1. What is the concept of an abstract superclass?

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

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

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

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

Solutions:

1. An abstract superclass is a class that is designed to be inherited by other classes. It serves as a template or blueprint for its subclasses, providing common attributes and behavior that can be shared among multiple related classes. An abstract superclass may contain abstract methods, which are methods without an implementation. Subclasses inheriting from the abstract superclass are required to provide an implementation for these abstract methods. The purpose of an abstract superclass is to define a common interface and behavior that subclasses should adhere to, while allowing each subclass to have its own specific implementation.

2. When a class statement's top level contains a basic assignment statement, it creates a class attribute. This attribute will be shared by all instances (objects) of the class. It is essentially a variable that belongs to the class itself, rather than any specific instance of the class. Any modifications made to this class attribute will be reflected in all instances of the class.

3. A class needs to manually call a superclass's `\_\_init\_\_` method when it wants to initialize the attributes defined in the superclass. By calling the superclass's `\_\_init\_\_` method, the subclass can inherit and initialize the attributes and behavior defined in the superclass. This allows the subclass to reuse and extend the functionality of the superclass without having to redefine everything from scratch. The `\_\_init\_\_` method of the superclass is typically called using the `super()` function within the `\_\_init\_\_` method of the subclass.

4. To augment an inherited method, you can override the method in the subclass while still invoking the superclass's version of the method. This can be achieved by calling the superclass's method within the subclass's method using the `super()` function. By doing so, you can add additional functionality to the inherited method without completely replacing it. This is often referred to as method overriding with super() call. The subclass can perform its specific operations before or after calling the superclass's method, allowing for a combination of the original behavior and additional functionality.

5. The local scope of a class and a function differ in terms of the variables and attributes they can access.

In the local scope of a class, you can access and manipulate the attributes defined within the class using the `self` keyword. These attributes are shared by all instances of the class. Additionally, you can access class-level attributes, which are variables defined directly within the class but outside any methods. Class-level attributes are shared by all instances of the class and can be accessed using the class name.

In the local scope of a function, you can only access the variables defined within that function or passed as arguments. You cannot directly access the attributes of a class or other variables defined outside the function's scope unless they are explicitly passed as arguments. Function-local variables are typically temporary and exist only within the function's execution. Each time the function is called, a new set of local variables is created.