1. What is the concept of an abstract superclass?

An abstract superclass is a class that is designed to be inherited by other classes but not meant to be instantiated on its own. It provides a common interface and often includes abstract methods (methods without implementations). Subclasses that inherit from the abstract superclass must provide concrete implementations for these abstract methods. Abstract superclasses are used to define common behavior and structure that should be shared by multiple subclasses.

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

If a class statement's top level contains a basic assignment statement (a statement that assigns a value to a variable), the assignment statement becomes part of the class's namespace. It does not affect the class's methods or behavior. The assigned value can be accessed using the class name.

3. Why does a class need to manually call a superclass's init method?

When a class inherits from a superclass, it can extend or override the superclass's behavior. However, the superclass's \_\_init\_\_ method is not automatically called when creating an instance of the subclass. To ensure that the initialization code of the superclass is executed, the subclass needs to manually call the superclass's \_\_init\_\_ method using super().\_\_init\_\_() within its own \_\_init\_\_ method.

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

To augment an inherited method without completely replacing it, you can use method overriding. In the subclass, define a method with the same name as the method you want to modify. Use the super() function to call the superclass's version of the method, and then add your additional behavior before or after the call. This way, you retain the original functionality while adding the desired changes.

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

In a class, the local scope refers to the namespace within the class's methods and attributes. When you define a variable within a method of a class, it is considered local to that method and can only be accessed within that method or other methods of the same class. However, variables defined at the class level (outside methods) are accessible across all methods of the class.

In a function, the local scope refers to the namespace within the function. Variables defined within a function are local to that function and can only be accessed within the function itself.

In both cases, local variables are limited in scope and are not accessible outside the context where they are defined.