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

Which method should I implement to overload the operator in a class?

To overload the + operator, we will need to implement \_\_add\_\_() function in the class. With great power comes great responsibility. We can do whatever we like inside this function. But it is more sensible to return the Point object of the coordinate sum.

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

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.

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. It can be used with string, list, tuple, set, bytes, or range objects or custom class object that implements sequence methods \_\_getitem\_\_() and \_\_len\_\_() methods.

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

Python provides the operator x += y to add two objects in-place by calculating the sum x + y and assigning the result to the first operands variable name x . You can set up the in-place addition behavior for your own class by overriding the magic “dunder” method \_\_iadd\_\_(self, other) in your class definition.

Q5. When is it appropriate to use operator overloading?

It allows us to provide an intuitive interface to our class users, plus makes it possible for templates to work equally well with classes and built-in types. Operator overloading allows C++ operators to have user-defined meanings on user-defined types or classes.