1. What is the concept of an abstract superclass?

Ans:  
We can treat an abstract class as a superclass and extend it; its subclasses can override some or all of its inherited abstract methods. If through this overriding a subclass contains no more abstract methods, that class is concrete (and we can construct objects directly from it).

2. What happens when a class statement's top level contains a basic assignment statement?  
ans:  
In Python, when a class statement's top level contains a basic assignment statement, it creates a class-level attribute or variable that is shared among all instances (objects) of that class. Class-level attributes are also sometimes referred to as "class variables" because they belong to the class itself rather than to individual instances of the class.

3. Why does a class need to manually call a superclass's \_\_init\_\_ method?  
ans:  
A class needs to manually call a superclass's **\_\_init\_\_** method in Python because it ensures proper initialization of attributes and behavior inherited from the superclass, allows for customization in the subclass's initialization, and sets up the common state defined in the superclass. This manual call is typically done using **super().\_\_init\_\_()** inside the subclass's **\_\_init\_\_** method.

4. How can you augment, instead of completely replacing, an inherited method?  
Ans: You can augment (extend) an inherited method in a subclass without completely replacing it by following these steps in Python:

Call the Superclass Method: Inside the subclass's method, call the superclass's method using super(). This allows you to execute the code in the superclass's method before or after your custom code.

Add Your Custom Code: After calling super().method\_name(), you can include your custom code to extend or modify the behavior of the inherited method.

5. How is the local scope of a class different from that of a function?  
Ans:  
The local scope of a function contains variables and is limited to that function, while the local scope of a class contains attributes and methods shared among all instances of the class.

Function-local variables have a shorter lifetime, limited to the function call, while class-level attributes and methods exist as long as the class definition is in memory.

Classes are a core concept in object-oriented programming (OOP), whereas functions can be used in both procedural and OOP paradigms.