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

Ans:

\_\_iter\_\_ () and \_\_ next\_\_() methods are used to support iteration

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

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

You need to provide custom [\_\_getitem\_\_()](http://docs.python.org/reference/datamodel.html#object.__getitem__), [\_\_setitem\_\_](http://docs.python.org/reference/datamodel.html#object.__setitem__) and [\_\_delitem\_\_](http://docs.python.org/2/reference/datamodel.html#object.__delitem__) hooks.

These are passed a [slice object](http://docs.python.org/library/functions.html#slice) when slicing the list; these have start, stop and step attributes. However, these values could be None, to indicate defaults. Take into account that the defaults actually change when you use a negative stride

However, they also have a [slice.indices() method](https://docs.python.org/reference/datamodel.html" \l "slice.indices), which when given a length produces a tuple of (start, stop, step) values suitable for a range() object

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

Ans: We can set up the in-place addition behavior for our own class by overriding the magic “dunder” method \_\_iadd\_\_(self, other) in our class definition.

Q5. When is it appropriate to use operator overloading?

Ans: The purpose of operator overloading is **to provide a special meaning of an operator for a user-defined data type**. With the help of operator overloading, you can redefine the majority of the python operators. You can also use operator overloading to perform different operations using one operator.