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

1. The \_\_iter\_\_ returns the iterator object and is implicitly called at the start of loops.
2. The \_\_next\_\_ method returns the next value and is implicitly called at each loop increment. \_\_next\_\_ raises a StopIteration exception when there are no more value to return, which is implicitly captured by looping constructs to stop iterating.

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

1. To have a user-friendly string representation of the object which can be read by a normal user rather than the programmer, we can overload the \_\_str\_\_ method.
2. In cases where the \_\_str\_\_() method is not defined, Python uses the \_\_repr\_\_() method to print the object, as well as to represent the object when str() is called on it. This method can be overloaded to obtain the parsable string representation of an object. [1]

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

By overloading \_\_getitem\_\_(), \_\_setitem\_\_() and \_\_delitem\_\_() methods and providing appropriate definition for it.

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

In place addition where 1st argument stores the sum in it after addition can be captured using \_\_iadd\_\_ method. It should only be used with mutable objects. [2]

**Q5. When is it appropriate to use operator overloading?**

1. To overwrite the pre existing way in which the operator behaves
2. To handle the situations where the existing magic function throws an error as it is unsupported.
3. To extend the capability of an operator to work with user defined objects

**References**:

1. <https://realpython.com/operator-function-overloading/>
2. <https://www.geeksforgeeks.org/inplace-vs-standard-operators-python/>