**Lab 1**

**Queation#7**

1. **Class** — A blueprint created by a programmer for an object. This defines a set of attributes that will characterize any object that is instantiated from this class.
2. **Object** — An instance of a class. This is the realized version of the class, where the class is manifested in the program.
3. **Instance Variables** —Instance variables are owned by instances of the class. This means that for each object or instance of a class, the instance variables are different. Unlike class variables, instance variables are defined within methods.

**Examples of class, object and Instance Variables**

class car:

def \_\_init\_\_(self,company,color):

self.company = company

self.color = color

def display(self):

print ('This is a', self.color, self.company)

* Here **init** function work like a instance variable
* Class is **car**
* **Colour** is object
* When we create an object, the \_\_init\_\_ function inside the class is automatically invoked.

1. **Association** — Association is a type of relationship among more than one class. Use this when the object of one class contains a reference to one or more of another class.

**Types:**

* + - * 1. Association.
        2. Directed Association.
        3. Reflexive Association.
        4. Multiplicity.
        5. Aggregation.
        6. Composition.
        7. Inheritance/Generalization.
        8. Realization.

1. **Access Modifiers** — Python does not have access modifiers. If you want to access an instance (or class) variable from outside the instance or class, you are always allowed to do so.
   1. **Uses**— The single underscore prefix for a member variable or method is a commonly used convention to denote a private method. It doesn't actually change access privilege. So, for example, "\_privateMethod" can be accessed from inside the class or outside of it.