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

The \_\_iter\_\_ returns the iterator object and is implicitly called at the start of loops. The \_\_next\_\_ method returns the next value and is implicitly called at each loop increment.

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

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

The slice() method returns a portion of an iterable as an object of the slice class based on the specified range.

### Syntax:

slice(stop)

slice(start, stop, step)

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)

Q5. When is it appropriate to use operator overloading?

**Operator Overloading** means giving extended meaning beyond their predefined operational meaning. For example operator + is used to add two integers as well as join two strings and merge two lists. It is achievable because ‘+’ operator is overloaded by int class and str class