```python
class CSStudent:
    # Class Variable
    stream = 'cse'

    # The init method or constructor
    def __init__(self, roll):
        # Instance Variable
        self.roll = roll

    # Objects of CSStudent class


a = CSStudent(101)
b = CSStudent(102)

print(a.stream)  # prints "cse"
print(b.stream)  # prints "cse"
print(a.roll)  # prints 101

# Class variables can be accessed using class
# name also
print(CSStudent.stream)  # prints "cse"

##End of Programclass CSStudent:
    # Class Variable
    stream = 'cse'

    # The init method or constructor
    def __init__(self, roll):
        # Instance Variable
        self.roll = roll

        # Adds an instance variable

    def setAddress(self, address):
        self.address = address

        # Retrieves instance variable

    def getAddress(self):
        return self.address

    # Driver Code


a = CSStudent(101)
a.setAddress("Patna, Bihar")
print(a.getAddress())

##End of Program# Write Python code here
class sampleclass:
    count = 0    # class attribute

    def increase(self):
        sampleclass.count += 1

# Calling increase() on an object
s1 = sampleclass()
s1.increase()
print(s1.count)

# Calling increase on one more
# object
s2 = sampleclass()
s2.increase()
print(s2.count)

print(sampleclass.count)

##End of Program# A Python program to demonstrate inheritance

# Base or Super class. Note object in bracket.
# (Generally, object is made ancestor of all classes)
# In Python 3.x "class Person" is
# equivalent to "class Person(object)"
class Person(object):

    # Constructor
    def __init__(self, name):
        self.name = name

        # To get name

    def getName(self):
        return self.name

        # To check if this person is employee

    def isEmployee(self):
        print("I am in Person Class")
        return False


# Inherited or Sub class (Note Person in bracket)
class Employee(Person):

    # Here we return true
    def isEmployee(self):
        print("I am in Employee Class")
        return True


# Driver code
emp = Person("Geek1")  # An Object of Person
print(emp.getName(), emp.isEmployee())

```

```
emp = Employee("Geek2")  # An Object of Employee
print(emp.getName(), emp.isEmployee())

##End of Program#Super  Python code to demonstrate how parent constructors
# are called.

# parent class
class Person(object):

    # __init__ is known as the constructor
    def __init__(self, name, idnumber):
        self.name = name
        self.idnumber = idnumber

    def display(self):
        print(self.name)
        print(self.idnumber)

    # child class


class Employee(Person):
    def __init__(self, name, idnumber, salary, post):
        self.salary = salary
        self.post = post

        # invoking the __init__ of the parent class
        Person.__init__(self, name, idnumber)  #important to declare to get base class vars

    def display(self):
        Person.display(self)
        print(self.salary)
        print(self.post)



    # creation of an object variable or an instance


p1 = Person('Rahul', 886012)
# calling a function of the class Person using its instance
p1.display()


p2 = Employee('Ashish', 105, 5000, "VP")
# calling a function of the class Person using its instance
p2.display()


##End of Program# If you forget to invoke the __init__() of the parent class then its instance variables would not be available to the child class.
# The following code produces an error for the same reason.
```

```python
# Python program to demonstrate error if we
# forget to invoke __init__() of parent.

class A:
    def __init__(self, n='Rahul'):
        self.name = n


class B(A):
    def __init__(self, roll):
        self.roll = roll


object = B(23)
print(object.name) #AttributeError: 'B' object has no attribute 'name'


##End of Program# Python example to show working of multiple
# inheritance
class Base1(object):
    def __init__(self):
        self.str1 = "Geek1"
        print("Base1")


class Base2(object):
    def __init__(self):
        self.str2 = "Geek2"
        print("Base2")


class Derived(Base1, Base2):
    def __init__(self):
        # Calling constructors of Base1
        # and Base2 classes
        Base1.__init__(self)
        Base2.__init__(self)
        print("Derived")

    def printStrs(self):
        print((self.str1, self.str2))


ob = Derived()
ob.printStrs()

##End of Program#3. Multilevel inheritance: When we have child and grand child relationship.
# A Python program to demonstrate inheritance

# Base or Super class. Note object in bracket.
# (Generally, object is made ancestor of all classes)
# In Python 3.x "class Person" is
# equivalent to "class Person(object)"
class Base(object):

    # Constructor
    def __init__(self, name):
        self.name = name

    # To get name
    def getName(self):
        return self.name

    # Inherited or Sub class (Note Person in bracket)


class Child(Base):

    # Constructor
    def __init__(self, name, age):
        Base.__init__(self, name)
        self.age = age

    # To get name
    def getAge(self):
        return self.age

    # Inherited or Sub class (Note Person in bracket)


class GrandChild(Child):

    # Constructor
    def __init__(self, name, age, address):
        Child.__init__(self, name, age)
        self.address = address

    # To get address
    def getAddress(self):
        return self.address

    # Driver code


g = GrandChild("Geek1", 23, "Noida")
print((g.getName(), g.getAge(), g.getAddress()))


##End of Program# Python program to demonstrate private members
# of the parent class
class C(object):
       def __init__(self):
             self.c = 21

             # d is private instance variable
```

```
              self.__d = 42
class D(C):
       def __init__(self):
              self.e = 84
              C.__init__(self)
object1 = D()

# produces an error as d is private instance variable
print(D.d)

##End of Program
class Cricket_Personalities:
    sport_played = "Cricket"

    def __init__(self, fname, lname):
        print("I am called always on declaration of object")
        self.firstname = fname
        self.lastname = lname


    def printname(self):
        print(self.firstname, self.lastname)


# Use the Person class to create an object, and then execute the printname method:

x = Cricket_Personalities("Sachin", "Tendulkar")
x.printname()
print(x.sport_played)

y = Cricket_Personalities("Rahul", "Dravid")
y.printname()

##End of Program
```