1. What is the concept of an abstract superclass?

Abstract Super Class. A common superclass for several subclasses. Factor up common behavior. Define the methods they all respond to. Methods that subclasses should implement are declared abstract.

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

An assignment statement evaluates the expression list (remember that this can ... Note: If the object is a class instance and the attribute reference occurs on both ... is not a top level module

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

Super(name) creates a new instance of Super which is immediately discarded. You need to call \_\_init\_\_() on the specific instance (self), super() is preferred and you no longer need to pass arguments to super(), e.g. super().\_\_init\_\_(name) is sufficient.

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

What we really want to do here is somehow augment the original giveRaise, instead of replacing it altogether. The good way to do that in Python is by calling to the original version directly, with augmented arguments, like this:

class Manager(Person):

def giveRaise(self, percent, bonus=.10):

Person.giveRaise(self, percent + bonus) # Good: augment original

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

Declaring a variable with self.(variable name) inside a function inside a class.

Declaring a variable inside a function inside a class: only that function can access it (it's in that function's scope)