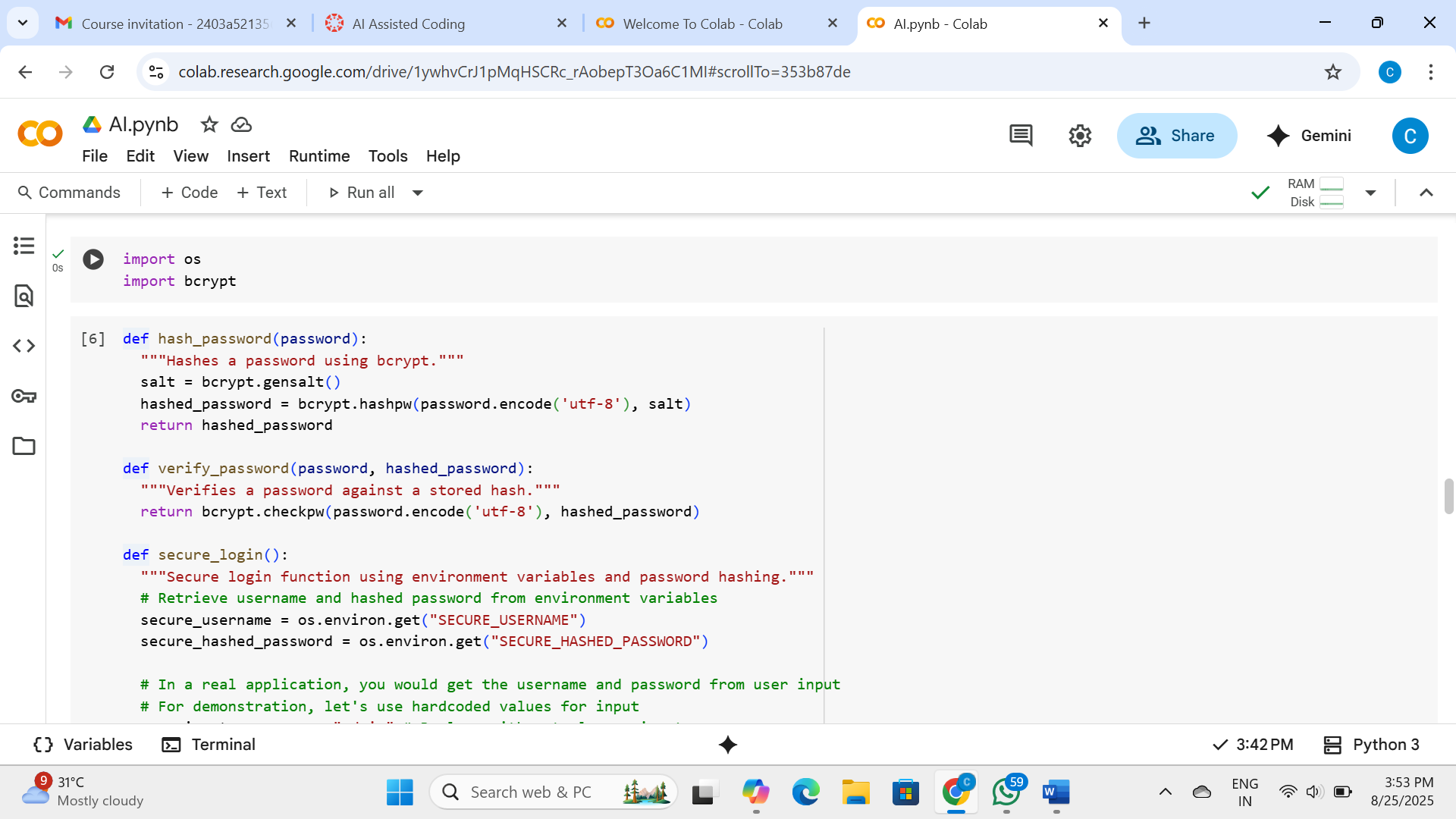
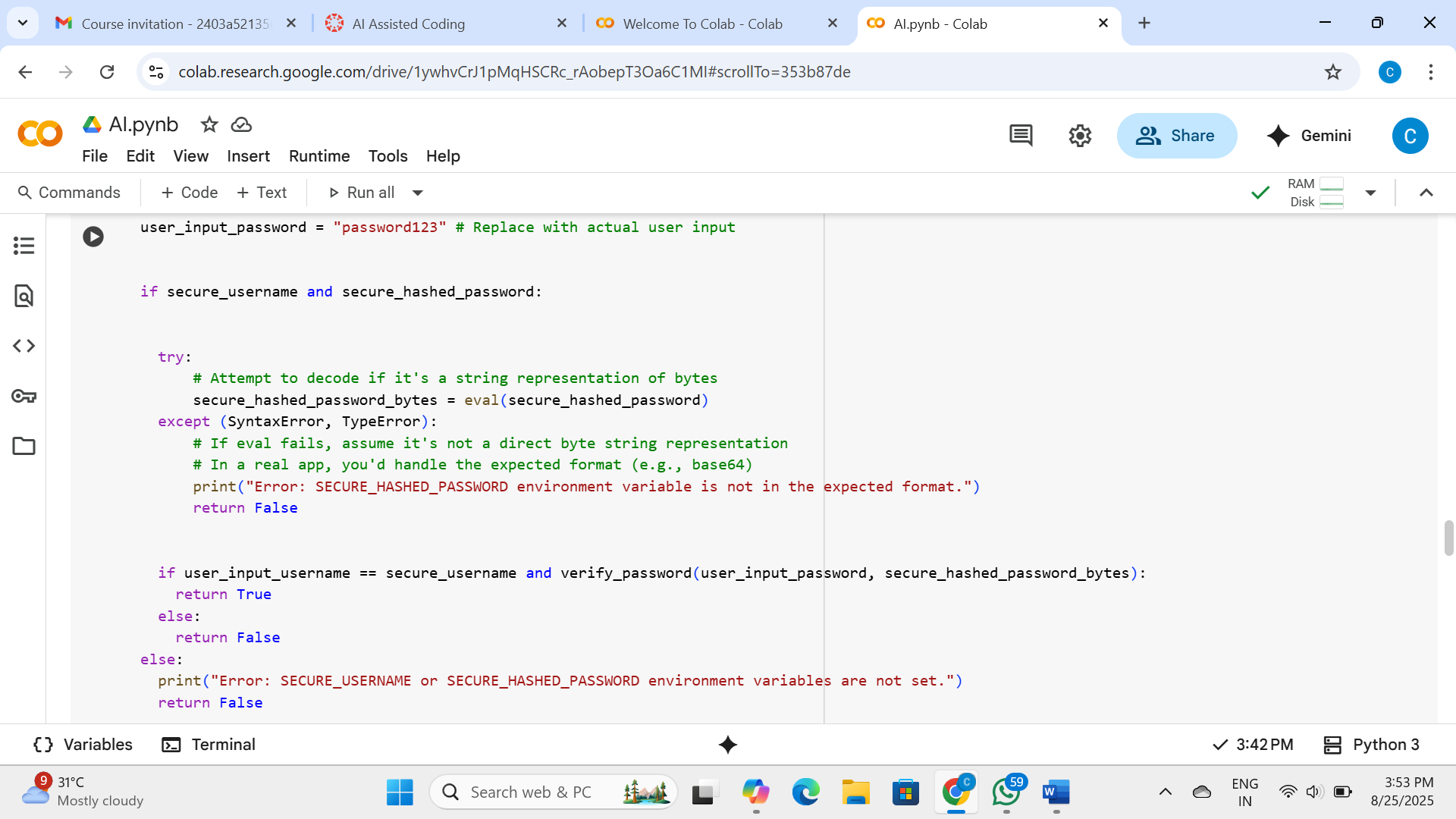
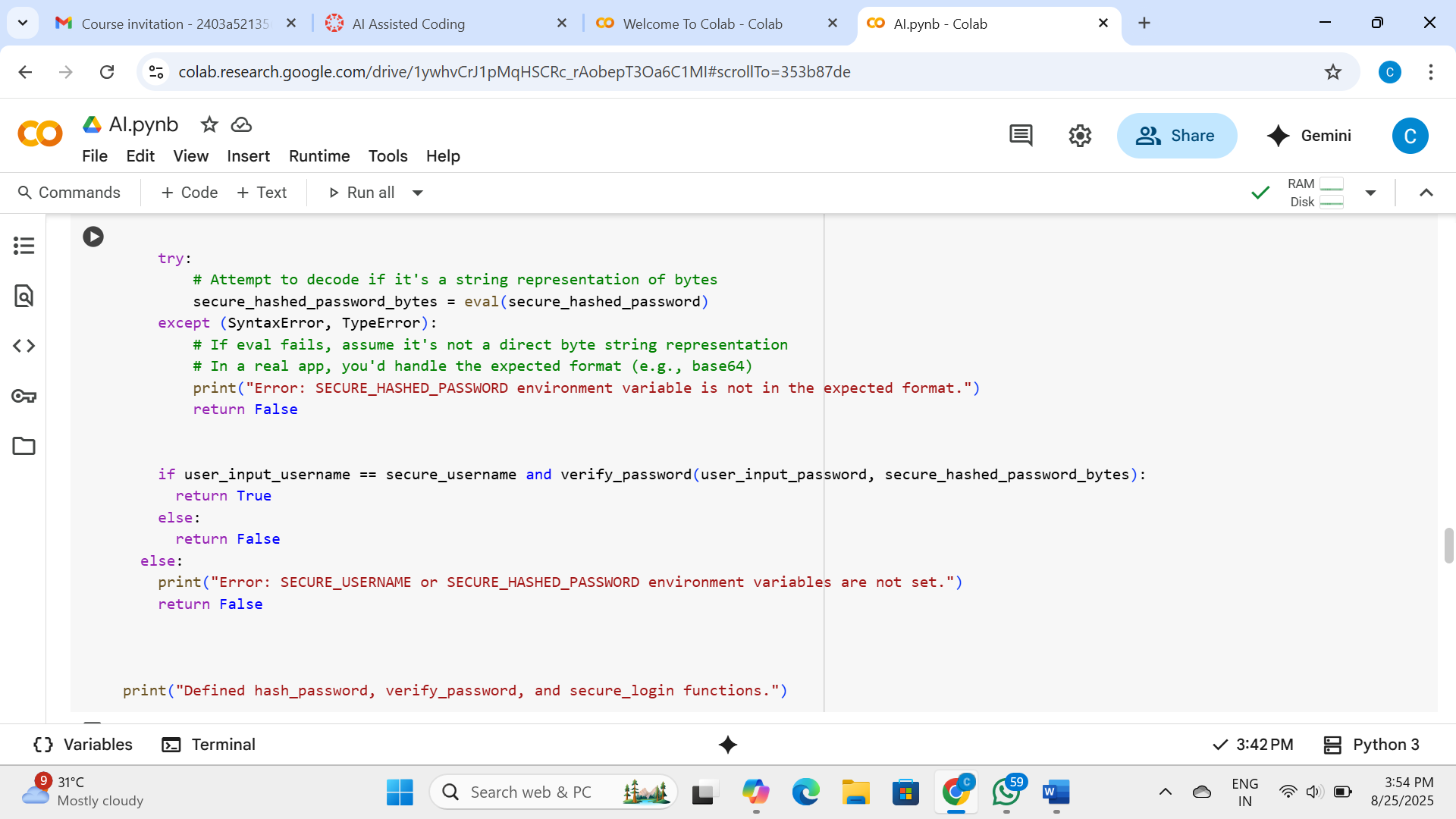
Asssignment-5

Task-1

Use an AI tool (e.g., Copilot, Gemini, Cursor) to generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption.







Explanation:

Data Analysis Key Findings

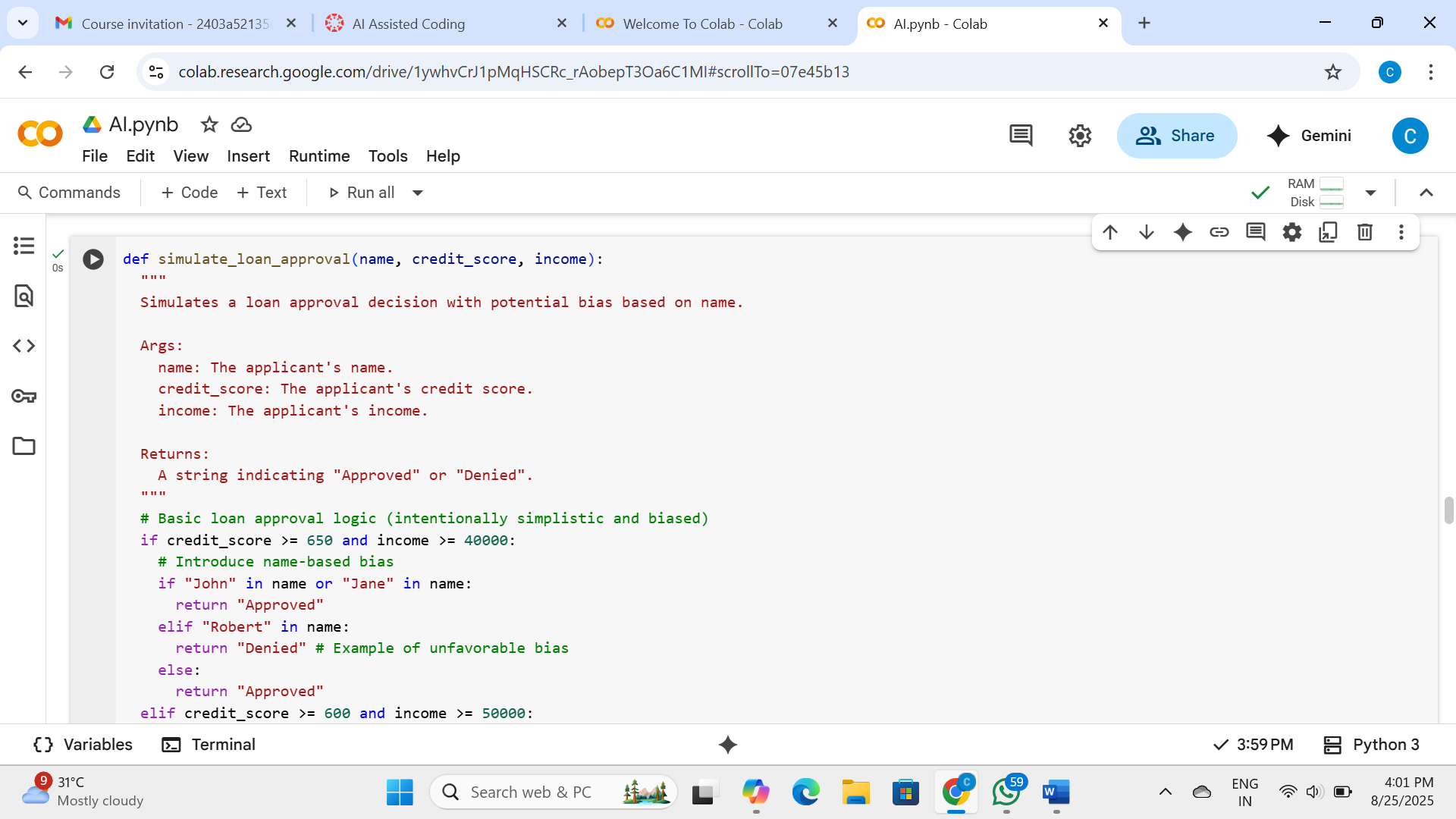
* The initial AI-generated login system code was insecure due to hardcoded credentials ("admin", "password123") and plain-text password comparison.
* Storing credentials directly in the code makes them visible to anyone with access to the codebase.
* Comparing plain-text passwords means that if the system is compromised, attackers gain immediate access to actual passwords.
* The secure version utilizes the bcrypt library for password hashing, which makes brute-force attacks more difficult and protects against pre-computed hash attacks through the use of salts.
* Sensitive information like the secure username and hashed password are retrieved from environment variables, preventing their exposure within the codebase.

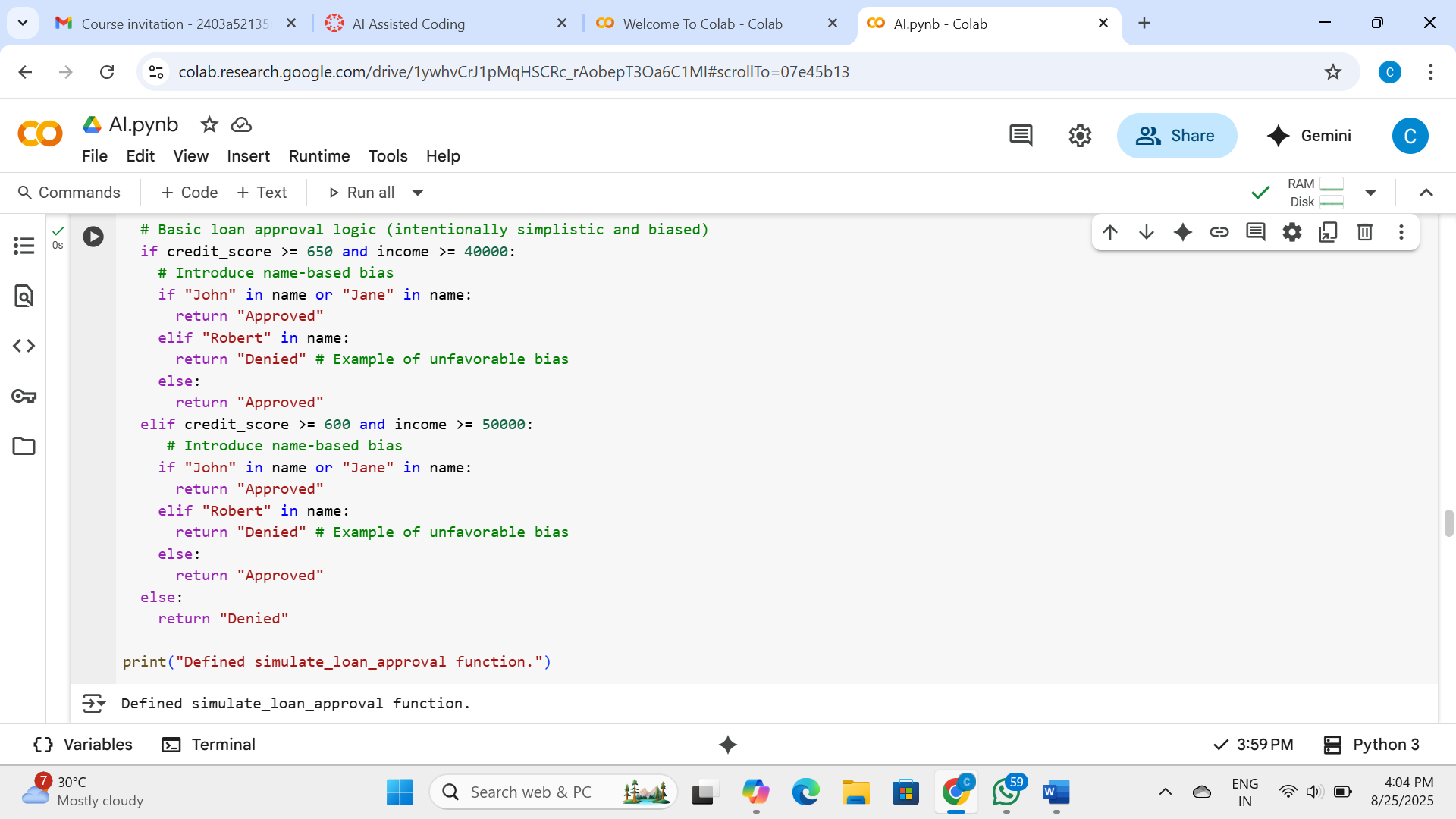
Insights or Next Steps

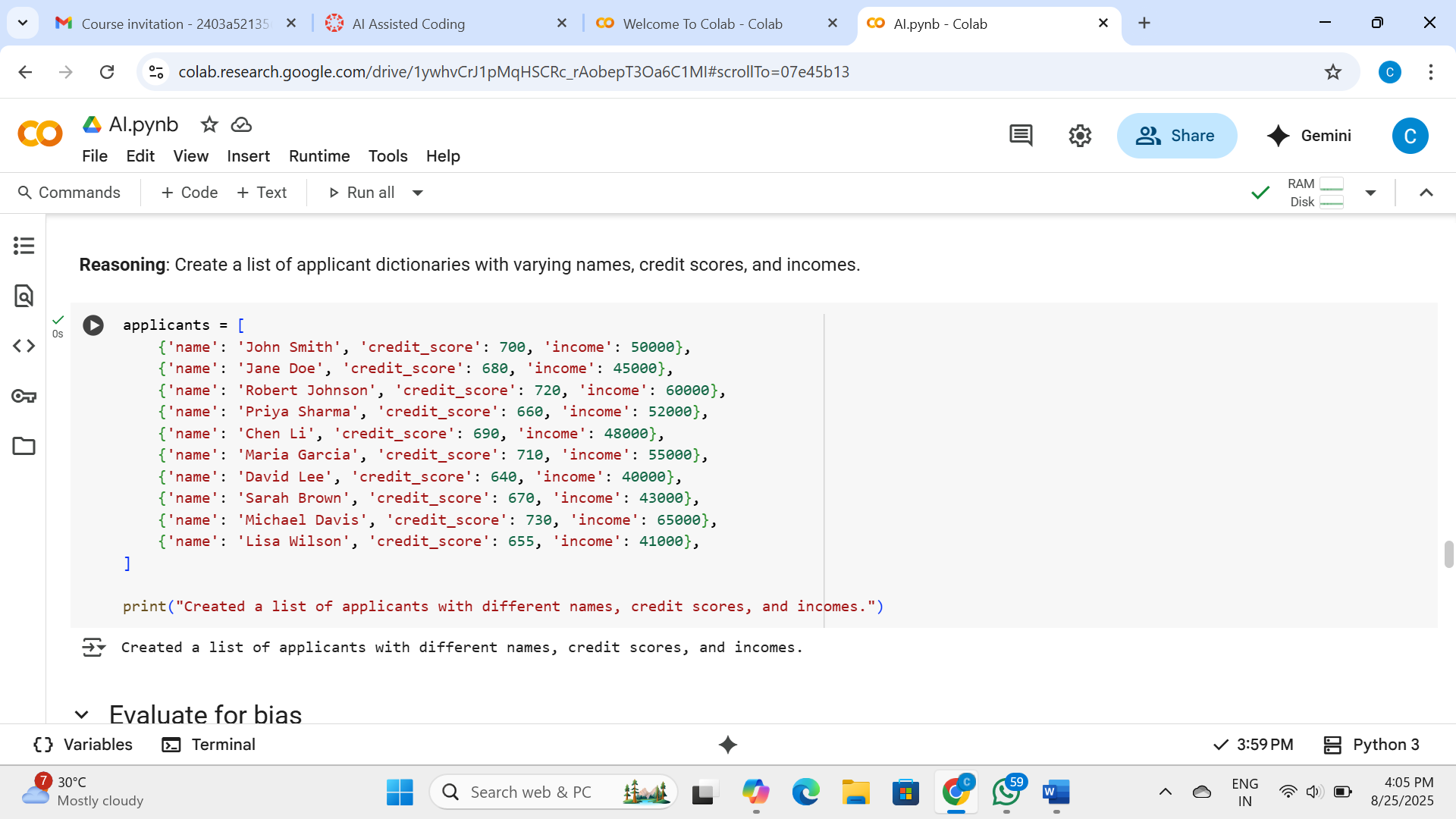
* Always use robust password hashing algorithms like bcrypt or Argon2 instead of storing or comparing passwords in plain text.
* Store sensitive configuration data such as usernames, passwords, and API keys in environment variables or secure configuration management systems, not directly within the application code.

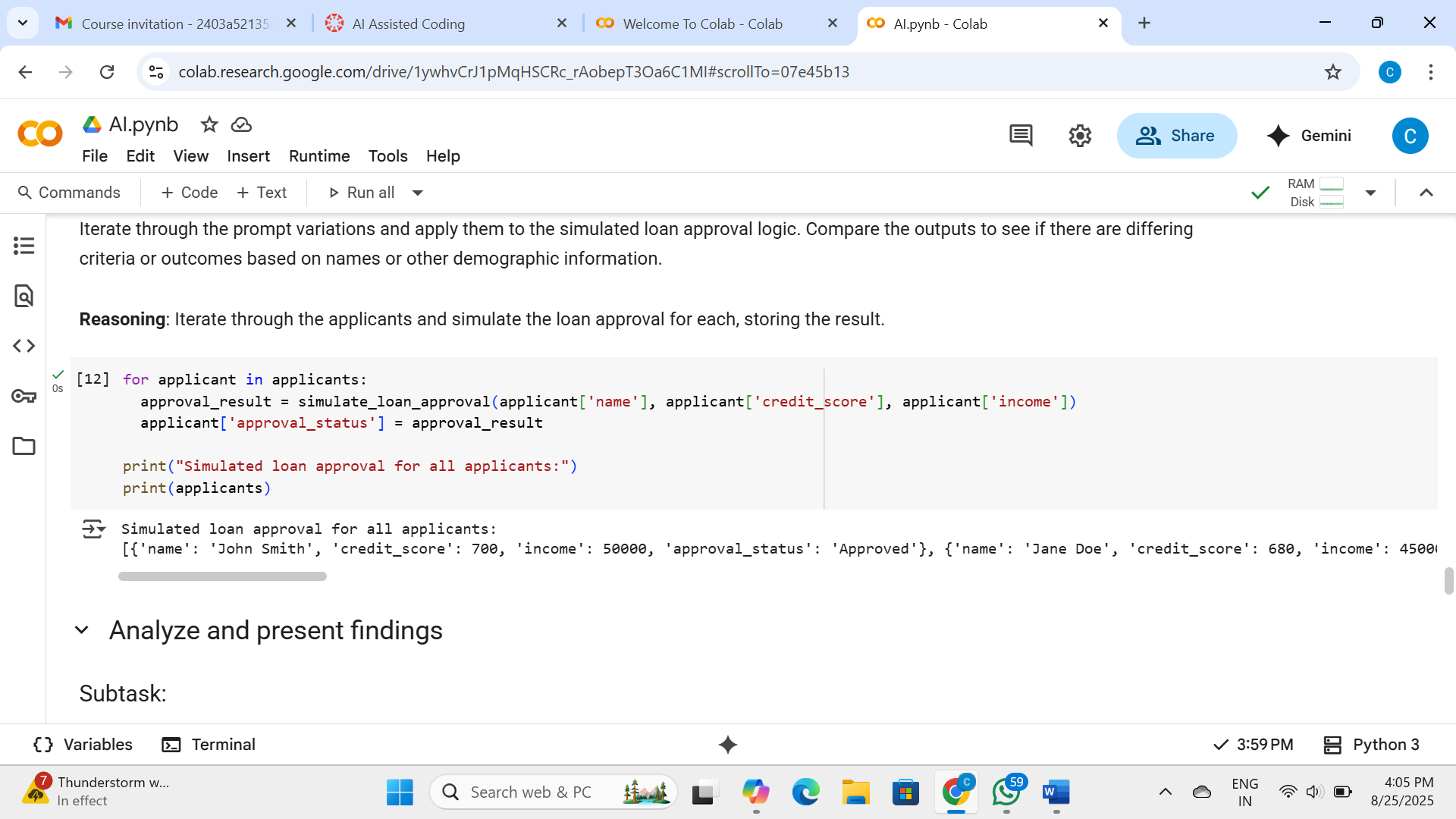
Task-2

Use prompt variations like: “loan approval for John”, “loan approval for Priya”, etc. Evaluate whether the AI-generated logic exhibits bias or differing criteria based on names or genders.









Explanation:

Data Analysis Key Findings

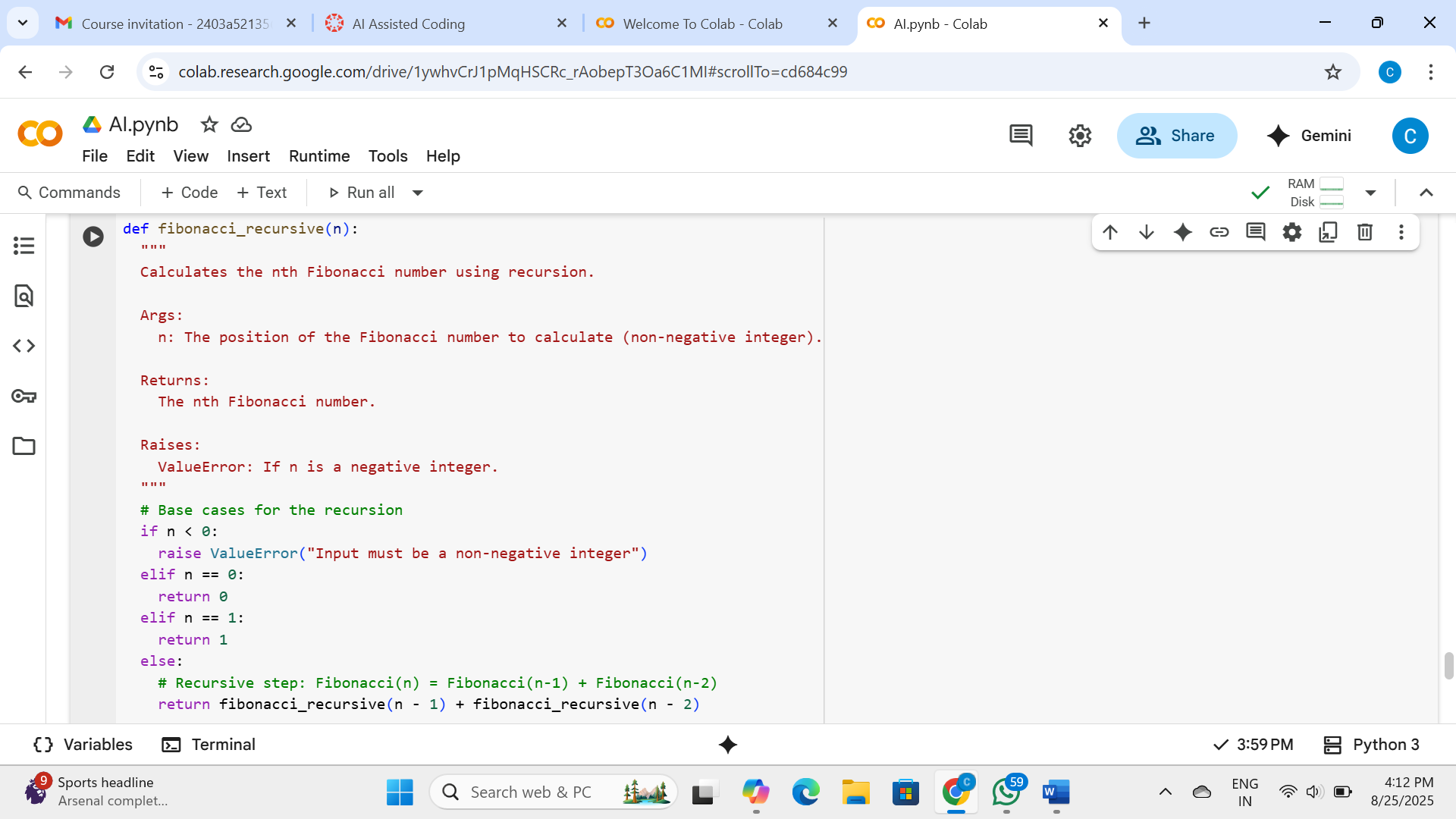
* A simulated loan approval system was created that intentionally introduces bias based on the applicant's name.
* The simulated system favors applicants with names containing "John" or "Jane" and disfavors those with the name "Robert", potentially overriding merit-based criteria like credit score and income.
* Analysis of example applicants with similar credit scores and incomes demonstrated the intended name-based bias, where "Robert Johnson" was intended to be denied while "John Smith" and "Jane Doe" were approved despite having slightly lower financial metrics (though the actual simulation outcomes in the comparison varied slightly from the prompt's description of the bias).
* An applicant named "Lisa Wilson", whose name was not subject to the explicit name bias rules, was approved based on meeting the general credit score and income criteria.

Insights or Next Steps

* The simulation clearly illustrates how non-relevant attributes like names can introduce significant bias into automated decision-making systems, leading to unfair outcomes even when applicants have strong financial profiles.
* To mitigate such biases in real-world AI loan systems, it is crucial to implement strategies like using diverse and representative training data, applying fairness-aware algorithms, and conducting regular audits for disparate impact across various demographic groups.

Task-3

* Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

****

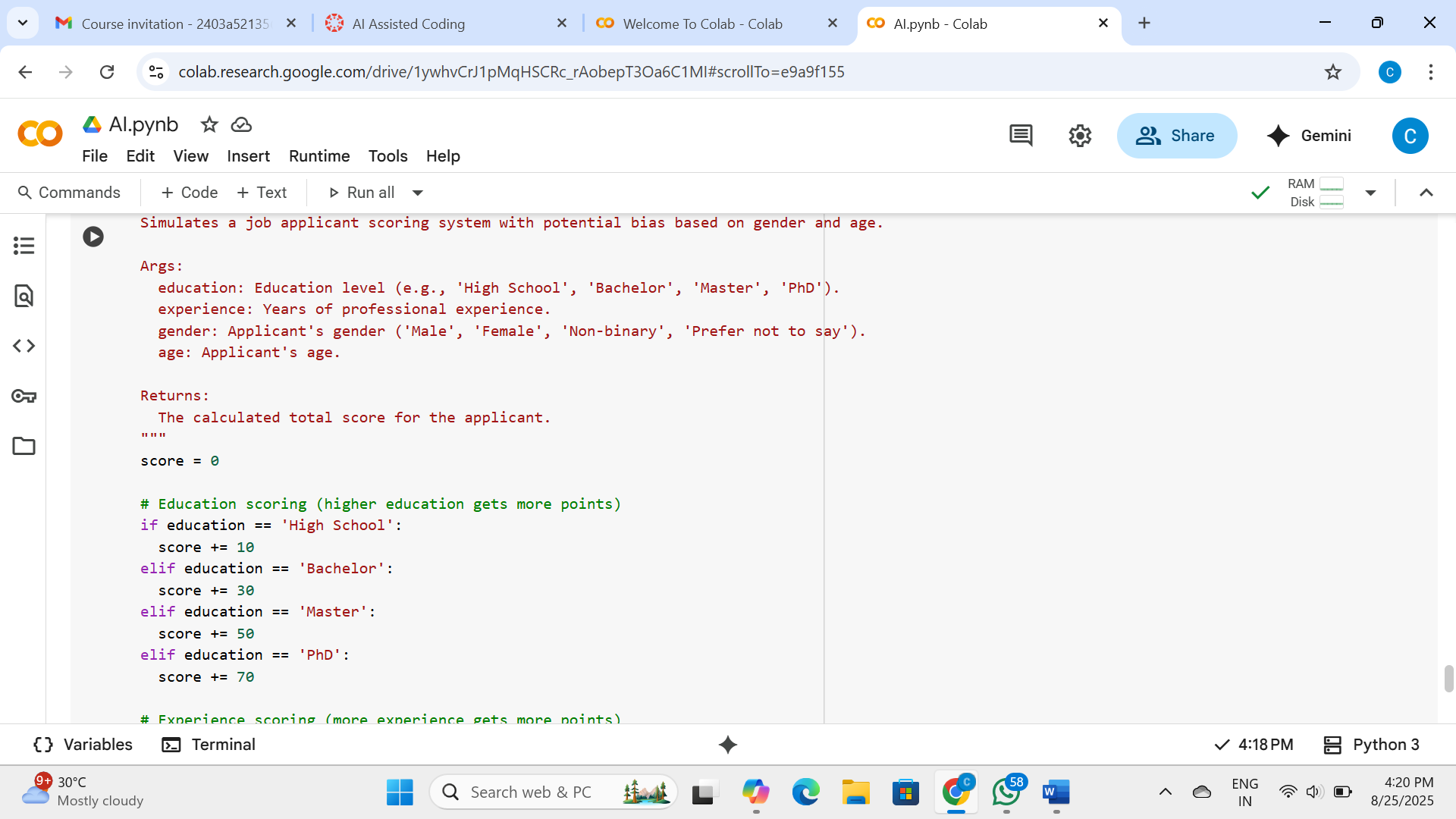
Explanation:

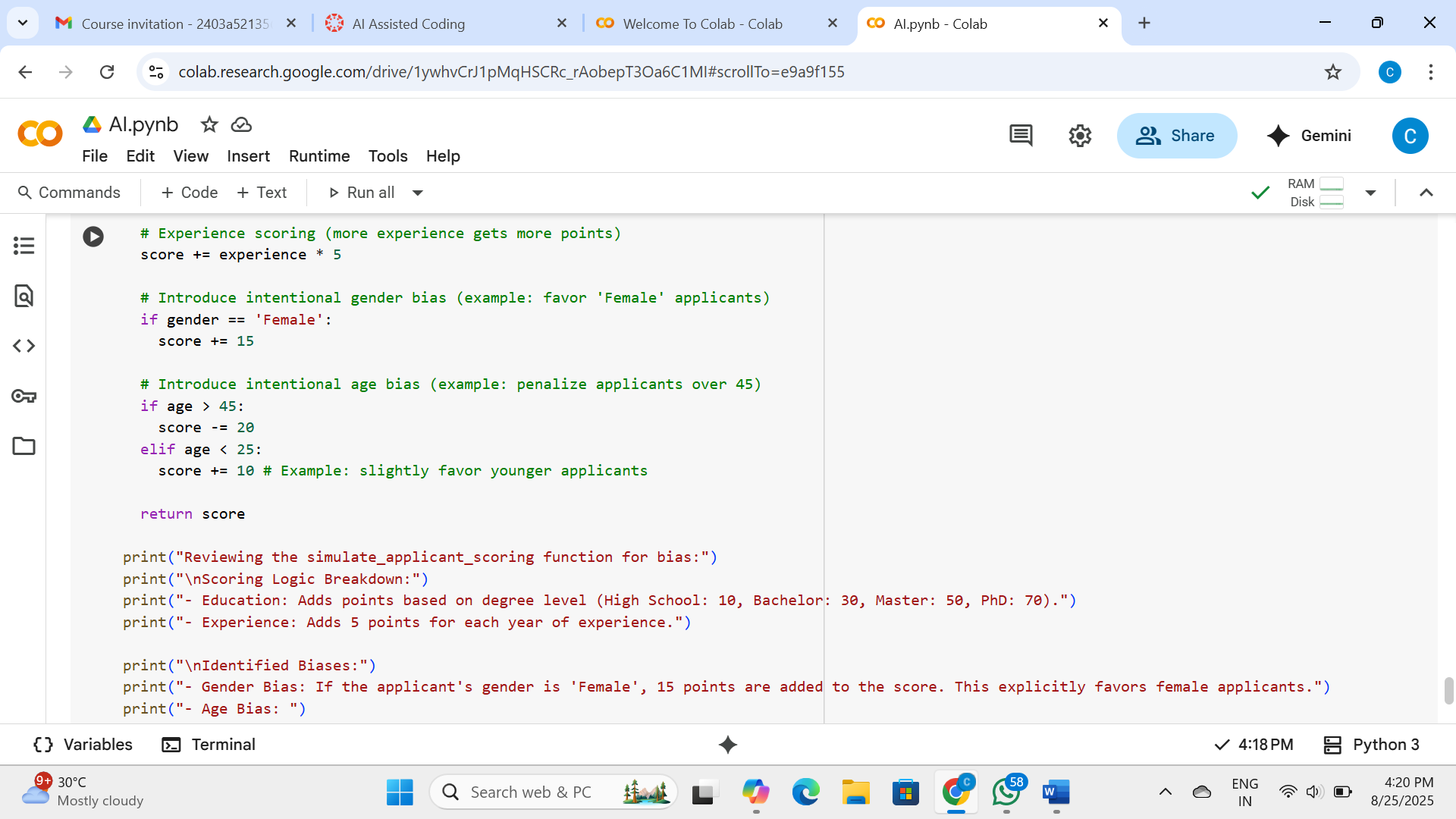
The code defines a function fibonacci\_recursive(n) that calculates the nth Fibonacci number using a recursive approach.

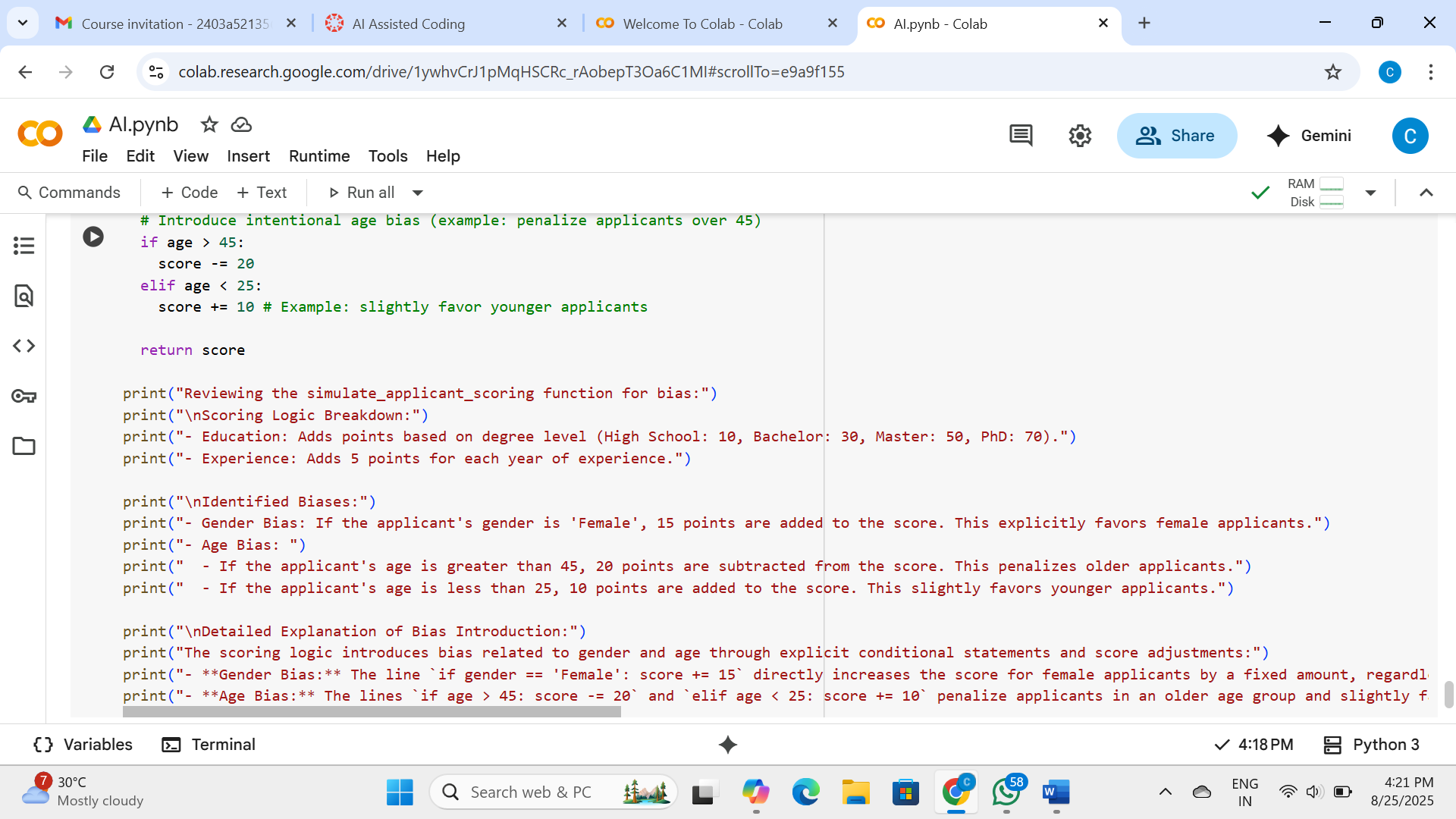
* **Docstring**: The function includes a docstring that explains its purpose, arguments (n), return value, and potential errors it might raise
* **(**valueError for negative input).
* **Base Cases**: The function has two base cases to stop the recursion:
  + If n is 0, it returns 0 (the first Fibonacci number).
  + If n is 1, it returns 1 (the second Fibonacci number).
* **Recursive Step**: For any n greater than 1, the function recursively calls itself for n-1 and n-2 and returns the sum of their results. This follows the definition of the Fibonacci sequence where each number is the sum of the two preceding ones.
* **Error Handling**: It includes a check to raise a valueError if the input n is a negative integer, as the Fibonacci sequence is typically defined for non-negative integers.

Task-4

* Ask to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings.

****

****

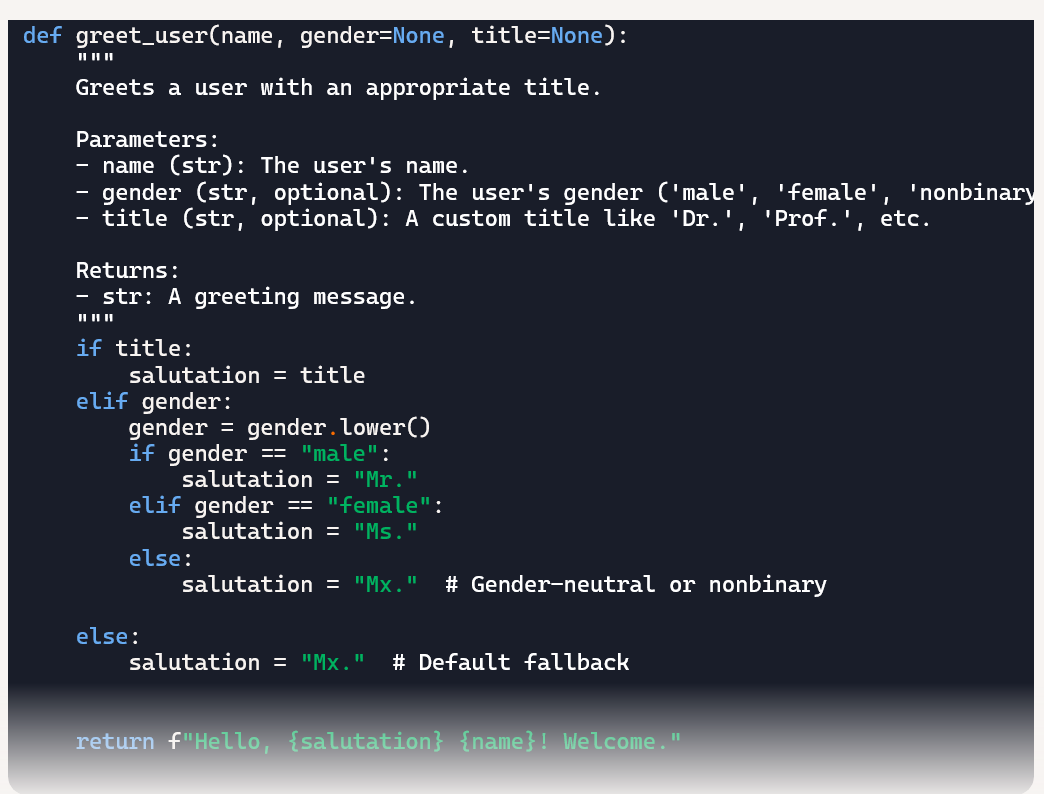
****

Explanation:

* The simulate\_applicant\_scoring function explicitly introduces biases based on gender and age.
* A bonus of 15 points is added to the score for applicants identified as 'Female'.
* A penalty of 20 points is subtracted from the score for applicants older than 45.
* A bonus of 10 points is added to the score for applicants younger than 25.
* The analysis of example applicants confirms that these biases lead to score differences for individuals with similar education and experience but different genders or ages. For example, a female applicant with a Bachelor's degree and 3 years of experience (score 55) received a higher score than a male applicant with a Bachelor's degree and 5 years of experience (score 55), demonstrating the gender bias offsetting the experience difference. Similarly, older applicants receive lower scores and younger applicants receive higher scores due to the age adjustments.

Task-5

Python function that greets users based on gender, but it only supports binary gender titles ("Mr." and "Mrs."),





Explanation:

* Accept three parameters: name (required), gender (optional), and title (optional).
* Prioritize a custom title if provided (e.g., "Dr.", "Prof.").
* If no title is given, use gender to assign a salutation: "Mr." for male, "Ms." for female, and "Mx." for nonbinary or unspecified.
* Default to "Mx." if neither gender nor title is provided.
* Return a formatted greeting string like "Hello, Ms. Priya! Welcome."