Week 5

1) **Design a Library Management System using classes and objects to manage books across different branches.**

   a. Create a class Library with the following:
      - A class attribute library_name (same for all branches).
      - A class attribute total_books to track the total number of books across all branches.
      - Instance attributes: branch_name and books (list of books in that branch).

   b. Implement methods:
      - add_book(book_title) → Adds a book to the branch and updates the total book count.
      - display_books() → Displays all books in that branch.
      - display_total_books() → A class method that shows the total number of books across all branches.

   c. Create two or more library branches and demonstrate adding books, displaying books per branch, and showing the total book count.

2) **Develop a Student Result System using inheritance to manage students, their subjects, and final results.**
   a. Create a base class Person with attributes name and roll_no.
   b. Create a derived class Student (inherits from Person) with additional attributes subject_marks (dictionary).
   c. Create another derived class Result (inherits from Student) that:
      i. Calculates total marks and percentage.
      ii. Displays pass/fail status (pass if percentage ≥ 40).
   d. Do the following:
      i. Create at least three students.
      ii. Enter marks for 5 subjects per student.
      iii. Display their total marks, percentage, and result status.

3) **Create an Employee Payroll System to manage employee salaries using multiple inheritance.**
   a. Create a class Employee with attributes emp_id and emp_name.
   b. Create a class Salary with attributes basic, hra, and da, and a method calculate_salary() to compute gross salary.
   c. Create a class Payroll that inherits from both Employee and Salary and includes:
      i. A method display_details() to show the employee details and total salary.
   d. Create objects for at least three employees, input their salary details, and display their payroll.
   e. Demonstrates multiple inheritance where one class inherits from two parent classes.

Week-6:

1)Design a University Course Registration System using classes, objects, class attributes, and hybrid inheritance. The system should manage students, courses, faculties, and registrations.

A. Classes to Create

1. Person *(Base Class)*
   - o Attributes: name, email
2. Student *(Inherits from Person)*
   - o Attributes: student_id, registered_courses (list)
   - o Method: register_course(course) → Registers a student for a course if available.
3. Faculty *(Inherits from Person)*
   - o Attributes: faculty_id, assigned_courses (list)
   - o Method: assign_course(course) → Assigns a course to the faculty.
4. Course *(Independent Class)*
   - o Attributes: course_code, course_name, capacity, registered_students (list).
   - o Method: add_student(student) → Registers a student only if capacity allows.
5. University *(Main Controller Class)*
   - o Class attribute: total_students → Keeps track of total students registered across all courses.
   - o Attributes: all_courses (dictionary), all_students (dictionary).
   - o Methods:
     - ▪ add_course(course)
     - ▪ add_student(student)
     - ▪ display_all_courses()
     - ▪ display_all_students()

B. Additional Conditions

- A course cannot have more students than its capacity.
- If a student registers for a course successfully, update both the student's list and the course's registered list.
- Maintain a global counter for the total number of students registered in the university.
- Demonstrate the system by creating at least 5 students, 3 faculties, and 4 courses.

2) **Develop a Hospital Patient Management System that manages doctors, patients, and appointments.**

A. Classes to Create

1. Person *(Base Class)*
   - o Attributes: name, age, gender.
2. Doctor *(Inherits from Person)*
   - o Attributes: doctor_id, specialization, available_slots (list of available timings).
   - o Method: book_slot(slot) → Books a slot if available.
3. Patient *(Inherits from Person)*
   - o Attributes: patient_id, disease, assigned_doctor, appointment_slot.
4. Appointment *(Aggregation Class — Uses Doctor & Patient Objects)*
   - o Attributes: appointment_id, doctor, patient, slot.
   - o Method: confirm_appointment() → Confirms appointment if the doctor's slot is available.

5. Hospital *(Main Controller Class)*
    - Attributes: all_doctors, all_patients, all_appointments.
    - Methods:
        - add_doctor(doctor)
        - add_patient(patient)
        - book_appointment(patient_id, doctor_id, slot)
        - display_all_appointments()

B. Additional Conditions

- No two patients can book the same doctor slot.
- If a slot is already booked, raise an error message.
- Maintain a class attribute total_appointments in Hospital to track how many appointments have been made.
- Demonstrate the system using at least 3 doctors, 6 patients, and 5 appointment bookings.