1. What is the concept of an abstract superclass?

A class is called an Abstract class if it contains one or more abstract methods. An abstract method is a method that is declared, but contains no implementation. Abstract classes may not be instantiated, and its abstract methods must be implemented by its subclasses.

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

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

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.

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

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.

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

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.