**1. What is the concept of an abstract superclass?**

An abstract superclass is a common superclass for several subclasses. It is considered as a blueprint for other classes. It allows you to create a set of methods that must be created within any child classes built from the abstract class. It is used for defining a common Application Program Interface(API) for a set of subclasses

**2. What happens when a class statement's top level contains a basic assignment statement?**

When a class definition is entered, a new namespace is created, and used as the local scope — thus, all assignments to local variables go into this new namespace. [1]

**3. Why does a class need to manually call a superclass's \_\_init\_\_ method?**

If we don’t call the superclass’s \_\_init\_\_ method, then only the instructions in the calling class’s constructor will get executed. To initialize the attributes of super class, the code needs to be repeated in the child class as well. We can avoid this repetition of code by calling the superclass’s \_\_init\_\_ method.

**4. How can you augment, instead of completely replacing, an inherited method?**

This can be done by calling the original version of the method with augmented arguments.

Example:

class Manager(Person):

def giveRaise(self, percent, bonus=.10):

Person.giveRaise(self, percent + bonus)

**5. How is the local scope of a class different from that of a function?**

The local namespace for a function is created when the function is called, and deleted when the function returns or raises an exception that is not handled within the function.Recursive invocations each have their own local namespace.

When a class definition is entered, a new namespace is created, and used as the local scope — thus, all assignments to local variables go into this new namespace. In particular, function definitions bind the name of the new function here.

**References**:

1. https://docs.python.org/3/tutorial/classes.html