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

**Answer**: An abstract superclass is a 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). This is called the "template method" pattern (and this is not possible with an interface, which cannot provide final methods).

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

**Answer**:

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

**Answer**: 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?**

**Answer**: Each method that is inherited from a superclass can be augmented to perform some different action in the new class. If there is no augmentation then the inherited methods will perform as defined in the superclass. An inherited method is augmented simply by creating a new definition for the method in your class definition.

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

**Answer**: