example of operator overloading?**
    A: Overloading the + operator in the **Point** class.
    (Point class mein + operator ko overload karna.)

18. **Q: What is duck typing?**
    A: A concept where the type of an object is determined by its behavior rather than its class.
    (Aisi soch jahan object ka type uske behavior se tay kiya jata hai na ke uski class se.)

19. **Q: Can you give an example of duck typing?**
    A: The **make_it_quack** function works with both **Duck** and **Person** classes.
    (make_it_quack function Duck aur Person dono classes ke sath kaam karta hai.)

20. **Q: What is abstraction?**
    A: Abstraction simplifies complex systems by breaking them into manageable parts.
    (Abstraction mushkil systems ko asan hisson mein baant kar simplify karta hai.)

21. **Q: What are abstract classes?**
    A: Incomplete templates that cannot be instantiated directly.
    (Aisi templates jo incomplete hoti hain aur seedha istemal nahi ki ja sakti.)

22. **Q: What is the purpose of abstract methods?**
    A: To define required methods for child classes and ensure consistency.
    (Child classes ke liye zaroori methods define karna aur consistency ensure karna.)

23. **Q: How do you create an abstract class in Python?**
    A: Use the **abc** module and decorate methods with **@abstractmethod**.
    (abc module ka istemal karke aur methods ko **@abstractmethod** se decorate karke.)

24. **Q: Can you instantiate an abstract class?**
    A: No, you cannot create objects from an abstract class.
    (Nahi, aap abstract class se objects nahi bana sakte.)

25. **Q: What happens if you try to instantiate a class with an abstract method?**
    A: It raises a TypeError.
    (Yeh TypeError raise karta hai.)

26. **Q: What is a class variable?**
    A: A variable shared by all instances of a class.
    (Aisi variable jo class ke tamam instances ke darmiyan share hoti hai.)

27. **Q: Can you give an example of a class variable?**
    A: **name** in the **City** class.
    (City class mein **name** ek class variable hai.)

28. **Q: What are instance methods?**
    A: Methods that operate on an instance of the class and can access instance attributes.
    (Methods jo class ke instance par kaam karte hain aur instance attributes tak access karte hain.)

29. **Q: What is the first parameter of an instance method?**
    A: The first parameter is always **self**.
    (Pehla parameter hamesha **self** hota hai.)

30. **Q: What are class methods?**
    A: Methods that operate on the class itself and can access class attributes.
    (Methods jo class par kaam karte hain aur class attributes tak access karte hain.)

31. **Q: What is the first parameter of a class method?**
    A: The first parameter is always **cls**.
    (Pehla parameter hamesha **cls** hota hai.)

32. **Q: What decorator is used for class methods?**
    A: The **@classmethod** decorator.
    (Class methods ke liye **@classmethod** decorator istemal hota hai.)

33. **Q: What are static methods?**
    A: Methods that do not depend on class or instance data.
    (Methods jo class ya instance data par depend nahi karte.)

34. **Q: What decorator is used for static methods?**
    A: The **@staticmethod** decorator.
    (Static methods ke liye **@staticmethod** decorator istemal hota hai.)

35. **Q: Can static methods access instance or class attributes?**
    A: No, they cannot access instance or class attributes.
    (Nahi, yeh instance ya class attributes tak access nahi kar sakte.)

36. **Q: What is the difference between cls and class name in class methods?**
    A: **cls** is dynamic and supports inheritance, while class name is static.
    (cls dynamic hai aur inheritance ko support karta hai, jabke class name static hai.)
37. **Q: Can you give an example of a class method?**
    A:
```
class Human:
    specie = "Homo sapiens"

    @classmethod
    def get_specie(cls):
        return cls.specie
```
    (Yeh ek example hai class method ka.)
45. **Q: What is the output of Human.get_specie()?**
    A: '**Homo sapiens**'
    (Output hoga: '**Homo sapiens**')

Q: Why use cls() in class methods?
A: It allows creating instances of the class, supporting inheritance.
(Yeh class ke instances banane ki ijazat deta hai, inheritance ko support

Questions and Answers

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

   A: A programming paradigm that organizes software design around objects and classes.

   (Aik programming ka andaaz jo software design ko objects aur classes ke ird gird ghoomta hai.)

2. **Q: What is a class?**

   A: A blueprint or template for creating objects.

   (Aik naksha ya template jo objects banane ke liye hota hai.)

3. **Q: What is an object?**

   A: An instance of a class with its own unique data.

   (Class ka ek instance jo apna unique data rakhta hai.)

4. **Q: Why use OOP?**

   A: For modularity, reusability, maintainability, scalability, and real-world modeling.

   (Modularity, reusability, maintainability, scalability, aur haqeeqi duniya ka modeling karne ke liye.)

5. **Q: What is encapsulation?**

   A: Bundling data and methods into a single unit (class) and restricting access.

   (Data aur methods ko ek jaga bandhna aur access ko restrict karna.)

6. **Q: What is abstraction?**

   A: Hiding complex implementation details and exposing only necessary features.

   (Mushkil implementation details ko chhupa kar sirf zaroori features dikhana.)

7. **Q: What is inheritance?**

   A: A mechanism where a child class inherits attributes and methods from a parent class.

   (Aik tareeqa jahan child class parent class se attributes aur methods inherit karta hai.)

8. **Q: What is polymorphism?**

   A: The ability to treat objects of different classes as objects of a common superclass.

   (Mukhtalif classes ke objects ko ek common superclass ke objects ki tarah treat karne ki salahiyat.)

9. **Q: What is the purpose of the `__init__` method?**

   A: It initializes the object when it is created.

   (Yeh object banate waqt usay initialize karta hai.)

10. **Q: What does the `self` keyword refer to?**

A: It refers to the current instance of the class.

(Yeh class ka maujooda instance dikhata hai.)

11. **Q: How do you create a class in Python?**

A: Use the `class` keyword followed by the class name.

(Python mein `class` keyword istemal karke class banate hain.)

12. **Q: What are attributes in a class?**

A: Variables that belong to an object, representing its state or properties.

(Variables jo object se taluq rakhte hain, jo uski state ya properties dikhate hain.)

13. **Q: What are methods in a class?**

A: Functions that belong to an object, defining its behavior.

(Functions jo object se taluq rakhte hain, jo uska behavior define karte hain.)

14. **Q: How do you instantiate an object?**

A: Call the class name like a function.

(Class ka naam function ki tarah call karke object banate hain.)

15. **Q: What is a class variable?**

A: A variable shared by all instances of a class.

(Aisi variable jo class ke tamam instances ke darmiyan share hoti hai.)

16. **Q: What is the difference between public, protected, and private attributes?**

A: Public attributes are accessible from anywhere, protected attributes are for internal use, and private attributes are accessible only within the class.

(Public attributes har jagah se access kiye ja sakte hain, protected attributes internal istemal ke liye hain, aur private attributes sirf class ke andar access kiye ja sakte hain.)

17. **Q: What is the significance of the `@abstractmethod` decorator?**

A: It marks a method as abstract, requiring child classes to implement it.

(Yeh method ko abstract mark karta hai, jo child classes ko implement karna zaroori hota hai.)

18. **Q: Can you instantiate an abstract class?**

A: No, you cannot create objects from an abstract class.

(Nahi, aap abstract class se objects nahi bana sakte.)

19. **Q: What is method overriding?**

A: When a subclass provides a specific implementation

Questions and Answers

1. **Q: What is a class in Python?**
   A: A blueprint or template for creating objects.
   (Python mein class ek naksha ya template hai jo objects banane ke liye istemal hota hai.)

2. **Q: What is an object?**
   A: An instance of a class with its own unique data.
   (Object class ka ek instance hai jo apna unique data rakhta hai.)

3. **Q: How do you define a class in Python?**
   A: Use the **class** keyword followed by the class name.
   (Python mein class define karne ke liye **class** keyword istemal karte hain.)

4. **Q: What is the purpose of the __init__ method?**
   A: It initializes the object's attributes when an object is created.
   (Yeh object banate waqt uske attributes ko initialize karta hai.)

5. **Q: What is a parameterized constructor?**
   A: A constructor that takes parameters to initialize attributes.
   (Aisi constructor jo parameters leti hai attributes ko initialize karne ke liye.)

6. **Q: Can you give an example of a parameterized constructor?**
   A:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

1. (Yeh ek example hai parameterized constructor ka.)

2. **Q: What is a default constructor?**
   A: A constructor that does not take any parameters and is provided by Python if none is defined.
   (Aisi constructor jo koi parameters nahi leti aur agar koi define nahi ki gayi ho to Python khud provide karta hai.)

3. **Q: What is a destructor?**
   A: A special method called when an object is about to be destroyed.
   (Aik khaas method jo tab call hota hai jab object khatam hone wala hota hai.)

4. **Q: What is the name of the destructor method in Python?**
   A: The destructor method is named **__del__**.
   (Python mein destructor method ka naam **__del__** hota hai.)

5. **Q: Can you give an example of a destructor?**
   A:

```
class Car:
    def __del__(self):
        print("Car has been destroyed.")
```

1. **Q: What happens when you delete an object using the del keyword?**
   A: It triggers the destructor method.
   (Yeh destructor method ko trigger karta hai.)

2. **Q: What is the purpose of the self keyword?**
   A: It refers to the current instance of the class.
   (Yeh class ka maujooda instance dikhata hai.)

3. **Q: How do you create an object of a class?**
   A: By calling the class name like a function.
   (Class ka naam function ki tarah call karke object banate hain.)

4. **Q: What are attributes in a class?**
   A: Variables that belong to an object, representing its state or properties.
   (Variables jo object se taluq rakhte hain, jo uski state ya properties dikhate hain.)

5. **Q: What are methods in a class?**
   A: Functions that belong to an object, defining its behavior.
   (Functions jo object se taluq rakhte hain, jo uska behavior define karte hain.)

6. **Q: How do you access an object's attributes?**
   A: Using dot notation (e.g., **object.attribute**).
   (Object ke attributes ko dot notation se access karte hain, jaise **object.attribute**.)

7. **Q: What is the output of calling a method on an object?**
   A: It executes the method's code and may return a value or print output.
   (Yeh method ka code execute karta hai aur shayad value return kare ya output print kare.)

8. **Q: What is the significance of using constructors and destructors?**
   A: They help manage the lifecycle of objects, initializing and cleaning up resources.
   (Yeh objects ki lifecycle ko manage karne mein madadgar hote hain, resources ko initialize aur clean up karte hain.)

Q: Can you create multiple objects from the same class?
A: Yes, you can create multiple instances of a class. Each instance can have its own unique attribute values while sharing the same methods defined in the class. For example, if you have a **Vehicle** class, you can create multiple objects like **car1**, **car2**, and **car3**, each representing different vehicles with their own attributes such as color and speed. # Define the Vehicle class

```
class Vehicle:

    def __init__(self, color, speed):

        self.color = color  # Attribute

        self.speed = speed  # Attribute


    def display(self):

        print(f"Vehicle color: {self.color}, Current speed: {self.speed} km/h")


# Create multiple objects (instances) of the Vehicle class

car1 = Vehicle("Red", 60)

car2 = Vehicle("Blue", 80)

car3 = Vehicle("Green", 100)


# Access attributes and call methods for each object

car1.display()  # Output: Vehicle color: Red, Current speed: 60 km/h
```

```
car2.display()  # Output: Vehicle color: Blue, Current speed: 80 km/h

car3.display()  # Output: Vehicle color: Green, Current speed: 100 km/h
```

✅ Class and Instance Attributes

1. **What is a class attribute?**
   Attribute shared among all instances.
   (Yeh woh attribute hota hai jo sab objects mein same hota hai.)

2. **What is an instance attribute?**
   Attribute unique to each instance.
   (Yeh har object ka alag hota hai.)

3. **Where is a class attribute defined?**
   Inside the class, outside methods.
   (Class ke andar likha jata hai lekin kisi function ke bahar.)

4. **Where is an instance attribute defined?**
   Inside the __init__ method.
   (Yeh constructor mein define hota hai.)

5. **Are class attributes stored once or multiple times?**
   Once.
   (Sirf aik dafa memory mein store hota hai.)

6. **How do you access a class attribute?**
   Using class or instance.
   (Class ka naam ya object dono se access kar saktay hain.)

7. **How do you access an instance attribute?**
   Using the instance only.
   (Sirf object se hi access kar saktay hain.)

8. **What happens if you modify a class attribute through an instance?**
   It creates a new instance attribute.
   (Naya instance attribute ban jata hai, class wala shadow ho jata hai.)

9. **Which attribute shows all attributes of an instance or class?**
   __dict__
   (Yeh dictionary hoti hai jo sab attributes dikhati hai.)

10. **Write a line to print class attributes of class Dog.**
    print(Dog.__dict__)
    (Yeh Dog class ke sab attributes dikha dega.)

✅ Code-Based Questions (Attributes)

11. **What will this code print?**

```
class A:
    val = 10

a = A()

a.val = 20

print(A.val)
```

Answer: 10
(Class ka attribute change nahi hua, sirf instance ka hua.)

12. **What will this print?**

```
class A:
    x = 5
a1 = A()
a2 = A()
a1.x = 10
print(a2.x)
```

Answer: 5
(a2 par koi shadowing nahi hui, isliye original class value print hogi.)

✅ **Instance, Class, and Static Methods**

13. **What is an instance method?**
    Method that works on instance data.
    (Yeh method object ke data par kaam karta hai.)

14. **What is the first parameter of an instance method?**
    self
    (Yeh object ko refer karta hai.)

15. **What is a class method?**
    Method that works on class data.
    (Yeh method class ke data ko modify karta hai.)

16. **Which decorator is used for a class method?**
    @classmethod
    (Yeh class method banata hai.)

17. **What is the first parameter of a class method?**
    cls
    (Yeh class ko refer karta hai.)

18. **What is a static method?**
    Method not dependent on class or instance.
    (Yeh independent utility function hota hai.)

19. **Which decorator is used for a static method?**
    @staticmethod
    (Yeh static method define karta hai.)

20. **Can a static method access class attributes?**
    No
    (Yeh class ya instance ka data access nahi kar sakta.)

✅ **Code-Based Questions (Methods)**

21. **What is output?**

```
class Person:
    species = "Homo sapiens"

    @classmethod
```

```
    def update_species(cls, new_species):
        cls.species = new_species


Person.update_species("Homo neanderthalensis")
print(Person.species)
```

Answer: Homo neanderthalensis
(Class method ne species ko update kar diya.)

22. **What is output?**

```
python
CopyEdit
class X:
    @staticmethod
    def add(a, b):
        return a + b


print(X.add(3, 4))
```

Answer: 7
(Static method directly kaam karta hai, bina object ke.)

23. **What is output?**

```
python
CopyEdit
class Y:
    def __init__(self, val):
        self.val = val


    def show(self):
        print(self.val)


obj = Y(10)
obj.show()
```

Answer: 10
(Instance method ne instance attribute print kiya.)

✅ **Quick Recap Viva**

24. **Keyword for instance reference?**
    self
    (Yeh current object ko refer karta hai.)

25. **Keyword for class reference?**
    cls
    (Yeh current class ko refer karta hai.)

26. **Which method doesn't use self or cls?**
    Static method
    (Yeh independent hoti hai.)

27. **What is the difference between instance and class method in one line?**
    Instance methods use object data, class methods use class data.
    (Object aur class ke data ka fark hota hai.)

🔐 Encapsulation

Q1: What is encapsulation?
A1: Encapsulation is the process of bundling data and methods within a class and restricting direct access to some of the object's components.

➡ Roman Urdu:
Encapsulation ka matlab hai ke data aur us se related functions ko ek class mein chhupa lena, taake koi direct access na kar sake.

Q2: What are access modifiers in Python?
A2:

- Public: No prefix (e.g., name)
- Protected: Single underscore prefix (e.g., _age)
- Private: Double underscore prefix (e.g., __salary)

➡ Roman Urdu:
Public ka matlab hai sab access kar sakte hain. Protected ka matlab hai sirf class aur uski child classes access kar sakti hain. Private ka matlab hai ke sirf class ke andar hi access ho sakta hai.

Q3: How do you access private variables?
A3: By using getter and setter methods.

➡ Roman Urdu:
Private variables ko direct nahi, balke getter aur setter methods ke zariye access karte hain.

🚀 Inheritance

Q4: What is inheritance?
A4: Inheritance allows a class (child) to inherit properties and methods from another class (parent).

➡ Roman Urdu:
Inheritance ka matlab hai ke aik class doosri class ke functions aur properties ko use kar sakti hai.

Q5: What are types of inheritance in Python?
A5:

- Single Inheritance
- Multiple Inheritance

➡ Roman Urdu:
Single mein aik parent hota hai. Multiple mein ek se zyada parents hote hain.

Q6: What is the use of super()?
A6: It is used to call the parent class's constructor or methods.

➞ **Roman Urdu:**
super() se parent class ka constructor ya method call kiya jata hai.

Q7: What is method overriding?
A7: When a child class redefines a method from its parent class.

➞ **Roman Urdu:**
Jab child class parent ka method dobara likhti hai apne tareeqe se.

🔄 **Polymorphism**

Q8: What is polymorphism?
A8: Polymorphism allows the same method to behave differently in different classes.

➞ **Roman Urdu:**
Polymorphism ka matlab hai aik hi method alag classes mein alag tareeqe se kaam kare.

Q9: What are types of polymorphism in Python?
A9:

- Method Overriding
- Operator Overloading
- Duck Typing

➞ **Roman Urdu:**
Method overriding mein child method parent wale ko overwrite karta hai. Operator overloading mein symbols ka meaning change hota hai. Duck typing mein type nahi, behavior dekha jata hai.

Q10: What is duck typing?
A10: If an object behaves like a type, it is treated as that type, regardless of its actual class.

➞ **Roman Urdu:**
Agar koi object kisi cheez ki tarah behave kare to usay woh hi samjha jata hai—chahe woh us type ka ho ya nahi.

🧩 **Abstraction**

Q11: What is abstraction?
A11: Abstraction means hiding the internal details and showing only essential features.

➞ **Roman Urdu:**
Abstraction mein sirf zaroori cheezein dikhate hain, baki details chhupa lete hain.

Q12: How is an abstract class defined in Python?
A12: Using ABC module and @abstractmethod decorator.

➡ **Roman Urdu:**
Abstract class banane ke liye ABC module aur @abstractmethod ka use karte hain.

Q13: Why can't we create objects of abstract classes?
A13: Because they have incomplete methods that must be implemented in a subclass.

➡ **Roman Urdu:**
Kyunkay abstract class mein kuch methods adhoore hote hain—unko pehle complete karna padta hai.

📦 **object Class**

Q14: What is the object class in Python?
A14: It is the base class for all Python classes.

➡ **Roman Urdu:**
Har class Python mein object class se hi banti hai—ye base class hoti hai.

Q15: Name some common methods from the object class.
A15: __init__(), __str__(), __eq__(), __repr__() etc.

➡ **Roman Urdu:**
Ye built-in methods hote hain jo har class mein automatically hote hain, jaise constructor, print ka method, etc.

✅ **Class vs Static vs Instance Variables in Python (Viva Format)**

1. What are class variables?

Class variables are variables that are shared among all instances of a class. They are defined inside the class but outside any method.
Roman Urdu: Class variables wo variables hain jo tamam objects mein common hote hain. Ye class ke andar declare hote hain, lekin kisi function ke andar nahi.

2. What are instance variables?

Instance variables are specific to each object of a class. They are defined using self inside the constructor or methods.
Roman Urdu: Instance variables har object ke liye alag hote hain. Inhein self ke sath banaya jata hai.

3. Are class variables and static variables the same in Python?

In Python, class variables are sometimes called static variables, but usually we just use the term **class variable**.
Roman Urdu: Python mein static aur class variable ko aksar interchangeably use kiya jata hai, lekin dono ka matlab class-level data hota hai.

4. How do you access class variables?

You can access class variables using the class name or an instance.
Roman Urdu: Class variable ko class ka naam ya object dono se access kiya ja sakta hai.

Example:

python

CopyEdit

print(Bakery.type)

print(cake1.type)

5. How do you modify class variables?

To properly modify a class variable, you should use the class name. If you use an instance, it creates a new instance variable instead.
Roman Urdu: Agar aap object ke zariye modify karein to wo class variable nahi, balkay naya instance variable ban jata hai.

Example:

python

CopyEdit

Bakery.type = "pastry" # Modifies class variable

cake1.type = "cookie" # Creates instance variable

6. What happens when you modify a class variable through an instance?

It creates a new instance variable for that object, and it no longer shares the class variable.
Roman Urdu: Agar object se modify karo to wo apna naya variable bana leta hai, aur class wale se alag ho jata hai.

7. What is the difference between class and instance variables?

Feature	Class Variable	Instance Variable
Scope	Shared by all instances	Unique to each instance
Creation	Defined when class is created	Defined when object is created
Access	Accessed via class or object	Accessed via object
Modification	Modified via class name only	Modified via object only
Use Case	Store shared data like counts/stats	Store data unique to each object

Roman Urdu: Class variable sab ke liye common hota hai, instance variable har object ke liye alag hota hai.

✅ **Example Code for Better Understanding**

python

CopyEdit

```
class Bakery:
    type = "cake"  # Class variable

    def __init__(self, flavor, price):
        self.flavor = flavor  # Instance variable
        self.price = price    # Instance variable

cake1 = Bakery("Chocolate", 25.00)
cake2 = Bakery("Vanilla", 22.00)

print(Bakery.type)      # Output: cake
print(cake1.flavor)     # Output: Chocolate
print(cake2.price)      # Output: 22.0
```

```
Bakery.type = "pastry"  # Modifying class variable
print(cake1.type)       # Output: pastry


cake1.type = "cookie"   # Creates new instance variable
print(cake1.type)       # Output: cookie
print(cake2.type)       # Output: pastry
```

✅ **12. Composition vs Aggregation in OOP (Viva Format)**

---

1. What is Composition in OOP?

Composition is a "has-a" relationship where one class contains another class, and the contained class's object cannot exist without the container class.

Roman Urdu: Composition ek strong relationship hoti hai. Agar main object destroy ho jaye to andar wala object bhi destroy ho jata hai. Jaise car aur uska engine.

---

2. What is Aggregation in OOP?

Aggregation is also a "has-a" relationship, but it's weaker. The contained object can exist even if the container is destroyed.

Roman Urdu: Aggregation mein object aik dosray se loosely connected hote hain. Agar main object destroy ho jaye to contained object zinda reh sakta hai. Jaise university aur departments.

---

3. Difference Between Composition and Aggregation

Feature	Composition	Aggregation
Relationship	Strong "has-a"	Weak "has-a"
Object Dependency	Dependent on container	Independent of container
Example	Car has-a Engine	University has-a Department
Lifespan Link	Ends with container	Continues after container is gone

Roman Urdu: Composition mein object ki zindagi container pe depend karti hai. Aggregation mein nahi.

---

4. Difference Between Composition and Inheritance

Feature	Composition ("has-a")	Inheritance ("is-a")
Type	Object relationship	Class hierarchy
Coupling	Loosely coupled	Tightly coupled
Flexibility	More flexible	Less flexible
Reusability	Through contained objects	Through inherited classes

Roman Urdu: Composition zyada flexible hoti hai aur components ko easily replace kiya ja sakta hai. Inheritance mein tightly linked hota hai.

5. Code Example of Composition and Aggregation

python

CopyEdit

```
# Composition
class Engine:
    def start(self):
        return "Engine starting"

class Car:
    def __init__(self):
        self.engine = Engine()  # Strongly owned by Car

    def start(self):
        return f"Car starting: {self.engine.start()}"

# Aggregation
class Department:
    def __init__(self, name):
        self.name = name

class University:
    def __init__(self, name):
        self.name = name
        self.departments = []

    def add_department(self, department):
        self.departments.append(department)
```

Roman Urdu: Composition mein Car apna engine khud banata hai. Aggregation mein University departments ko accept karti hai lekin unhe khud nahi banati.

✅ **13. Method Resolution Order (MRO) in Python (Viva Format)**

---

1. What is Method Resolution Order (MRO) in Python?

MRO is the order in which Python searches for methods or attributes in a class hierarchy, especially when multiple inheritance is involved.

Roman Urdu: MRO batata hai ke Python kis order mein classes ko check karega jab kisi method ko call kiya jaye.

---

2. How is MRO calculated in Python?

Python uses the **C3 Linearization algorithm** to calculate MRO.

Roman Urdu: C3 algorithm ensure karta hai ke subclass pehle aayein, aur kisi class ko dobara repeat na kiya jaye.

---

3. How to check the MRO of a class?

Use the built-in ClassName.mro() method.

Example:

python

CopyEdit

print(D.mro())

Roman Urdu: mro() function se aap check kar sakte hain ke Python kis tarteeb mein method dhoond raha hai.

---

4. MRO Example with Simple Inheritance

python

CopyEdit

```
class A:
    def greet(self): return "Hello from A"


class B(A):
    def greet(self): return "Hello from B"


class C(A):
    def greet(self): return "Hello from C"


class D(B, C): pass


d = D()
print(d.greet())        # Hello from B
print(D.mro())          # [D, B, C, A, object]
```

Roman Urdu: Jab d.greet() call hota hai to sabse pehle D, phir B, phir C, phir A ko check kiya jata hai. B mein method milta hai, aur wahi run hota hai.

---

5. MRO Example with Diamond Inheritance

python

CopyEdit

```
class X:
    def greet(self): return "Hello from X"

class Y(X):
    def greet(self): return "Hello from Y"

class Z(X):
    def greet(self): return "Hello from Z"

class W(Y, Z): pass

w = W()
print(w.greet())        # Hello from Y
print(W.mro())          # [W, Y, Z, X, object]
```

Roman Urdu: W class mein greet method nahi hai, to Python pehle Y ko check karta hai (found), aur wahi method call hota hai. MRO ensure karta hai ke order consistent rahe.

---

6. Why is MRO important?

It avoids ambiguity and confusion when multiple parent classes define the same method or attribute.

Roman Urdu: MRO help karta hai Python ko decide karne mein ke kis method ko run karna hai jab multiple options hon.

Q1: What is a class decorator in Python?

Answer:
A class decorator is a function that takes a class as input and returns a modified version of that class. It's used to add or change behavior of the class without changing its source code.

Roman Urdu Explanation:
Class decorator aik function hota hai jo aik class ko input leta hai aur us class mein koi naya behavior add karke modify kar deta hai. Ye class ke code ko directly touch kiye bina us mein changes laata hai.

Q2: How does the @property decorator work in Python?

Answer:
The @property decorator allows you to turn a method into a read-only property. It lets you access the method like an attribute (without parentheses).

Roman Urdu Explanation:
@property decorator kisi method ko property banata hai jise method ke bajaye attribute ki tarah use kiya ja sakta hai — yani () lagane ki zarurat nahi hoti.

Q3: What is the use of the @setter decorator?

Answer:
The @setter decorator is used with a property to define how its value can be modified. It allows you to validate or control value assignment.

Roman Urdu Explanation:
@setter decorator se aap property ki value change karne ka tareeqa define karte ho — is se aap value set karne se pehle check bhi laga sakte ho.

Q4: What does the @deleter decorator do in Python?

Answer:
The @deleter decorator allows you to define a method that will be executed when the property is deleted using del.

Roman Urdu Explanation:
@deleter decorator aapko yeh define karne deta hai ke jab property delete ki jaye (del use kar ke), to kya function chale.

Q5: Can you give an example of a class decorator in Python?

Answer:
Yes. Here's an example:

python

CopyEdit

```
def add_greeting(cls):
    def greet(self):
        return f"Hello from {self.__class__.__name__}"
    cls.greet = greet
    return cls


@add_greeting
class Person:
    def __init__(self, name):
        self.name = name


p = Person("Ali")
print(p.greet())  # Output: Hello from Person
```

Roman Urdu Explanation:
Upar diye gaye example mein add_greeting aik decorator hai jo class mein aik naya method greet() add karta hai. Jab hum @add_greeting likhte hain, to Person class automatically modify ho jaati hai.

Q6: How can decorators be used to count function calls?

Answer:
By creating a class with __call__() method, you can track how many times a function is called:

python

CopyEdit

```
class CountCalls:
    def __init__(self, func):
        self.func = func
        self.calls = 0

    def __call__(self, *args, **kwargs):
        self.calls += 1
        print(f"Call {self.calls}")
        return self.func(*args, **kwargs)

@CountCalls
def greet(name):
    print(f"Hi {name}")

greet("Ali")
greet("Zara")
```

Roman Urdu Explanation:
CountCalls aik decorator class hai jo function call hone par __call__ method chalaata hai. Har baar call honay par count barhta hai — yani track hota hai ke function kitni dafa chala.

Q7: What is a computed property and how is it useful?

Answer:
A computed property is a property that returns a calculated value based on other attributes. It is defined using @property.

Example:

```
python
CopyEdit
class Person:
    def __init__(self, weight, height):
        self.weight = weight
        self.height = height

    @property
    def bmi(self):
        return self.weight / (self.height ** 2)
```

Roman Urdu Explanation:
Computed property wo hoti hai jo kisi formula ya calculation par base karti hai. Jaise BMI example mein, bmi attribute weight aur height ki base par calculate hota hai.

Q8: Why are property decorators useful in OOP?

Answer:
They let you control how data is accessed, changed, or deleted while keeping a clean and simple interface.

Roman Urdu Explanation:
Property decorators se aap data ka access, uski setting ya deletion control kar sakte ho, aur code bhi readable aur maintainable rehta hai.

Q1: What is a callable in Python?

Answer:
In Python, a callable is an object that can be invoked or called like a function. If an object has a __call__() method, it becomes callable.

Roman Urdu Explanation:
Python mein, callable aik aisi cheez hoti hai jo function ki tarah call ki ja sakti hai. Agar kisi object ke paas __call__() method ho, to wo callable ho jata hai.

Q2: Can you give an example of a callable object?

Answer:
Yes, here's an example where a class defines the __call__() method, making its instances callable:

python

CopyEdit

```
class MyClass:
    def __call__(self):
        print("I'm callable!")


obj = MyClass()
obj()  # Output: I'm callable!
```

Roman Urdu Explanation:
Jab ek class __call__() method define karti hai, to us class ka object callable ban jata hai. Upar wale example mein, obj() ko call karte hi "I'm callable!" print hota hai.

Q3: How can we check if an object is callable in Python?

Answer:
You can use the callable() function to check if an object is callable:

python

CopyEdit

```
print(callable(obj))  # Output: True
```

Roman Urdu Explanation:
Agar aap yeh check karna chahte hain ke koi object callable hai ya nahi, to aap callable() function ka use kar sakte ho. Agar wo callable ho, to True return hoga.

Q4: What is the difference between functions, methods, and classes being callable?

Answer:

- **Functions**: These are naturally callable as they are defined with the def keyword.
- **Methods**: These are functions that belong to a class and are callable through an instance of that class.
- **Classes**: You can call a class like a function, and it will return a new instance of that class.

Roman Urdu Explanation:

- **Functions**: Ye naturally callable hote hain aur aap inhe direct function ki tarah call kar sakte ho.
- **Methods**: Ye class ka part hote hain aur class ke object se call kiye jaate hain.
- **Classes**: Aap class ko bhi function ki tarah call kar sakte ho, jisme wo class ka naya instance return karegi.

Q5: Can you explain how modules and packages are organized in Python?

Answer:

- A **module** is a single Python file containing classes, functions, and variables.
- A **package** is a directory containing multiple modules and an __init__.py file to make it a package.

Roman Urdu Explanation:

- **Module** aik Python file hota hai jisme classes, functions, aur variables hote hain.
- **Package** aik directory hoti hai jisme multiple modules aur __init__.py file hota hai jo usay package bana deta hai.

Q6: How do you import classes from modules in Python?

Answer:
You can import classes from modules using the import statement:

python

CopyEdit

from animals import Dog, Cat

Roman Urdu Explanation:
Aap import statement ke zariye classes ko module se import karte ho. Jaise from animals import Dog, Cat se Dog aur Cat classes ko import kiya ja sakta hai.

Q7: Can you explain how to structure a Python project with modules and packages?

Answer:
Here's an example project structure:

markdown

CopyEdit

my_project/

│

├── animals/

```
│   ├── __init__.py
│   ├── mammals.py
│   └── birds.py
│
├── vehicles/
│   ├── __init__.py
│   ├── cars.py
│   └── bikes.py
│
└── main.py
```

In main.py, you can import classes from animals and vehicles packages:

python

CopyEdit

```
from animals import Dog, Cat

from vehicles import Car, Bike
```

Roman Urdu Explanation:
Is example mein, animals aur vehicles folders mein modules hain. Har folder mein __init__.py hota hai jo us folder ko package banaata hai. main.py mein aap classes ko directly import karte ho jese from animals import Dog, Cat.

Q8: What is the benefit of organizing your code into modules and packages?

Answer:
Organizing code into modules and packages improves **readability**, **maintainability**, and **scalability**. It helps keep the code clean and easier to manage, especially as the project grows.

Roman Urdu Explanation:
Modules aur packages mein code ko organize karne se aapka code **readable**, **maintainable**, aur **scalable** hota hai. Yeh aapke project ko manage karne mein madad karta hai, khaas tor par jab project bara ho.

Q9: What does the __init__.py file do in a package?

Answer:
The __init__.py file is used to mark a directory as a package. It allows you to import the contents of the directory as a module.

Roman Urdu Explanation:
__init__.py file directory ko package banaati hai. Iske zariye aap us directory ke contents ko module ke roop mein import kar sakte hain.

Q10: How can you use modules in the main program?

Answer:
Once the modules are imported, you can create instances of the classes and call their methods:

python

CopyEdit

```
dog = Dog()
```

```
cat = Cat()

car = Car("Toyota Corolla")

bike = Bike("Harley Davidson")
```

Roman Urdu Explanation:
Jab aap modules ko import kar lete hain, to aap un modules ke classes ke instances create kar sakte hain aur unke methods ko call kar sakte hain.