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

Sol: to support iteration in a class we can use the following two operator overloading methods:

1. ‘\_\_iter\_\_()’
2. ‘\_\_next\_\_()”

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

Sol: The two operator overloading methods that manage printing are:

1. ‘\_\_str\_\_()’ method: This method is called by the ‘str()’ built-in function and is used to return a string representation of an object.
2. ‘\_\_repr\_\_()’ method: This method is called by the ‘repr()’ built-in function and is used to return a string representation of an object that can be used to recreate the object.

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

Sol: To intercept slice operations in a class you can define the ‘\_\_getitem\_\_()’ method which is called when the object is indexed with square brackets ‘[]’.

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

Sol: To capture in-place addition in a class, you can define the ‘\_\_iadd\_\_()’ method, which is called when the object is modified with the ‘+=’ operator.

Q5. When is it appropriate to use operator overloading?

Sol: Operator overloading can be used when we want to define custom behavior for built-in operators in python.