1. **What is the concept of an abstract superclass?**

Ans :- An abstract superclass is one way to provide re-usable code. You can extend the abstract class and inherit the code. This is sometimes more convenient than using static methods or object composition to share code. The abstract class can "fix" parts of the code (by making it final).

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

Ans:- We use **Python assignment statements** to assign objects to names. The target of an assignment statement is written on the left side of the equal sign (=), and the object on the right can be an arbitrary expression that computes an object.

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

Ans:- The main reason for always calling base class \_init\_\_ is that base class may typically create member variable and initialize them to defaults. So if you don't call base class init, none of that code would be executed and you would end up with base class that has no member variables

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

Ans:- A more sophisticated way to augment an inherited method involves forwarding. Message forwarding allows you to augment an inherited method in such a way that it can perform its inherited action and some new action.

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

Ans:-  if you assign a value to a name inside a function, then that name will have a local Python scope. In contrast, if you assign a value to a name outside of all functions—say, at the top level of a module—then that name will have a global Python scope