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

***Ans.***The \_\_iter\_\_ and the \_\_next\_\_ methods can support iteration.

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

***Ans.***

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

***Ans.***The slice() method returns a portion of an iterable as an object of the slice class based on the specified range. It can be used with [string](https://www.tutorialsteacher.com/python/python-string), [list](https://www.tutorialsteacher.com/python/python-list), [tuple](https://www.tutorialsteacher.com/python/python-tuple), [set](https://www.tutorialsteacher.com/python/python-set), bytes, or [range](https://www.tutorialsteacher.com/python/range-method) objects or [custom class](https://www.tutorialsteacher.com/python/python-class) object.

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

***Ans.***[In-place assignment operators](https://blog.finxter.com/python-operators-overview/" \l "In-Place_Assignment_Operators" \o "https://blog.finxter.com/python-operators-overview/#In-Place_Assignment_Operators" \t "https://blog.finxter.com/python-in-place-addition-operator/_blank) perform an operation in-place on a variable provided as first operand. They overwrite the value of the first operand variable with the result of the operation when performing the operator without assignment.

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

***Ans.***When one or both operands are of a user-defined class or structure type, operator overloading makes it easier to specify user-defined implementation for such operations. This makes user-defined types more similar to the basic primitive data types in terms of behaviour.