Python Mid Term Assignment

What you need to submit:
Create a public github repository, upload your code there and Provide its public link as assignment submission or you can submit google docs public link.

Task	Details	Marks
1. Create the StudentDatabase class	Define a class named `StudentDatabase` with one class attribute named `student_list`. Initially, it should be an empty list. Create a class method `add_student()` to insert an object of the `Student` class into `student_list`.	10
2. Create the Student class	Define a class `Student` with the following instance attributes: - student_id: Unique identifier for the student. - name: Full name of the student. - department: The department of the student. - is_enrolled: A boolean indicating if the student is currently enrolled.	10
3. Initialize Student Object	In the constructor of the `Student` class, initialize the attributes. Insert the `Student` object into `student_list` using the method `add_student()`. This part will be done manually, i.e., no need for a menu option.	10
4. Implement enroll_student() method	Add a method `enroll_student()` in the `Student` class that checks if the student is not already enrolled. If not, change `is_enrolled` to `True`.	10
5. Implement drop_student() method	Add a method `drop_student()` in the `Student` class that changes `is_enrolled` to `False` to indicate the student has dropped out.	10
6. Implement view_student_info() method	Add a method `view_student_info()` in the `Student` class to display the details of the student including `student_id`, `name`, `department`, and enrollment status.	15
7. Menu System	Create a menu-driven system with the following	15

	options: 1. View All Students 2. Enroll Student 3. Drop Student 4. Exit Handle the user's choice using input prompts.	
8. Error Handling	Implement error handling for: - Invalid student ID when enrolling or dropping a student. - Trying to enroll a student who is already enrolled. - Trying to drop a student who is not enrolled.	10
9. Data Privacy	Make the attributes (such as `student_id`, `name`, `department`, `is_enrolled`) as protected/private as possible using Python's class mechanisms. This will ensure that these attributes cannot be accessed directly outside the class.	10