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?**

[**Class attributes**](https://www.geeksforgeeks.org/g-fact-34-class-or-static-variables-in-python/) belong to the class itself they will be shared by all the instances. Such attributes are defined in the class body parts usually at the top.  All object refer to single copy

class classattr:

    cnt = 0     # class attribute

def inc(self):

        classattr.cnt += 1

x = classattr()

x. inc()

print(x.cnt)

s1 = sampleclass()

s1.increase()

print(s1.count)

s2 = sampleclass()

s2.increase()

print(s2.count)

print(classattr.count)

1

2

2

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

If we need super's \_\_init\_\_ to be done in addition to what is being done in the current /child class's \_\_init\_\_, we must call it, since that will not happen automatically. But if we don't need anything from super's \_\_init\_\_, no need to call it.

class C(object):

def \_\_init\_\_(self):

self.b = 1

class D(C):

def \_\_init\_\_(self):

super().\_\_init\_\_() # in Python 2 use super(D, self).\_\_init\_\_()

self.a = 1

class E(C):

def \_\_init\_\_(self):

self.a = 1

d = D()

d.a

1

d.b # This works because of the call to super's init

1

e = E()

1

>>> e.b # This is going to fail since nothing in E initializes b...

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

To Redefine it in the subclass , but call back to the super class version of the method manually from the new version of teh method in the subclass that is pass the self instance to the super class version of the method manually. Superclass.method(self,..)

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

Functions declared inside class will be shared by all the objects. Instance Attributes are not shared by objects