1. What is the concept of an abstract superclass?
Answer: A common super class for all subclasses that has all methods which the subclasses should re-use and re-define as per their requirement. The abstract superclass should have all the behaviour the sub classes should refer to

2. What happens when a class statement's top level contains a basic assignment statement?
Answer: A top level statement acts like a class variable which can be either private, protected or public. If it's protected it can only be called by the instance of the class or subclass or the class name with the value preceding it with an underscore _
Such as **ClassName._protectedVariableName**

Top level statement having a basic assignment generally acts as a public assignment whose value can be accessed by its subclasses, and also globally if the module is inherited

3. Why does a class need to manually call a superclass's __init__ method?
Answer: Call super(). __init__(args) within the child class to call the constructor of the immediate parent class with the arguments args . If a child class directly inherits from more than one parent class, the constructor of the first class in the list of parent classes will be called

```
class Building:
    def __init__(self,name):
        print("the building is", name)

class House(Building):
    def __init__(self,name):
        super().__init__(name)
        print("the {0} building is made".format(name))

house = House("new")
```

4. How can you augment, instead of completely replacing, an inherited method?
Answer: We can augment an inherited function by calling it directly, this way the original definition isn't replaced. In case the method is called through the class directly, an instance should be passed manually.

```
class Manager(Person):

def giveRaise(self, percent, bonus=.10):

Person.giveRaise(self, percent + bonus) # Good: augment original
```

.

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

Answer: A variable defined inside a class outside any function is a public variable for all the functions within that class, This is a local scope of a class. A variable defined inside a function inside a class is accessible only within that function if the variable is declared without self. This is local scope of a function

```
class Breathe:
   value = 6    #local scope of class

   def __init__(self, day):
      self.day = day

   def test(self,check):   ## check is local function variable
      check = check
      print("the check is {0}".format(check))
      print("the value uis {0}".format(self.value))


br = Breathe("monday")
br.test("ok")

the check is ok    >>> local function scope
the value is 6    >>> Local class scope
```