1. What is the concept of an abstract superclass?

Abstract superclasses are useful for creating modular and extensible code, as they allow you to define a common interface that can be shared by multiple subclasses. They also make it easier to reason about and test your code, as it can rely on the consistency of the interface defined by the 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?

When class statement's top level contains a basic assignment statement, it is considered as class attribute. Change in the value of class attribute will affect all the instances of the class.

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

By doing so,we can access those methods of the super-class (parent class) which have been overridden in a sub-class (child class) that inherits from it.

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

The way to do that in Python is by calling to the original version directly, with augmented arguments.

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

In class, if the variable is declared without self then it is accessible within that function only, kinda local variable. However if it declared using self like self.variable\_name = 'somevalue', then it is accessible via any object but not via the class name.

Whereas, if a variable is declared within a function then it is a local variable and is accessible to that function only.