Assignment

-----------------

1. Create a program that prompts the user for a number. If the number is divisible by 3, print "The number is divisible by 3." If it is divisible by 5, print "The number is divisible by 5." If it is divisible by both, print "The number is divisible by both

2. Write a program that asks the user for the original price of an item. If the price is greater than $100, apply a 20% discount and print the discounted price. Otherwise, print "No discount applied

3. Write a program that prompts the user for their age. If the age is 18 or older, print "You are eligible to vote." If it is under 18, print "You are not eligible to vote

Question 4:

Write a Python program that functions as a basic calculator. The program should do the following:

Prompt the user to enter four numbers (operands).

Perform the following operations using these four numbers:

- Addition

- Subtraction

- Multiplication

- Division (handle division by zero appropriately)

Display the results of each operation.

Requirements:

Use input() to get user input.

Ensure that the program handles invalid input (e.g., non-numeric values).

Print the results.

Assignment

Write a program that scans a list and prints the first negative number it finds. If there is none, print "No negative numbers found".

Write a program that separates the numbers in a list into two new lists: one with even numbers and one with odd numbers.

Write a program that takes a list of words and prints only the words with length greater than 5.

Write a program that:

Adds items to a list

Removes items

Shows all items

Quits

Use a loop and if statements to build a simple text-based menu.

Lumba Assignment ()

Function to Remove Duplicates from a List

- Write a function remove\_duplicates that takes a list as a parameter and prints a new list with duplicates removed.

Function to Calculate Average

- Create a function calculate\_average that takes a list of numbers as a parameter and prints the average of those numbers.

Function to Create a Simple Calculator

- Create a function simple\_calculator that takes two numbers and an operator (+, -, \*, /) as parameters and prints the result of the operation.

Assignment

Hotel Management System

You are required to create a Python program that simulates a simple hotel management system.

The program should use standalone functions to perform the following tasks.

Create Room

Function Name: create\_room(room\_number, price, is\_booked=False)

Parameters:

room\_number → The number of the room.

price → The price of the room.

is\_booked → Boolean value (True or False) indicating if the room is booked. Default is False.

Description: Adds a new room to the list of rooms.

Return Value: The updated list of rooms.

Book Room

Function Name: book\_room(rooms, room\_number)

Parameters:

rooms → List of room dictionaries.

room\_number → The room number to be booked.

Description: Changes the booking status of a room to True if it is available.

Return Value: A message indicating whether the booking was successful or not.

Check Room Availability

Function Name: check\_availability(rooms, room\_number)

Parameters:

rooms → List of room dictionaries.

room\_number → The room number to check.

Description: Checks if a specific room is available for booking.

Return Value: True if available, False if not.

Show All Rooms

Function Name: show\_all\_rooms(rooms)

Parameters:

rooms → List of room dictionaries.

Description: Displays all rooms with their details (room number, price, booking status).

Return Value: The list of all rooms.

Remove Room

Function Name: remove\_room(rooms, room\_number)

Parameters:

rooms → List of room dictionaries.

room\_number → The room number to remove.

Description: Deletes a specific room from the list.

Return Value: A message indicating whether the room was removed successfully or not.

Program Requirements

Use a list of dictionaries to store rooms.

Each dictionary should have:

"room\_number"

"price"

"is\_booked"

Task Management System

You are tasked with creating a Task Management System to help users manage their tasks efficiently. Define the expected behavior for the following functions:

Add Task

Function Name: add\_task

Parameters :

task\_name (string): The name of the task.

due\_date (string): The due date for the task (e.g., "2025-08-16").

priority (string): The priority level of the task (e.g., "High", "Medium", "Low").

Output :

Returns a success message if the task is added successfully.

If a task with the same name already exists, return an error message.

Remove Task

Function Name: remove\_task

Parameters :

task\_id (string): The ID of the task to remove.

Output :

Returns a success message if the task is removed successfully.

If the task does not exist, return an error message.

Update Task

Function Name: update\_task

Parameters :

task\_id (string): The ID of the task to update.

new\_task\_name (string): The new name for the task.

new\_due\_date (string): The new due date for the task.

new\_priority (string): The new priority level for the task.

Output :

Returns a success message if the task is updated successfully.

If the task ID does not exist, return an error message.

List All Tasks

Function Name: list\_all\_tasks

Parameters : None

Output :

Returns a list of all tasks, including their names, IDs, due dates, and priorities.

Search Task by Name

Function Name: search\_task\_by\_name

Parameters :

task\_name (string): The name of the task to search for.

Output :

Returns the details of the task if found (including ID, due date, and priority).

If no task matches the name, return a message indicating that the task was not found.

Instructions :

Implement each function to manage tasks effectively.

Ensure that all functions handle edge cases (e.g., invalid value, task not found).

Provide appropriate user feedback for all operations

Create a program that prompts the user for a number. If the number is divisible by 3, print "The number is divisible by 3." If it is divisible by 5, print "The number is divisible by 5." If it is divisible by both, print "The number is divisible by both

Write a program that:

Adds items to a list

Removes items

Shows all items

Use a loop and if statements to build a simple text-based menu.

Create two separate Python programs using Object-Oriented Programming (OOP) concepts. Each program should be well-structured with classes, methods, and proper use of \_\_init\_\_ and self

Simple Contact Book

Objective:

Create a contact book to store and manage names and phone numbers.

Requirements:

Create a class called ContactBook.

It should have the following methods:

- add\_contact(name, phone) – Adds a new contact.

- remove\_contact(name) – Removes a contact by name.

- view\_contacts() – Displays all contacts.

Use a dictionary inside the class to store contacts (name as key, phone as value).

Example Usage:

book = ContactBook()

book.add\_contact("Alice", "123-456-7890")

book.add\_contact("Bob", "987-654-3210")

book.view\_contacts()

book.remove\_contact("Alice")

book.view\_contacts()

2. Simple Counter

\*Objective:\*

Create a counter that can be increased, decreased, reset, and displayed.

\*Requirements:\*

- Create a class called `Counter`.

- The counter starts at 0.

- It should have the following methods:

- `increment()` – Increases the count by 1.

- `decrement()` – Decreases the count by 1.

- `reset()` – Sets the count back to 0.

- `show()` – Prints and returns the current count.

\*Example Usage:\*

counter = Counter()

counter.increment()

counter.increment()

counter.show() # Output: 2

counter.decrement()

counter.show() # Output: 1

counter.reset()

counter.show() # Output: 0

**75 million Assignment**

**Design a simple Library Management System using Object-Oriented Programming.**

**Create a Book class with:**

**Attributes: title, author, isbn, available\_copies.**

**Method: display\_info() → prints book details.**

**Create a Member class with:**

**Attributes: member\_id, name, borrowed\_books (list of borrowed book titles).**

**Methods:**

**borrow\_book(book) → decreases the book’s available copies by 1 and adds the book title to borrowed\_books. If no copies are available, print "Book not available".**

**return\_book(book) → increases the book’s available copies by 1 and removes the book title from borrowed\_books.**

**Create a Library class with:**

**Attribute: books (a list of Book objects).**

**Methods:**

**add\_book(book) → adds a new book to the library.**

**search\_book(title) → searches for a book by title and returns it if found.**

**display\_books() → prints all available books and their details.**

**In the main program:**

**Create at least three books and add them to the library.**

**Create two members.**

**Allow members to borrow and return books.**

**Display library inventory and member-borrowed books after each operation.**

**2:29 PM**

**Forwarded**

**Thank you for today assignment.**

**Simple School Management System (OOP)**

**Objective:**

**Create a system where a School can manage Students and Courses, and students can enroll in courses.**

**Requirements**

**1. Student Class**

**Attributes:**

**name (string)**

**student\_id (string or int — must be unique)**

**Methods:**

**\_\_str\_\_() → e.g., "Alice (ID: S001)"**

**2. Course Class**

**Attributes:**

**course\_name (string — e.g., “Math 101”)**

**course\_code (string — e.g., “MATH101”)**

**students (list — stores Student objects enrolled in this course)**

**max\_capacity (int — default: 10)**

**Methods:**

**add\_student(student) → adds student to students list if capacity allows**

**get\_student\_count() → returns number of enrolled students**

**is\_full() → returns True if course is full**

**list\_students() → prints all enrolled student names and IDs**

**\_\_str\_\_() → e.g., "Math 101 (MATH101) - Enrolled: 3/10"**

**3. School Class**

**Attributes:**

**name (string — e.g., “Greenwood High”)**

**students (list of Student objects)**

**courses (list of Course objects)**

**Methods:**

**add\_student(student) → adds to school’s student list**

**add\_course(course) → adds to school’s course list**

**enroll\_student\_in\_course(student\_id, course\_code) →**

**- Finds the student and course**

**- If both exist and course is not full → enrolls student**

**show\_all\_courses() → prints all courses and current enrollment**

**find\_student\_by\_id(student\_id) → returns Student object or None**

**find\_course\_by\_code(course\_code) → returns Course object or None**

**New here: dunder or magic method -> \_\_str\_\_()**

**2:29 PM**

**Forwarded**

**Thank you for today assignment.**

**Simple School Management System (OOP)**

**Objective:**

**Create a system where a School can manage Students and Courses, and students can enroll in courses.**

**Requirements**

**1. Student Class**

**Attributes:**

**name (string)**

**student\_id (string or int — must be unique)**

**Methods:**

**\_\_str\_\_() → e.g., "Alice (ID: S001)"**

**2. Course Class**

**Attributes:**

**course\_name (string — e.g., “Math 101”)**

**course\_code (string — e.g., “MATH101”)**

**students (list — stores Student objects enrolled in this course)**

**max\_capacity (int — default: 10)**

**Methods:**

**add\_student(student) → adds student to students list if capacity allows**

**get\_student\_count() → returns number of enrolled students**

**is\_full() → returns True if course is full**

**list\_students() → prints all en**

**…Read more**

**2:29 PM**

**Forwarded**

**Birthday Surprise Program Assignment**

**Objective:**

**Write a Python program that provides either a joke or a birthday wish based on user input.**

**Instructions:**

**Create a Jokes Class:**

**- This class should have a list of jokes.**

**- Include a method to return a joke based on an index.**

**Create a BirthdayWishes Class:**

**- This class should have a list of birthday wishes.**

**- Include a method to return a wish based on an index.**

**Create a BirthdaySurprise Class:**

**- This class should combine the Jokes and BirthdayWishes classes.**

**- Include a method that takes user input to either return a joke or a birthday wish.**

**User Interaction:**

**- In the main part of your program, ask the user if they want a "joke" or a "wish."**

**ask user to select within a range (index) of the "joke" or a "wish"**

**- Based on their inputs, call the appropriate method from the BirthdaySurprise class and display the result.**

**Example Output:**

**- If the user chooses "joke," display a random joke.**

**- If the user chooses "wish," display a random birthday wish.**

**Tips:**

**You do not need to use any imports for this assignment.**

**You can start with a few jokes and wishes, and feel free to add more if you like!**

**Good luck, and have fun coding!**

**2:29 PM**

**Forwarded**

**Thank you for today assignment.**

**Simple School Management System (OOP)**

**Objective:**

**Create a system where a School can manage Students and Courses, and students can enroll in courses.**

**Requirements**

**1. Student Class**

**Attributes:**

**name (string)**

**student\_id (string or int — must be unique)**

**Methods:**

**\_\_str\_\_() → e.g., "Alice (ID: S001)"**

**2. Course Class**

**Attributes:**

**course\_name (string — e.g., “Math 101”)**

**course\_code (string — e.g., “MATH101”)**

**students (list — stores Student objects enrolled in this course)**

**max\_capacity (int — default: 10)**

**Methods:**

**add\_student(student) → adds student to students list if capacity allows**

**get\_student\_count() → returns number of enrolled students**

**is\_full() → returns True if course is full**

**list\_students() → prints all enrolled student names and IDs**

**\_\_str\_\_() → e.g., "Math 101 (MATH101) - Enrolled: 3/10"**

**3. School Class**

**Attributes:**

**name (string — e.g., “Greenwood High”)**

**students (list of Student objects)**

**courses (list of Course objects)**

**Methods:**

**add\_student(student) → adds to school’s student list**

**add\_course(course) → adds to school’s course list**

**enroll\_student\_in\_course(student\_id, course\_code) →**

**- Finds the student and course**

**- If both exist and course is not full → enrolls student**

**show\_all\_courses() → prints all courses and current enrollment**

**find\_student\_by\_id(student\_id) → returns Student object or None**

**find\_course\_by\_code(course\_code) → returns Course object or None**

**New here: dunder or magic method -> \_\_str\_\_()**

**2:29 PM**

**Forwarded**

**Question 1:**

**Hospital Patient Management System\***

**Design a Hospital Patient Management System.**

**Create a Patient class with attributes: patient\_id, name, age, and disease.**

**Create a Doctor class with attributes: doctor\_id, name, specialization.**

**Create a Hospital class that stores both doctors and patients, with methods:**

**-admit\_patient(patient)**

**-assign\_doctor(patient\_id, doctor\_id)**

**-discharge\_patient(patient\_id)**

**-show\_patients() – List admitted patients and their assigned doctors.**

**Task: Write a program that simulates admitting patients, assigning doctors, and discharging them.**

**Question 2:**

**Online Shopping Cart**

**Imagine you are building a small E-commerce Shopping Cart System.**

**1.Create a Product class with attributes: name, price, and quantity.**

**2.Create a Cart class that allows:**

**-add\_product(product, quantity) – Add a product to the cart.**

**-remove\_product(product) – Remove a product from the cart.**

**-calculate\_total() – Calculate the total price of products in the cart.**

**3.Create a Customer class with attributes like name and a Cart object.**

**Task: Write a program that allows a customer to add multiple products to their cart, remove some, and display the final bill.**

**Question 3:**

**Airline Reservation System**

**Design a simple Airline Reservation System.**

**1.Create a Flight class with attributes: flight\_number, destination, capacity, and booked\_seats, with methods.**

**-book\_seat() – Book a seat if available.**

**-cancel\_seat() – Cancel a booked seat.**

**-available\_seats() – Show remaining seats.**

**2.Create a Passenger class with attributes: name, passport\_number, and a method book\_flight(flight).**

**3.Create an Airline class to manage flights.**

**Task: Write a program that creates flights, books passengers on them, and displays flight availabilit**