1. What is the concept of an abstract superclass?

An abstract superclass is a class that is meant to be inherited by other classes, and it may contain abstract methods or attributes. Abstract methods are methods that are declared in the superclass but have no implementation. Subclasses must provide their own implementation for these abstract methods. Abstract superclasses are typically used to define a common interface for a group of related classes.

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

When a class statement's top level contains a basic assignment statement, it defines a class attribute. This attribute is shared among all instances of the class and is associated with the class itself rather than instances of the class.

Example:

class MyClass:

class\_attribute = "I am a class attribute"

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

A class needs to manually call a superclass's \_\_init\_\_ method if it wants to initialize the attributes defined in the superclass. This is necessary when the subclass has its own \_\_init\_\_ method and wants to extend the initialization process rather than replace it entirely. The super() function is commonly used to call the superclass's \_\_init\_\_ method.

Example:

class Parent:

def \_\_init\_\_(self, x):

self.x = x

class Child(Parent):

def \_\_init\_\_(self, x, y):

super().\_\_init\_\_(x) # Call the \_\_init\_\_ method of the superclass

self.y = y

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

To augment, instead of completely replacing, an inherited method, you can call the superclass's method using super() and then add or modify functionality as needed in the subclass.

Example:

class Parent:

def method(self):

print("Original method in Parent")

class Child(Parent):

def method(self):

super().method() # Call the method of the superclass

print("Additional functionality in Child")

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

The local scope of a class is different from that of a function in terms of access to variables. In a class, variables defined within methods are considered attributes and can be accessed using self. These attributes are accessible throughout the class. In a function, variables defined within the function are limited to the scope of that function and are not accessible outside of it.