**ADITYA AMIN**

**ASSIGN : 04**

Q1. Which two operator overloading methods can you use in your classes to support iteration?

\_\_next\_\_ method that defines how to retrieve the next item in the iteration. The \_\_iter\_\_ method is called when an iteration begins.

Q2. In what contexts do the two operator overloading methods manage printing?

\_\_str\_\_: This method is responsible for returning a string representation of the object. It is called by the built-in str() function and by the print() function when the object is passed as an argument to print(). The \_\_str\_\_ method should return a human-readable string that represents the object's state or value.

\_\_repr\_\_: This method is responsible for returning a string representation that is unambiguous and can be used to recreate the object. It is called by the built-in repr() function and by the interactive shell when the object is evaluated. The \_\_repr\_\_ method should return a string that provides a detailed and precise representation of the object.

Q3. In a class, how do you intercept slice operations?

To intercept slice operations in a class, you can define the \_\_getitem\_\_ method with appropriate logic to handle slicing. The \_\_getitem\_\_ method allows objects to implement the behavior of indexing and slicing using square brackets ([]).

Q4. In a class, how do you capture in-place addition?

To capture in-place addition operations in a class, you can define the \_\_iadd\_\_ method. The \_\_iadd\_\_ method is used to handle the in-place addition (+=) operation for objects of a class.

Q5. When is it appropriate to use operator overloading?

Operator overloading should be used when it enhances the readability and expressiveness of your code and provides a natural and intuitive way to work with objects of a class based on the semantics of the operator being overloaded.