OOPS

1. What is Object-Oriented Programming (OOP).
Oops stands for object oriented programming language,it is a programming model that organizes code using objects and classes. It helps structure programs with concepts like inheritance, encapsulation, and polymorphism.

2.What is a class in OOP?
Claas is a collection of object .
Class is a dataype of object.
Class is a blueprint of object.

3.What is an object in OOP?
An object is an instance of a class. It holds actual data and can use the class's defined methods.

4.What is the difference between abstraction and encapsulation?
Abstraction hides complex logic and shows only essential details. Encapsulation hides data using access control and groups data with the methods that use it.

5.What are dunder methods in Python?
Dunder methods (like __**`init`**__,`__str__`) are special methods with double underscores. They define how objects behave with built-in operations.

6.Explain the concept of inheritance in OOp?
One class can have inherit the properties and methods of another class this process is known ias inheritance.

7.What is polymorphism in OOP?
Polymorphism means many forms.
An entity can works in multiple role this capability is known as polymorphism.

8. How is encapsulation achieved in Python?
Encapsulation is done by making variables private using `_` or `__`, and controlling access with methods like getters and setters.

9. What is a constructor in Python?
A constructor is the `__init__` method, automatically called when an object is created. It initializes the object's attributes.

10.What are class and static methods in Python?
`@classmethod` takes `cls` and can modify class state. `@staticmethod` doesn’t take `self` or `cls`, and behaves like a regular function inside the class.

11.What is method overloading in Python?
Python does not support method overloading.
Method overloading means one class with same function name and different parameters.

12.What is method overriding in OOp?
Python supports method overriding.
Method overriding must haver function with same name and same parameters.

13.What is a property decorator in Python?
The `@property` decorator turns a method into a read-only attribute. It's useful for computed values or controlling attribute access.

14. Why is polymorphism important in OOP?
Polymorphism allows functions and methods to work with different object types, increasing flexibility and reducing code duplication.

15.What is an abstract class in Python?
An abstract class, created using the `abc` module, defines a common interface but can't be instantiated. Subclasses must implement its abstract methods.

16.What are the advantages of OOP?
OOP offers modularity, reusability, scalability, and easier maintenance. It models real-world problems more effectively using classes and objects.

17.What is the difference between a class variable and an instance variable?
Class variable we declare inside the class and we can use it inside the method as well .
Instance variable we declared outside the class and we can used it anywhere.

18.What is multiple inheritance in Python?
Multiple inheritance lets a class inherit from more than one parent class. It combines features from all base classes into one child class.

19.H Explain the purpose of ‘’__str__’ and ‘__repr__’ ‘ methods in Python?
`__str__` returns a readable string for users (used in `print()`), while `__repr__` returns a technical string for developers (used in console/debugging).

20.What is the significance of the ‘super()’ function in Python?
`super()` lets you call a method from the parent class, often used in `__init__` to inherit and initialize parent attributes.

21.What is the significance of the __del__ method in Python?
`__del__` is the destructor method, called when an object is deleted. It's used for cleanup like closing files or releasing resources.

22.What is the difference between @staticmethod and @classmethod in Python?
`@staticmethod` doesn’t access class or instance data. `@classmethod` takes `cls` as a parameter and can modify class-level variables.

23.How does polymorphism work in Python with inheritance?
Polymorphism works when a subclass overrides a method from its superclass. The correct method is chosen at runtime based on the object’s type.

24.What is method chaining in Python OOP?
Method chaining is calling multiple methods in one line, where each method returns `self`.
Example: `obj.method1().method2()`.

25.What is the purpose of the __call__ method in Python?
The `__call__` method lets an object be used like a function. When `obj()` is used, Python internally calls `obj.__call__()`.