1. What is the concept of an abstract superclass?

**Concrete Methods in Abstract Base Classes :**   
Concrete classes contain only concrete (normal)methods whereas abstract classes may contain both concrete methods and abstract methods. The concrete class provides an implementation of abstract methods, the abstract base class can also provide an implementation by invoking the methods via super().

# Python program invoking a

# method using super()

import abc

from abc import ABC, abstractmethod

class R(ABC):

def rk(self):

print("Abstract Base Class")

class K(R):

def rk(self):

super().rk()

print("subclass ")

# Driver code

r = K()

r.rk()

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?

For inheritance A child class needs to identify which class is its parent class. This can be done by mentioning the parent class name in the definition of the child class.

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

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

The local class is in the scope of the enclosing scope, and has the same access to names outside the function as does the enclosing function. Declarations in a local class can use only type names, static variables, extern variables and functions, and enumerators from the enclosing scope.