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

**add** () and **mul** () methods can be used for both integers and string data objects. String is iterable object. When two strings are passed to + operator, it will return concatenated string. When \* (number) is preceded by string, then that string is repeated those many number of time.

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

It depends on the input parameter result is printed. Example, If both inputs are string function will print string output.

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

To intercept slice operations in a class, you can define the \_\_getitem\_\_() method. This method is called when the object is accessed using the square bracket notation ([]). By customize the behavior of the \_\_getitem\_\_() method based on your requirements. For example, you could add bounds checking, error handling, or modify the slicing behavior to suit your needs.

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

Python in its definition provides methods to perform inplace operations, i.e doing assignment and computation in a single statement using “operator” module. For example,

x += y is equivalent to x = operator.iadd(x, y)

iadd() function is used to assign and add the current value. This operation does “a+=b” operation. Assigning is not performed in case of immutable containers, such as strings, numbers and tuples.

import operator  
x = 1  
y = 2  
x = operator.iadd(x,y)  
print(x)

3

Q5. When is it appropriate to use operator overloading?

When we have two objects which are a physical representation of a class (user-defined data type) and we have to add two objects with binary ‘+’ operator it throws an error, because compiler don’t know how to add two objects. So we define a method for an operator and that process is called operator overloading.