**FSDS MAY BATCH 2022(Python Assignment -3)**

**Submitted by: Shubham Tiwari**

Q1: What is the concept of an abstract superclass?

Ans: It is a class where we can not create the objects directly ,its basic purpose is to define other classes i.e what methods and properties they are expected to have, in short a [class](https://www.scaler.com/topics/python/class-in-python/) containing one or more than one abstract method is called an abstract superclass.By default python does not come up with abstract classes ,infact it comes with a module that provides the base for defining Abstract Base classes(ABC) and that module name is ABC.ABC works by decorating methods of the base class as abstract and then registering concrete classes as implementations of the abstract base. A method becomes abstract when decorated with the keyword @abstractmethod.

For example:

from abc import ABC

class Passengers(ABC):

*#* **body of the class**

Q2: What happens when a class statement’s top level contains a basic assignment statement?

Ans:When a class statement top level contains a basic assignment statement(X=Y),it basically attaches a data attribute to (class, X ).Therefore like all other class attributes it will be shared by all instances and we also know that data attributes are not callable functions.

Q3: Why does a class need to manually call a superclass’s \_\_init\_\_ method?

Ans: The main reason for always calling superclass \_init\_\_ method is that base class may 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 would end up with base class that has no member variables.

Q4: How can you augment, instead of completely replacing, an inherited method?

Ans: We can easily augment by redefining it in a subclass, but call back to the superclass version of the method manually from the new version of the method in the subclass i.e pass the self instance to the superclass version of the method manually as shown:

Superclass.method(self,……..)

Q5: How is the local scope of a class different from that of a function?

Ans: A variable created inside a function belongs to the local scope of that function, and can only be used inside that function. For example, if we assign a value to a name inside a function, then that name will have a **local Python scope**. In contrast, if we assign a value to a name outside of all functions say, at the top level of a module then that name will have a **global Python scope**.