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

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 StopIteration exception when there are no more value to return

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

The str() built-in is used to cast an instance of a class to a str object, or more appropriately, to obtain a user-friendly string representation of the object which can be read by a normal user rather than the programmer. You can define the string format your object should be displayed in when passed to str() by defining the \_\_str\_\_() method in your class. Moreover, \_\_str\_\_() is the method that is used by Python when you call [print()](https://realpython.com/python-print/) on your object.

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

import profile

import sys

print sys.version

class InterceptedList(list):

def addSave(func):

def newfunc(self, \*args):

func(self, \*args)

print 'saving'

return newfunc

\_\_setslice\_\_ = addSave(list.\_\_setslice\_\_)

\_\_delslice\_\_ = addSave(list.\_\_delslice\_\_)

class InterceptedList2(list):

def \_\_setitem\_\_(self, key, value):

print 'saving'

list.\_\_setitem\_\_(self, key, value)

def \_\_delitem\_\_(self, key):

print 'saving'

list.\_\_delitem\_\_(self, key)

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

X=+y

Q5. When is it appropriate to use operator overloading?

Operator overloading is mostly useful when you're making a new class that falls into an existing "Abstract Base Class" (ABC) -- indeed, many of the ABCs in standard library module [collections](http://docs.python.org/library/collections.html#module-collections) rely on the presence of certain special methods (and special methods, one with names starting and ending with double underscores AKA "dunders", are exactly the way you perform operator overloading in Python). This provides good starting guidance.