1. What is the concept of an abstract superclass?

Ans:- An abstract class is a class that is **declared abstract** —it may or may not include abstract methods. Abstract classes cannot be instantiated, but they can be subclassed. When an abstract class is subclassed, the subclass usually provides implementations for all of the abstract methods in its parent class.

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.

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

Ans:- An assignment statement **stores a value in a variable.** The assignment statement (or expression) is a fundamental construct.

We use Python assignment statements **to assign objects to names**. The target of an assignment statement is written on the left side of the equal sign (=), and the object on the right can be an arbitrary expression that computes an object. Assignment creates object references instead of copying the objects.

Assignment statements are used to (re)bind names to values and to modify attributes or items of mutable objects:

When a basic statement(x=y) appear at the top level of a class statements it attaches a data attribute to the class(class.x) like all class attributes this will be shared by all instance data attribute are not callable method function though.

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

Ans:- It's because one **needs to define something that is NOT done in the base-class**' \_\_init\_\_ , and the only possibility to obtain that is to put its execution in a derived-class' \_\_init\_\_ function.

This method is called when an object is created from a class and it **allows the class to initialize the attributes of the class**.

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

Ans:- **To augument instead of completely replacing an inherited method, redefine it in a subclass but call back to the superclass’s version of the method manually from the new version of the method in the subclass.** That is pass the self instance to the superclass version of the method manually

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

Ans:-A scope is **a region of the program** and broadly speaking there are three places, where variables can be declared: Inside a function or a block which is called local variables, In the definition of function parameters which is called formal parameters. Outside of all functions which is called global variables.

A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

x = 300

def myfunc():

x = 200

print(x)

myfunc()

print(x)

output:-200

300

The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.

class Person:  
  def \_\_init\_\_(self, name, age):  
    self.name = name  
    self.age = age  
  
  def myfunc(self):  
    print("Hello my name is " + self.name)  
  
p1 = Person("John", 36)  
p1.myfunc()

output:- Hello my name is john