1. What is the concept of an abstract superclass?

A1. An abstract superclass in Python is a class that is intended to be used only as a base class for other classes, and is not meant to be instantiated on its own. It is an abstract representation of a concept or functionality that is shared by multiple classes.

An abstract superclass may define abstract methods, which are methods that have no implementation in the superclass, but must be implemented in any concrete subclass that inherits from the superclass. This allows for polymorphism, where different subclasses can have their own implementation of the abstract method, but can still be treated as instances of the same superclass.

The concept of an abstract superclass is a key aspect of object-oriented programming, as it allows for modular and extensible code that can be easily customized and adapted to different use cases.

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

A2. When a class statement's top level contains a basic assignment statement, a class-level variable is created. This variable is shared by all instances of the class and can be accessed using the class name followed by the variable name. Assigning a value to a class-level variable affects all instances of the class, while assigning a value to an instance-level variable only affects the specific instance that it belongs to. This makes class-level variables useful for storing information that is shared by all instances of a class, while instance-level variables are useful for storing information that is specific to each instance.

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

A3. A class needs to manually call a superclass's **\_\_init\_\_** method in order to properly initialize any instance-level variables that are defined in the superclass. If the subclass does not call the superclass's **\_\_init\_\_** method explicitly, those instance-level variables will not be initialized, and the subclass may not function properly.

Additionally, the superclass's **\_\_init\_\_** method may perform other important initialization tasks, such as setting up connections to databases or other external resources, that are necessary for the proper functioning of the subclass. Therefore, it is important for the subclass to call the superclass's **\_\_init\_\_** method in order to ensure that the subclass is properly initialized and functioning as intended.

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

A4. To augment an inherited method instead of completely replacing it in Python, you can use the **super()** function to call the superclass's implementation of the method, and then modify or extend the result as needed. By calling the superclass's implementation first, you can preserve the original behavior of the method while still adding or modifying functionality in the subclass. To do this, define a new method in the subclass with the same name as the method in the superclass, and use **super()** to call the superclass's implementation. Then, modify the result of the superclass's implementation to add or change behavior as needed, and return the modified result.

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

A5. The local scope of a class in Python is different from that of a function in a few ways. First, a class defines a new namespace that is separate from the global namespace and any other namespaces defined in the program. This means that names defined within a class are not accessible from outside the class, unless they are explicitly exposed via class methods or attributes.

Second, methods defined within a class have access to the class's namespace as well as the global namespace, whereas functions defined outside of a class only have access to the global namespace and any namespaces that are explicitly passed as arguments.

Finally, unlike function scopes, class scopes are not created at runtime; they are created when the class is defined and remain unchanged for the life of the program.