All the students should do the question no. I I. Develop a set of classes for a college to use in various student service and personnel applications. Classes you need to design include the following: • Person—A Person contains a first name, last name, street address, zip code, and phone number. The class also includes a method that sets each data field, using a series of dialog boxes and a display method that displays all of a Person's information on a single line at the command line on the screen. • CollegeEmployee—CollegeEmployee descends from Person. A CollegeEmployee also includes a Social Security number, an annual salary, and a department name, as well as methods that override the Person methods to accept and display all CollegeEmployee data. • Faculty—Faculty descends from CollegeEmployee. This class also includes a Boolean field that indicates whether the Faculty member is tenured, as well as methods that override the CollegeEmployee methods to accept and display this additional piece of information. • Student—Student descends from Person. In addition to the fields available in Person, a Student contains a major field of study and a grade point average as well as methods that override the Person methods to accept and display these additional facts. Write an application named CollegeList that declares an array of four "regular" CollegeEmployees, three Faculty, and seven Students. Prompt the user to specify which type of person's data will be entered (C, F, or S), or allow the user to quit (Q). While the user chooses to continue (that is, does not quit), accept data entry for the appropriate type of Person. If the user attempts to enter data for more than four CollegeEmployees, three Faculty, or seven Students, display an error message. When the user quits, display a report on the screen listing each group of Persons under the appropriate heading of "College Employees," "Faculty," or "Students." If the user has not entered data for one or more types of Persons during a session, display an appropriate message under the appropriate heading

```
class Person:
    def __init__(self, first_name, last_name, street_address, zip_code, phone_nu
        self.first_name = first_name
        self.last_name = last_name
        self.street_address = street_address
        self.zip_code = zip_code
        self.phone_number = phone_number

    def set_data(self):
        self.first_name = input("Enter first name: ")
        self.last_name = input("Enter last name: ")
        self.street_address = input("Enter street address: ")
        self.zip_code = input("Enter zip code: ")
        self.phone_number = input("Enter phone number: ")
```

```
    def display(self):
        print(f"{self.first_name} {self.last_name} ({self.phone_number})")

class CollegeEmployee(Person):
    def __init__(self, first_name, last_name, street_address, zip_code, phone_nu
        super().__init__(first_name, last_name, street_address, zip_code, phone_
        self.ssn = ssn
        self.salary = salary
        self.department_name = department_name

    def set_data(self):
        super().set_data()
        self.ssn = input("Enter SSN: ")
        self.salary = input("Enter annual salary: ")
        self.department_name = input("Enter department name: ")

    def display(self):
        super().display()
        print(f"SSN: {self.ssn}, Salary: {self.salary}, Department: {self.depart

class Faculty(CollegeEmployee):
    def __init__(self, first_name, last_name, street_address, zip_code, phone_nu
        super().__init__(first_name, last_name, street_address, zip_code, phone_
        self.tenured = tenured

    def set_data(self):
        super().set_data()
        self.tenured = input("Is the faculty member tenured? (yes/no): ").lower(

    def display(self):
        super().display()
        print(f"Tenured: {self.tenured}")

class Student(Person):
    def __init__(self, first_name, last_name, street_address, zip_code, phone_nu
        super().__init__(first_name, last_name, street_address, zip_code, phone_
        self.major = major
        self.gpa = gpa

    def set_data(self):
        super().set_data()
        self.major = input("Enter major field of study: ")
        self.gpa = input("Enter GPA: ")

    def display(self):
        super().display()
        print(f"Major: {self.major}, GPA: {self.gpa}")
```

```
class CollegeList:
    def __init__(self):
        self.college_employees = []
        self.faculty_members = []
        self.students = []

    def run(self):
        while True:
            choice = input("Enter C for CollegeEmployee, F for Faculty, S for St
            if choice == "C":
                if len(self.college_employees) < 4:
                    employee = CollegeEmployee("", "", "", "", "", "", "", "")
                    employee.set_data()
                    self.college_employees.append(employee)
                else:
                    print("Error: Cannot enter more than 4 CollegeEmployees.")
            elif choice == "F":
                if len(self.faculty_members) < 3:
                    faculty = Faculty("", "", "", "", "", "", "", "", "")
```

1. Write a Python program to show that the variables with a value assigned in class declaration, are class variables and variables inside methods and constructors are instance variables.

```
class MyClass:
    class_variable = 10

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

    def set_variable(self, new_value):
        self.new_value = new_value

my_object1 = MyClass(20)
my_object2 = MyClass(30)

print("my_object1.class_variable =", my_object1.class_variable)
print("my_object2.class_variable =", my_object2.class_variable)
print("my_object1.instance_variable =", my_object1.instance_variable)
print("my_object2.instance_variable =", my_object2.instance_variable)

my_object1.set_variable(40)
print("my_object1.new_value =", my_object1.new_value)
```

2. Write a Python program to show that we can create instance variables inside methods

```
class MyClass:
    def __init__(self, initial_value):
        self.initial_value = initial_value

    def set_variable(self, new_value):
        self.new_value = new_value

my_object = MyClass(10)
my_object.set_variable(20)

print(my_object.initial_value)
print(my_object.new_value)
```

3. Write a python code to implement following scenario, In a company Factory, staff and Office staff have certain common properties - all have a name, designation, age etc. Thus they can be grouped under a class called CompanyMember. Apart from sharing those common features, each subclass has its own characteristic - FactoryStaff gets overtime allowance while OfficeStaff gets traveling allowance for an office job. The derived classes (FactoryStaff & OfficeStaff) has its own characteristic and, in addition, they inherit the properties of the base class (CompanyMember).

```
class MyClass:
    class_variable = 10

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

    def set_variable(self, new_value):
        self.new_value = new_value

my_object1 = MyClass(20)
my_object2 = MyClass(30)

print("my_object1.class_variable =", my_object1.class_variable)
print("my_object2.class_variable =", my_object2.class_variable)
print("my_object1.instance_variable =", my_object1.instance_variable)
print("my_object2.instance_variable =", my_object2.instance_variable)

my_object1.set_variable(40)
print("my_object1.new_value =", my_object1.new_value)
```

4. Write a python program to access hidden variable outside the class using object and which threw an exception.

```
class MyClass:
    __hidden_var = 10

my_object = MyClass()
print(my_object.__hidden_var)
```

5. Write a Python program to demonstrate that hidden members can be accessed outside a class

```
class MyClass:

    # Hidden member of MyClass
    __hiddenVariable = 0

    # A member method that changes
    # __hiddenVariable
    def add(self, increment):
        self.__hiddenVariable += increment
        print (self.__hiddenVariable)

# Driver code
myObject = MyClass()
myObject.add(2)
myObject.add(5)

# This line causes error
print (myObject.__hiddenVariable)
```