1. What is the concept of an abstract superclass?

An abstract superclasss is a blueprint of a main class under which several sub classses or methods are defined. It allows you to create a set of methods that must be created within any child classes built from the abstract class. A class which contains one or more abstract methods is called an abstract class.

For example:

class animal(ABC):

def move(self):

class bird(animal):

def move(self):

print(“It can fly”)

class human(animal):

def move(self):

print(“it has 2 legs”)

class dog(animal):

def move(self):

print(“it has 4 legs”)

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

When a class statement yop level contains a basic assignment, it produces a class variable, that variable can be accessed by any instances within the class and by the class itself. For example:

class myclass :

class\_variable=”Hello World”

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

self.instance\_variable= “instance variable”

print(myclass.class\_variable)

---> “Hello World”

obj=myclass()

print(obj.class\_variable)

---🡪 “Hello World”

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

When a subclass overrides the \_\_init\_\_ method of its superclass, it replaces the constructor of the superclass with its own constructor. However, if the subclass does not call the superclass's \_\_init\_\_ method explicitly, then the superclass's constructor will not be executed. This means that the superclass's initialization logic will not be inherited and the subclass may not behave correctly. This allows the subclass to inherit and extend the behavior of the superclass \_\_init\_\_ method.

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

We can augment an inherited method by using super () function after the iniherited method. This will help to add additional behavior before or after the call to the superclass's method. For example:

class animal:

def first(self):

print(“The animal is roaming”)

class goat(animal):

def first(self):

print(“The animal is a goat”)

start().first()

print(“The animal fled”)

output:

The animal is a goat

The animal is roaming

The animal fled

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

The class's local scope is mainly used for defining and accessing the attributes and methods of the class, whereas the function's local scope is used for defining and accessing local variables within the function.