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

The local scope of a class and a function differ in their behavior and the variables they encompass. Here are the key differences

Variables: In a class, variables defined within the class body are considered class-level variables or attributes.

Accessibility: Class-level variables in the local scope of a class can be accessed within any method of the class using the self keyword or the class name.

Lifetime: Class-level variables have a lifetime equal to the lifetime of the class instance. They persist as long as the class instance exists.

Usage: Class-level variables are commonly used to store data that is shared among different methods of the class or to define attributes that describe the characteristics or properties of the class instances.

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

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

\_\_repr\_\_(): This method is responsible for providing a detailed and unambiguous representation of the object. It should return a string that can be used to recreate the object or represent its internal state. The \_\_repr\_\_() method is used by the built-in repr() function to obtain the canonical string 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 and handle slice indexing using the slice() function. The \_\_getitem\_\_() method is an operator overloading method that allows you to customize the behavior of accessing elements or slices of an object

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

To capture in-place addition in a class, you can define the \_\_iadd\_\_() method. The \_\_iadd\_\_() method is an in-place addition operator overloading method that allows you to customize the behavior of the += operator for objects of your class

Q5. When is it appropriate to use operator overloading?

Operator overloading is appropriate in situations where it enhances the clarity and intuitiveness of code, improves code readability, and aligns with the natural behavior of objects in the domain being modeled.