1. What is the concept of an abstract superclass?

An abstract superclass, also known as an abstract base class (ABC) in Python, is a class that cannot be instantiated on its own and is meant to be subclassed by other classes. It provides a common interface and possibly some shared behavior for its subclasses, but it typically includes one or more abstract methods that must be implemented by any concrete subclass.

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

When a basic assignment statement appears at the top level of a class definition in Python, it defines a class attribute. This attribute is shared across all instances of the class and is typically used to store information or constants relevant to the class as a whole.

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

In Python, a class needs to manually call a superclass’s \_\_init\_\_ method to ensure that the initialization logic defined in the superclass is executed. This is important for properly setting up an instance with all the attributes and behaviors that the superclass provides. Here’s why and how to do it:

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

To augment, rather than completely replace, an inherited method in Python, you can extend the functionality of the method by adding additional behavior while still calling the original method. This is often done using the super() function to invoke the superclass’s method and then adding your own code.

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

Class Scope: Contains class attributes and methods, affecting all instances of the class and extending throughout the class body.

Function Scope: Contains local variables specific to the function and does not affect or have access to variables outside the function unless explicitly stated.