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

Ans: Classes can support iteration by defining (or inheriting) __getitem__ or __iter__. In all iteration contexts, Python tries to use __iter__ first, which returns an object that supports the iteration protocol with a __next__ method: if no __iter__ is found by inheritance search, Python falls back on the __getitem__ indexing method, which is called repeatedly, with successively higher indexes. If used, the yield statement can create the __next__ method automatically.

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

Ans: The __str__ and __repr__ methods implement object print displays. The former is called by the print and str built-in functions; the latter is called by print and str if there is no __str__, and always by the repr built-in, interactive echoes, and nested appearances. That is, __repr__ is used everywhere, except by print and str when a __str__ is defined. A __str__ is usually used for user-friendly displays; __repr__ gives extra details or the object's as-code form.

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

Ans: Slicing is caught by the __getitem__ indexing method: it is called with a slice object, instead of a simple integer index, and slice objects may be passed on or inspected as needed.

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

Ans: In-place addition tries __iadd__ first, and __add__ with an assignment second. The same pattern holds true for all binary operators. The __radd__ method is also available for right-side addition.

Q5. When is it appropriate to use operator overloading?

Ans: When a class naturally matches, or needs to emulate, a built-in type's interfaces. For example, collections might imitate sequence or mapping interfaces, and callables might be coded for use with an API that expects a function. You generally shouldn't implement expression operators if they don't

naturally map to your objects naturally and logically, though—use normally named methods instead.