AI Lab Assignment 6.1 Roll No – 2403A51333

Task Description #1

Use AI to create an Employee class with attributes (name, id, salary) and a method to calculate yearly salary.

Code:

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A screenshot of a computer

AI-generated content may be incorrect.

Observation :

The Employee class is well-designed with clear methods for salary calculation and bonuses.  
Dynamic input makes it flexible, and formatted output improves readability.  
However, the bonus is added to monthly salary, making it recurring instead of one-time.  
The code lacks input validation and handles only one employee per run.  
Overall, it is functional and clean but can be improved for scalability and flexibility.

Task Description -2:

Task: Prompt AI to generate a function that displays all Automorphic numbers between 1 and 1000 using a for loop.

code to list Automorphic numbers using a for loop :

A computer screen with colorful text

AI-generated content may be incorrect.Output :

A black and white text

AI-generated content may be incorrect.

While Loop:

A computer screen with text

AI-generated content may be incorrect.

Output :

A screenshot of a computer

AI-generated content may be incorrect.

**Comparison** :

The for loop version of the Automorphic number program is more concise and easier to read because it automatically handles initialization, iteration, and stopping conditions when looping from 1 to 1000. It is ideal when the number of iterations is known in advance. On the other hand, the while loop version offers more flexibility since it can run until a condition is met, making it useful in cases where the endpoint is not predetermined. However, it requires manual initialization and incrementing of the loop variable, which increases the chance of errors such as infinite loops. Overall, the for loop is the better choice for this specific task, while the while loop is more suitable for dynamic or condition-based looping.

Task Description -3 :

Task: Ask AI to write nested if-elif-else conditions to classify online shopping feedback as Positive, Neutral, or Negative based on a numerical rating (1–5).

Code using if-elif-else:

A screen shot of a computer program

AI-generated content may be incorrect.

Output :

A black background with white text

AI-generated content may be incorrect.

Dictionary-based :

A screen shot of a computer program

AI-generated content may be incorrect.

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

Observation :

The dictionary-based approach to classifying feedback is more concise and efficient compared to nested if-elif-else statements. It directly maps ratings to their corresponding feedback messages, reducing code complexity and improving readability. The use of the get() method also provides a clean way to handle invalid inputs by returning a default message without extra conditions. This design makes the code easier to maintain and extend, for example, if more rating categories or messages need to be added in the future. Overall, it is a cleaner, more scalable, and pythonic solution for handling rating-based feedback classification.

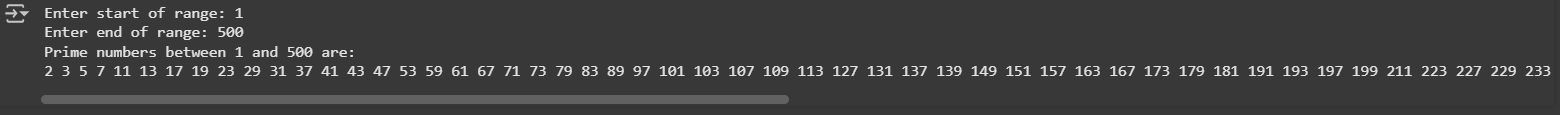
Task Description -4 :

Task: Generate a function using AI that displays all prime numbers within a user-specified range (e.g., 1 to 500).

A computer screen shot of a program code

AI-generated content may be incorrect.

Output:



Optimized code:

A computer screen shot of a program code

AI-generated content may be incorrect.

Output:

A screenshot of a computer

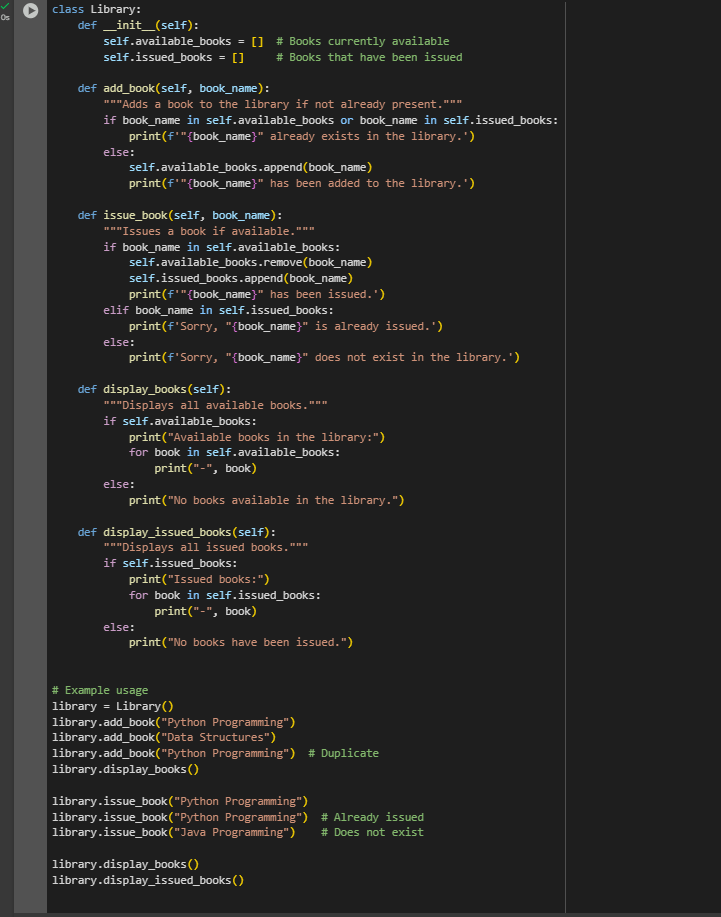
AI-generated content may be incorrect.

Observation :

The function efficiently generates prime numbers within a user-specified range by checking divisibility only up to the square root of each number. This reduces unnecessary computations compared to checking all numbers up to num, making it faster for larger ranges. It correctly handles edge cases, ignoring numbers less than 2, which are not prime. The implementation is simple, readable, and easy to understand, using a straightforward for loop and a boolean flag to track primality. Overall, this approach balances clarity and efficiency, making it suitable for moderate ranges like 1–500 or even higher.

Task Description -5 :

Task: Use AI to build a Library class with methods to add\_book(), issue\_book(), and display\_books().



Output:

A screenshot of a computer program

AI-generated content may be incorrect.

After adding comments and documentation:

A screen shot of a computer screen

AI-generated content may be incorrect.

A computer screen with text

AI-generated content may be incorrect.

Output:

A computer screen with white text

AI-generated content may be incorrect.

Observation:

The Library class is well-structured and fully documented with clear docstrings and inline comments, making it easy to understand and maintain. It effectively handles key edge cases, such as preventing duplicate book entries, managing requests for already issued books, and handling non-existent books gracefully. The class maintains separate lists for available and issued books, allowing clear tracking of the library’s inventory. Its methods provide straightforward functionality to add, issue, and display books, and the inclusion of descriptive messages improves user interaction. Overall, the design is robust, scalable, and user-friendly, providing a solid foundation for extending features like returning books or managing multiple users.