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

The \_\_iter\_\_ to return iterative object and run at the start of iteration and the \_\_next\_\_ can be used to return the next value and raised an error if we reach to the end on iteration

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

In class with the help of \_\_str\_\_ function we can overload print, and the other way is using builtins library and define a function with name “print” and redefine new task for print.

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

The \_\_getitem\_\_, \_\_setitem\_\_ or \_\_delitem\_\_ are the keywords which can be used inside the class to overload slicing and also for slicing same method can be used inside these functions to implement inserting, deleting and changing the values

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

We can redefine \_\_iadd\_\_(self, other) method inside the class to overload the in-place addition

Q5. When is it appropriate to use operator overloading?

If we want to define a class which is child of an abstract base class, it is better to overload operator because it helps users who are familiar with these operators and use them on other child of this abstract class can easily understand and perform your class, but in other cases it only increase complexity of the code, because there are lot of operator and implementing all with make your code lengthy.