1. What is the concept of an abstract superclass?

A) 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

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

A) If class has basic assignment statement then  the value of \_\_name\_\_ is set to contain the name of the module and will run the code from the assigned module.

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

A) 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) We can modify the existing method instead of completely replacing it.

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

A) Local (or function) scope is **the code block or body of any Python function or lambda**

**expression**. This Python scope contains the names that you define inside the function. These

names will only be visible from the code of the function.

Function scope: **Variables that are declared inside a function are called local variables and in**

**the function scope**. Local variables are accessible anywhere inside the function.