1. Difference Between Constructor and Destructor in Python

Constructor:

- A constructor is a special method (__init__) that is automatically called when an object is instantiated.

- Its primary purpose is to initialize the attributes of the object.

- Constructors can accept parameters to set initial values for the object's attributes.

Destructor:

- A destructor is a special method (__del__) that is automatically called when an object is about to be destroyed.

- Its main purpose is to perform any cleanup activities, such as closing files or releasing resources.

- Destructors are less commonly used in Python due to the automatic garbage collection provided by the interpreter.

2. Difference Between Public and Private Access Modifiers in Python

Public:

- Attributes and methods marked as public can be accessed from anywhere in the code, both inside and outside the class.

- This is the default access level in Python; no special notation is needed.

Private:

- Attributes and methods marked as private are intended to be accessible only within the class itself.

- In Python, private members are denoted by a double underscore prefix (e.g., __attribute).

- This helps to protect the data and prevents it from being accessed directly outside the class, encouraging encapsulation.

3. Method Overriding in Python Inheritance

Method Overriding:

- Method overriding occurs when a subclass provides a specific implementation of a method that is already defined in its superclass.

- When a method in a subclass has the same name, parameters, and functionality as a method in its parent class, the subclass method overrides the parent class method.

- This allows the subclass to modify or extend the behavior of the inherited method, enabling polymorphism and more flexible code design.

4. Abstraction in OOP and Its Role in Simplifying Complex Systems

Abstraction:

- Abstraction is a fundamental concept in object-oriented programming that involves hiding the complex implementation details of a system and exposing only the necessary parts to the user.

- It allows developers to work with simplified models of complex systems, focusing on high-level functionality rather than intricate internal workings.

Benefits of Abstraction:

1. Simplification: Users interact with a simplified interface, making it easier to understand and use the system without needing to know all the underlying details.

2.Encapsulation: By hiding implementation details, abstraction promotes encapsulation, ensuring that changes in the implementation do not affect users of the class.