✅ Basic-Level Questions

1. **Q: What is a class in Python?**

   In Python, a class is a blueprint for creating objects. It defines a set of attributes (data) and methods (functions) that describe the behavior and state of the objects created from the class.

2. **Q: How do you define a class in Python?** $\longrightarrow$

```
class ClassName:
    def __init__(self,
parameters):
        self.attribute = value

    def method_name(self):
        # method code
```

3. **Q: What is the difference between a class and an object?**

   A: Class: Like a blueprint for a house.

   Object: The actual house built from that blueprint.

4. **Q: What is the `__init__()` method?**

   The __init__() method in Python is a special method used to initialize a newly created object of a class. It is commonly known as the constructor.

5. **Q: How do you create an object from a class?**

   A: To create an object from a class in Python, you simply call the class as if it were a function, passing the required arguments defined in the __init__() method. object_name = ClassName(arguments)

✅ Intermediate-Level Questions

6. **Q: What is `self` in Python class methods?**

   A:. self refers to the current object instance.

   Used to access instance variables and methods inside the class.

---

7. **Q: What is a class variable vs an instance variable?**

   A: Class variable: Shared by all objects (e.g., school_name).

   Instance variable: Unique to each object (e.g., self.name).

---

8. **Q: What are class methods and static methods?**

   A: Class method: Uses @classmethod and cls as the first argument.

   Static method: Uses @staticmethod and has no automatic first argument.

---

9. **Q: Can you make variables private in Python?**.

   self.__private_var = value

---

10. **Q: Can a class inherit from another class in Python?**

```
class Animal:
    pass

class Dog(Animal):
    pass
```

🔹 Basic-Level Questions

11. **Q: What is an object in Python?**
A: An object is an instance of a class with its own data and behavior.

---

12. **Q: How do you create an object in Python?**
A: obj = ClassName(arguments)

---

13. **Q: What is the difference between a class and an object in Python?**
A: Class: Blueprint
1. .
Object: Real instance with data

---

14. **Q: What is the `__init__()` method in Python?**
A: A special method that runs when an object is created to initialize attributes.

15. **Q: How can you access object attributes in Python?**
A: print(obj.attribute_name)

🔹 Intermediate-Level Questions

16. **Q: What is the use of `self` in Python classes?**.
It allows each object to maintain its own state and access instance variables.

---

17. **Q: Can you modify an object's attribute after it is created?**
A: Yes, just assigning a new value
obj.name = "New Name"

---

18. **Q: What is object identity in Python? How do you check it?**
A: Object identity is its memory address.

Check with is: a is b (same object in memory)

19. **Q: What is the difference between `is` and ==?**
is: Checks identity (same memory) // ==: Checks value equality
20. **Q: What are magic methods or dunder methods in Python objects?**
A: Special methods with __ (e.g., __init__, __str__, __add__) that define object behavior.

🔹 Basic-Level Questions

21. **Q: What is Object-Oriented Programming?**
 A: A programming style based on objects and classes that model real-world entities.

---

22. **Q: What are the main OOP principles in Python?**
 A: Encapsulation, Inheritance, Polymorphism, Abstraction

---

23. **Q: How does Python support OOP?**
 A:. Python allows you to define classes, inherit, and use features like magic methods, making it a fully OOP-capable language.

---

24. **Q: What is a class and an object in Python?**
 A: Class: Code structure to define attributes and behavior

 Object: Actual instance of a class

25. **Q: How is encapsulation implemented in Python?**
 A: By hiding internal state using private variables and providing public methods.

🔹 Intermediate-Level Questions

26. **Q: What is inheritance in Python?**
 A: One class (child) can inherit from another (parent) to reuse code.

27. **Q: What is method overriding?**
 A: Child class redefines a method from the parent class.

28. **Q: What is polymorphism? Give an example.**
 A: $\longrightarrow$ Same method name behaves differently depending on the object type.

 Example:

 python
 Copy
 Edit

```
class Dog:
    def speak(self): return "Woof"

class Cat:
    def speak(self): return "Meow"
```

29. **Q: What is abstraction in Python?**
 A: Hiding complex details and showing only the essential features.
 Use abstract base classes with the abc module.

30. **Q: What is multiple inheritance in Python?**
 A: A class inherits from more than one parent class.

 python
 Copy
 Edit

```
class A: pass
class B: pass
class C(A, B): pass
```