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

**ANSWER:** 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. \_\_next\_\_ raises a Stop Iteration 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?**

**ANSWER:**  \_\_repr\_\_ (or its close relative, \_\_str\_\_) is called automatically when class instances are printed or converted to strings. These methods allow you to define a better display format for your objects than the default instance display.

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

**ANSWER**: To intercept slice operations in a class, you need to override the ‘\_\_getitem\_\_’ and ‘\_\_setitem\_\_’ methods. These methods are called when you try to access or modify an item using the slice notation.

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

**ANSWER:** 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?**

**ANSWER:** Operator overloading is mostly useful when you're making a new class that falls into an existing ‘‘Abstract Base Class’’ (ABC)