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

Answer: First of all we use \_\_init\_ method for normal initialization and then use methods such as \_\_add\_\_() which can be explicitly called to add values for the different objects for same class and instead of \_\_add\_\_() method there can be several types of objects depending on the type of operations you want to perform on the 2 objects created.

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

Answer: When 2 different operators are compared in any form like <,>,<=,>= etc then inside print statement if we do comparison between operators with objects as operand then we can mange printing the output result as these will have respective overloading methods defined inside the classes of which objects are already created and they do perform the respective operations

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

Answer: In a class , a[i:j] is now used as a.\_\_getitem\_\_(slice(i,j)). Similarly we can use a.\_\_setitem()\_\_ and a.\_\_delitem\_\_() when used as an assignment or deletion target respectively.

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

Answer: We can do it using the following:

Let a = [1,2,4,5]

And we want to add [1,2,3] to this list then we do operator.iadd(a,[1,2,3]).

Q5. When is it appropriate to use operator overloading?

Answer: We cannot do arithmetic operations for operators as we do for variables. We need to define a specific method where we can load all existing operator and performs operations on those operators but cannot generate a new operator. In this case of getting a result from existing operators and returning the result is appropriate case of operator overloading.