**Write a Rectangle class in Python language, allowing you to build a rectangle with length and width attributes.Create a Perimeter() method to calculate the perimeter of the rectangle and a Area() method to calculate the area of the rectangle.Create a method display() that display the length, width, perimeter and area of an object created using an instantiation on rectangle class.Create a Parallelepipede child class inheriting from the Rectangle class and with a height attribute and another Volume() method to calculate the volume of the Parallelepiped. **

```
class Rectangle:
  def __init__(self):
    self.length=int(input("Enter length: "))
    self.width=int(input("Enter width: "))
  def Perimeter(self):
    return 2*(self.length+self.width)
  def Area(self):
    return (self.length*self.width)
  def display(self):
    print("Length=",self.length)
    print("Width=",self.width)
    print("Perimeter=",self.Perimeter())
    print("Area=",self.Area())
class Parallelepipede(Rectangle):
  def __init__(self):
    Rectangle.__init__(self)
    self.height=int(input("Enter Height: "))
  def Volume(self):
    return self.height*self.length*self.width
p1=Parallelepipede()
```

```
Enter length: 12
Enter width: 15
Enter Height: 32
```

```
p1.Perimeter()
```

```
60
```

```
p1=Parallelepipede()
```

```
Enter length10
Enter width20
Enter Height: 25
```

```
p1.Volume()
```

```
5000
```

**Create a Python class Person with attributes: name and age of type string.Create a display() method that displays the name and age of an object created via the Person class.Create a child class Student which inherits from the Person class and which also has a section attribute.Create a method displayStudent() that displays the name, age and section of an object created via the Student class.Create a student object via an instantiation on the Student class and then test the displayStudent method. **

```
class Person:
  def __init__(self):
    self.name=input("Enter Nmae:")
    self.age=(input("Enter Age:"))
  def Display(self):
    print("Name: ",self.name)
    print("Age: ",self.age)
class Student(Person):
  def __init__(self):
    Person.__init__(self)
    self.section=input("Enter Section: ")
  def displaystudent(self):
    print("Name of Student: ",self.name)
    print("Age of Student: ",self.age)
    print("Section of Student: ",self.section)
```

```
p1=Person()
```

```
Enter Nmae:parikshit
Enter Age:24
```

```
p1.Display()
```

```
Name:  parikshit
Age:  24
```

```
s1=Student()
```

```
Enter Nmae:parikshit
Enter Age:24
Enter Section: A
```

```
s1.displaystudent()
```

```
Name of Student:  parikshit
```

```
Age of Student:  24
Section of Student:  A
```

Create a Python class called BankAccount which represents a bank account, having as attributes: accountNumber (numeric type), name (name of the account owner as string type), balance.Create a constructor with parameters: accountNumber, name, balance.Create a Deposit() method which manages the deposit actions.Create a Withdrawal() method which manages withdrawals actions.Create an bankFees() method to apply the bank fees with a percentage of 5% of the balance account.Create a display() method to display account details.Give the complete code for the BankAccount class.

```
class BankAccount:
  def __init__(self,accno,name,balance):
    self.accno=accno
    self.name=name
    self.balance=balance
  def Deposit(self,deposit):
    self.balance=self.balance + deposit
  def Withdrawal(self,withdrawal):
    if self.balance < withdrawal:
      print("Withdrawl amount is More than Balance")
    else:
      self.balance=self.balance-withdrawal
  def bankFees(self):
    return print("Bank Fees: ",(5/100)*self.balance)
  def display(self):
    print("Account Number: ",self.accno)
    print("Account Holder Name: ",self.name)
    print("Account Balance: ",self.balance)
```

```
b1=BankAccount(1213,"parikshit",5000)
```

```
b1.Deposit(500)
```

```
b1.Withdrawal(600)
```

```
b1.display()
```

```
Account Number:  1213
Account Holder Name:  parikshit
Account Balance:  4900
```

```
b1.bankFees()
```

```
     Bank Fees:  245.0
```

Define a Book class with the following attributes: Title, Author (Full name), Price.Define a constructor used to initialize the attributes of the method with values entered by the user.Set the View() method to display information for the current book.Write a program to testing the Book class.

```
class Book:
  def __init__(self):
    self.Title=input("Enter Title: ")
    self.Author=input("Enter Author Full Name: ")
    self.Price=(input("Enter Price: "))
  def View(self):
    print("Title: ",self.Title)
    print("Author Name: ",self.Author)
    print("Price: ",self.Price)
b1=Book()
b1.View()
```

```
     Enter Title: python programming
     Enter Author Full Name: vinita mam
     Enter Price: 499
     Title:  python programming
     Author Name:  vinita mam
     Price:  499
```

** - Create a class called TK_extended which inherits from TK class and having the attributes:

- Master: that represents the name of the main window
- title: that represents the title of the main window 2 - Create a method called create() that creates the window 3 - Create a method called resize(width, height) that can resize the window. 4 - Create a method called generate() to generate the window **

```
ERROR: Could not find a version that satisfies the requirement tkinter (from versions: n
ERROR: No matching distribution found for tkinter
```

**Create a child class Bus that will inherit all of the variables and methods of the Vehicle class. In the vehicle class create relevant methods and variables.

2. Define a property that must have the same value for every class instance (object). Define a class attribute "color” with a default value white. I.e., Every Vehicle should be white.**

```
class Vehicle:
    def __init__(self, name, mileage, capacity):
        self.name = name
        self.mileage = mileage
        self.capacity = capacity

    def fare(self):
        return self.capacity * 100

class Bus(Vehicle):
  def __init__(self,name, mileage, capacity,color="White"):
    Vehicle.__init__(self, name, mileage, capacity)
    self.color=color
  def info(self):
    print("Vehicle name: ",self.name,"\nMileage: ",self.mileage,"\nCapacity: ",self.capacity,
b1= Bus("tata tampo", 12, 50)
b1.info()
```

```
 Vehicle name:  tata tampo
 Mileage:  12
 Capacity:  50
 Fare:  5000
 Color:  White
```