1. What is the concept of an abstract superclass?

An abstract super class is defined to declare a set of methods that must be created in any subclass of this abstract class, this class can not instantiate.

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

It means we are defining a class attribute which can share in all objects and our class only has this line of code

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

This is not mandatory, but if we change some part of superclass which needs to run \_\_init\_\_ to call those changes, then because we do not define it manually the user will see an unknown error, because creating instance of subclass does not call super class init to initialize those changes.

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

We can call the inherited method with the name of the super class

class a:

def write (self):

print (“test a”)

class b(a):

def write(self):

print (“test b”)

a.write (self)

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

A local variable in function only can be use inside a function and all nested functions when we call that function, while a local variable in class can be share between all objects and if the value of it change, it will change for all of them.