1. What is the concept of an abstract superclass?

When we are designing classes, a superclass should contain the features which are shared by subclasses. Abstract super class is one way to provide re-usable code and we can extend the abstract class and inherit the class. Abstract class has the additional benefit that it does not have to provide a complete implementation (that would make sense to instantiate on its own), some parts can be left specified, but unimplemented (the abstract methods).

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

When the class statement’s top level has basic assignment statement it will become class variable.

Example:

class cls:

a=1 #class varaible

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

Base class typically create member variable and initialize them to defaults. So if we don't call base class init, none of that code would be executed and we will end up with base class that has no member variables.

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

This can be done by calling to the original version directly, with augmented arguments.

class cls(arg1):

def test1(self, a,b):

arg1.test1(self, a+b)

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

local scope of the class can be accessed anywhere in the class for any function but for the function it can be accessed with in that particular function