**ASSIGNMENT 3**

1. What is the concept of an abstract superclass?

Ans: Abstract superclass is a base class of many sub classes which contains various concrete or abstract methods. Abstract methods do nothing in superclass but gets overridden in subclasses and perform some specific functions. Abstract superclasses act as base class which tell us what kind of functions are present in class in any large program code. We can define abstract base classes by importing ABC (Abstract Base Classes) module.

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

Ans: It will be created as a class variable.

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

Ans: Whenever we are creating subclasses from a superclass and we need some variables or anything from superclass’s \_\_init\_\_method then we have to call it using the super() method. This is because if subclass have its own \_\_init\_\_method, then it will override the superclass’s method. Even if we don’t create \_\_init\_\_ method in subclass then also we have to call \_\_init\_\_method from superclass because it is not automatically called.

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

Ans: We can augment an inherited method by first calling them by super() method and then making the required changes. For example, if we have \_\_init\_\_method in superclass and we want to add more variables in the subclass \_\_init\_\_method but also want to have the same variables as superclass, then we can first create def \_\_init\_\_(self) then add super().\_\_init\_\_() and then add any new variables.

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

Ans: Local scope variables of a class can be accessed by class instances and its functions but variables created in a local scope of a function can only be accessed in the function.