**Object-Oriented Programming (OOP) – Theory Assignment**

1. **What is Object-Oriented Programming (OOP)?**
   * A programming paradigm based on objects that contain data (attributes) and methods (functions).
2. **What is a class in OOP?**
   * A blueprint for creating objects, defining attributes and methods that objects of that class will have.
3. **What is an object in OOP?**
   * An instance of a class with specific values assigned to its attributes.
4. **What is the difference between abstraction and encapsulation?**
   * **Abstraction:** Hides complex implementation details and shows only relevant features.
   * **Encapsulation:** Restricts direct access to an object's data by using access modifiers.
5. **What are dunder methods in Python?**
   * Special methods with double underscores (e.g., \_\_init\_\_, \_\_str\_\_) used to define behavior for built-in operations.
6. **Explain the concept of inheritance in OOP.**
   * A mechanism where a class (child) derives properties and methods from another class (parent).
7. **What is polymorphism in OOP?**
   * The ability of different classes to use the same method name but execute different behaviours.
8. **How is encapsulation achieved in Python?**
   * Using private (\_\_var), protected (\_var), and public (var) variables and getter/setter methods.
9. **What is a constructor in Python?**
   * A special method (\_\_init\_\_) automatically called when an object is created to initialize attributes.
10. **What are class and static methods in Python?**
    * **Class Method:** Uses @classmethod, works with class-level data.
    * **Static Method:** Uses @staticmethod, doesn’t depend on class or instance attributes.
11. **What is method overloading in Python?**
    * Defining multiple methods with the same name but different parameters (not natively supported, achieved via default arguments).
12. **What is method overriding in OOP?**
    * Redefining a method in a child class that is already defined in the parent class.
13. **What is a property decorator in Python?**
    * @property is used to define a method as a property, allowing controlled access to instance variables.
14. **Why is polymorphism important in OOP?**
    * It increases flexibility by allowing a common interface for different data types and enhances code reusability.
15. **What is an abstract class in Python?**
    * A class that cannot be instantiated and must be inherited; defined using ABC and @abstractmethod.
16. **What are the advantages of OOP?**
    * Code reusability, modularity, scalability, security (encapsulation), and easier maintenance.
17. **What is the difference between a class variable and an instance variable?**
    * **Class Variable:** Shared across all instances of a class.
    * **Instance Variable:** Unique to each object.
18. **What is multiple inheritance in Python?**
    * A feature where a class can inherit from multiple parent classes.
19. **Explain the purpose of \_\_str\_\_ and \_\_repr\_\_ methods in Python.**
    * **\_\_str\_\_**: Returns a user-friendly string representation of an object.
    * **\_\_repr\_\_**: Returns an official string representation for debugging purposes.
20. **19. Significance of super():**  
    It allows you to call methods of a parent class without explicitly naming it, making inheritance cleaner and reducing code duplication.
21. **20. Significance of the \_\_del\_\_ method:**  
    It acts as a destructor, automatically invoked when an object is about to be garbage collected, allowing cleanup of resources.
22. **21. Difference between @staticmethod and @classmethod:**  
    A static method does not receive any implicit first argument, while a class method receives the class itself as its first argument, enabling modification of class state.
23. **22. How polymorphism works in Python with inheritance:**  
    Different classes can define methods with the same name, and the appropriate method is called based on the actual object's type at runtime.
24. **23. What is method chaining in Python OOP:**  
    It is a technique where methods return the object itself, allowing multiple method calls to be linked in a single, concise statement.
25. **24. Purpose of the \_\_call\_\_ method:**  
    It allows an instance of a class to be invoked as if it were a function, providing a way to customize function-like behavior for objects.