What is Object-Oriented Programming (OOP)?
Object-Oriented Programming is a programming style where everything is treated as an object. It helps structure code into reusable pieces and is based on concepts like classes, objects, inheritance, and polymorphism.

What is a class in OOP?
A class is like a blueprint or template for creating objects. It defines the attributes and behaviors (methods) that the objects created from it will have.

What is an object in OOP?
An object is an instance of a class. It contains data (attributes) and functions (methods) that define its behavior.

What is the difference between abstraction and encapsulation?
Abstraction hides the complex implementation and shows only the necessary parts, while encapsulation hides the internal state of an object and only allows changes through methods.

What are dunder methods in Python?
Dunder methods (like `__init__`, `__str__`, etc.) are special methods with double underscores before and after their names. They are used to customize how objects behave with built-in Python functions and operators.

Explain the concept of inheritance in OOP
Inheritance allows a class to inherit properties and methods from another class, helping in code reuse and building hierarchies.

What is polymorphism in OOP?
Polymorphism means "many forms." It allows different classes to define methods with the same name, and the correct one is chosen based on the object calling it.

How is encapsulation achieved in Python?
Encapsulation in Python is done by using private variables (with a _ or __ prefix) and providing public methods (getters/setters) to access or modify them.

What is a constructor in Python?
A constructor is a special method called `__init__` that runs automatically when an object is created. It's used to initialize the object's attributes.

What are class and static methods in Python?
Class methods use `@classmethod` and take `cls` as the first argument; they can change class state. Static methods use `@staticmethod` and don't access class or instance data.

What is method overloading in Python?

Python doesn't support traditional method overloading, but we can mimic it using default arguments or `*args` and `**args`.

What is method overriding in OOP?
Method overriding is when a subclass provides a specific implementation of a method that's already defined in its parent class.

What is a property decorator in Python?
The `@property` decorator lets us define methods that can be accessed like attributes. It's useful for making class attributes readable but controlled.

Why is polymorphism important in OOP?
Polymorphism makes code more flexible and easier to maintain. It lets us write code that works on objects of different classes as long as they follow the same interface.

What is an abstract class in Python?
An abstract class is a class that can't be instantiated. It's used to define a common interface for its subclasses. We use the `abc` module for this.

What are the advantages of OOP?
OOP helps with code organization, reusability, and maintenance. It also makes it easier to model real-world systems and work with large codebases.

What is the difference between a class variable and an instance variable?
A class variable is shared across all instances, while an instance variable is specific to each object.

What is multiple inheritance in Python?
Multiple inheritance means a class can inherit from more than one parent class. Python supports it, but it can be tricky if the parent classes have methods with the same name.

Explain the purpose of `__str__` and `__repr__` methods in Python
`__str__` returns a readable string for the user, and `__repr__` returns a more detailed string meant for developers, often used for debugging.**What is the significance of the `super()` function in Python?**
`super()` is used to call methods from a parent class. It's helpful when working with inheritance, especially in constructors.

(MH What is the significance of the `__del__` method in Python?
The `__del__` method is called when an object is about to be destroyed. It's used for cleanup, like closing files or releasing resources.

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

`@staticmethod` doesn't take any special first argument (no `self` or `cls`). `@classmethod` takes `cls` and can access or modify class-level data.

How does polymorphism work in Python with inheritance?

With inheritance, polymorphism allows a child class to override a method from the parent class, and Python will call the correct version depending on the object type.

What is method chaining in Python OOP?

Method chaining is calling multiple methods on the same object in a single line, usually by returning `self` at the end of each method.

What is the purpose of the `__call__` method in Python?

`__call__` lets an object be called like a function. It's useful when we want an object to behave like a function.