1. What is the concept of an abstract superclass?

Ans:

In object-oriented programming, an abstract class is a [class](https://www.pythontutorial.net/python-oop/python-class/) that cannot be instantiated. However, you can create classes that inherit from an abstract class.

Typically, you use an abstract class to create a blueprint for other classes.

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

Ans:

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

class A(object):

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

print("world")

class B(A):

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

print("hello")

super().\_\_init\_\_()

When we want to use the augment on the existing behavious of the base class we have to call the \_\_init\_\_ method of the super class.

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

class Base(object):

def load(self):

print('do logic A')

print('do logic B')

class Child(Base):

def load(self):

super().load()

print('do logic C')

c = Child()

c.load()

This will print:

do logic A

do logic B

do logic C

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

Ans:

A class definition in particular creates a class scope, which is a special kind of scope. While the class scope is indeed enclosed by the global scope (class variables can access global variables), the class scopes DO NOT ENCLOSE the local scopes within it! So, the local method scopes within the class body cannot access the class scope during name lookup. (At least without qualifying it). In other words, a class's scope is only accessible from within the top level codebody of its own class definition. it is inaccessible from anywhere else, be it inside or outside, without using dot notation directly