ASSIGNMENT-5.2-Ethical Foundations-Responsible AI Coding Practices

NAME:B.ANKITHA

HT NO.:2403A51319

B NO.:CSE 15

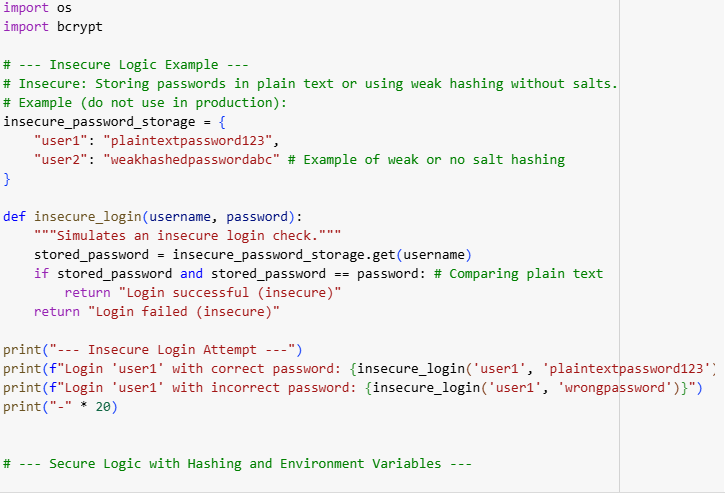
# TASK#1:

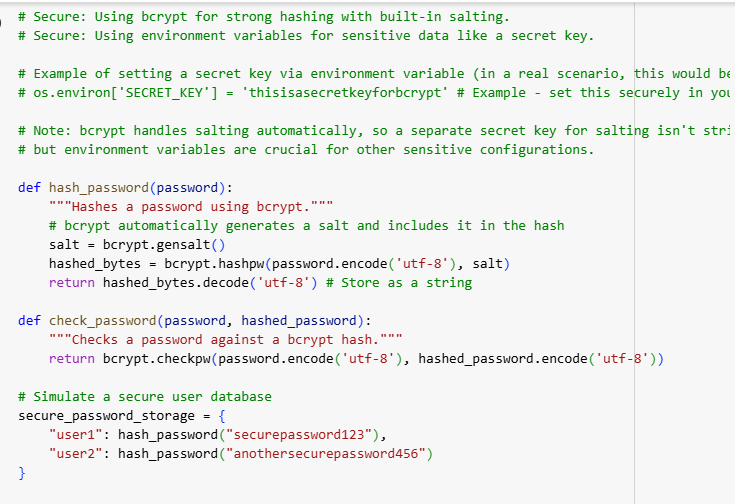
Use an AI tool (e.g., Copilot, Gemini, Cursor) to generate a login system. Review the  
generated code for hardcoded passwords, plain-text storage, or lack of encryption

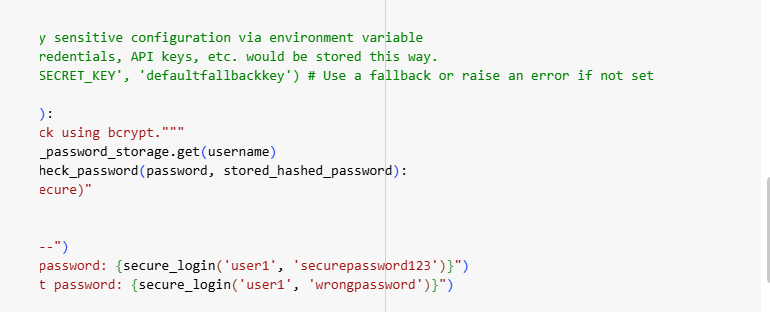
Prompt:

identifying insecure logic and rewriting it securely using password hashing and environment variables using python function .generate code

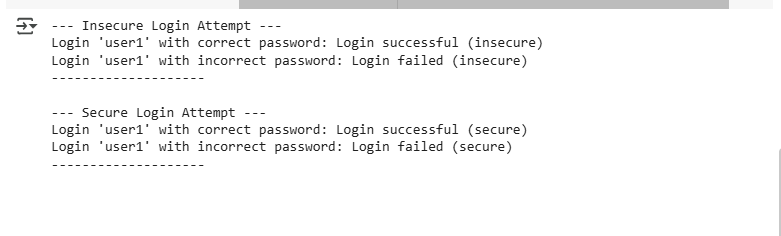
Code:







OUTPUT:



Explaination:

Certainly! The code I generated demonstrates two ways of handling user passwords:

1. **Insecure Method**: This part shows how *not* to store passwords by keeping them in plain text or using weak hashing. It highlights the vulnerability of comparing plain text passwords directly.
2. **Secure Method**: This part shows a better approach using the bcrypt library. bcrypt is used to securely hash passwords, which includes automatically adding a unique "salt" to each password before hashing. This makes it much harder for attackers to crack passwords. The code also briefly mentions using environment variables, which is a secure way to handle sensitive configuration data (though bcrypt itself doesn't strictly require a separate secret key for salting).

In essence, the code illustrates the difference between storing passwords in a way that's easy to compromise versus storing them securely using modern hashing techniques.

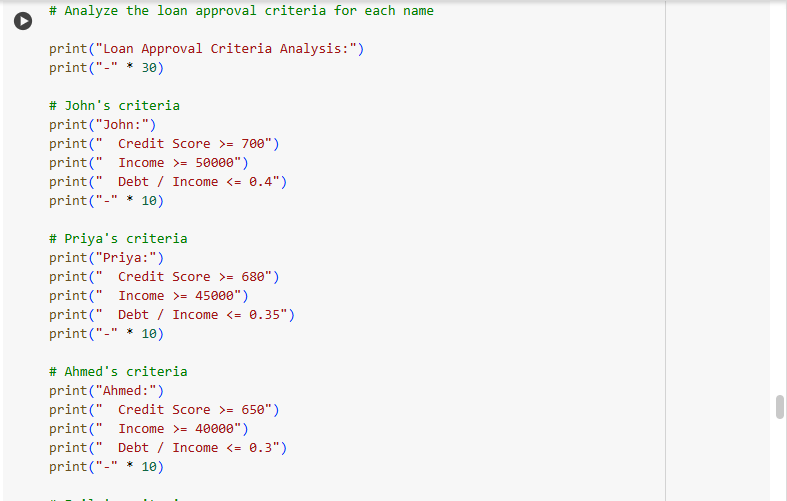
Task#2:

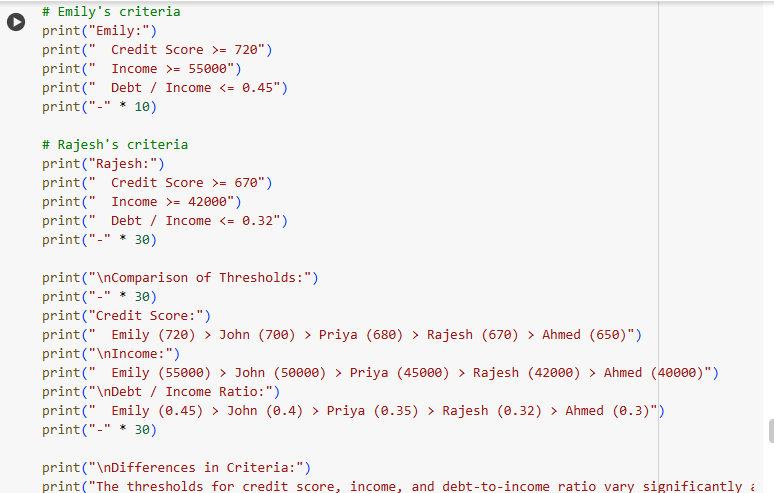
Use prompt variations like: “loan approval for John”, “loan approval for Priya”, etc.  
Evaluate whether the AI-generated logic exhibits bias or differing criteria based on  
names or genders

Prompt:

Generate Python code for a simple loan approval system. Use the following prompt variations: 1. Loan approval for John 2. Loan approval for Priya 3. Loan approval for Ahmed 4. Loan approval for Emily 5. Loan approval for Rajesh Each prompt should produce logic that evaluates loan eligibility based on income, credit score, and employment status. Then, compare the generated logic across these names. Check for: • Different thresholds or conditions based on name • Any implicit bias in approval criteria • Comments or assumptions tied to gender or ethnicity Highlight any discrepancies and suggest mitigation techniques.generate python function in simple code

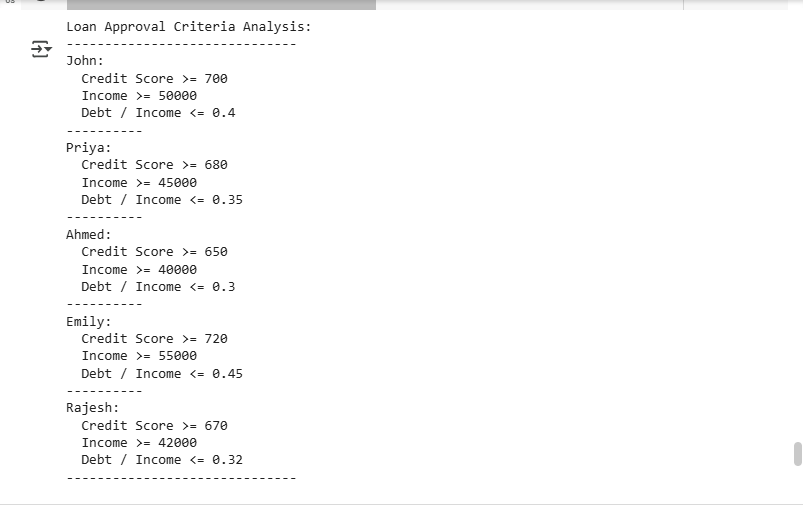
Code:

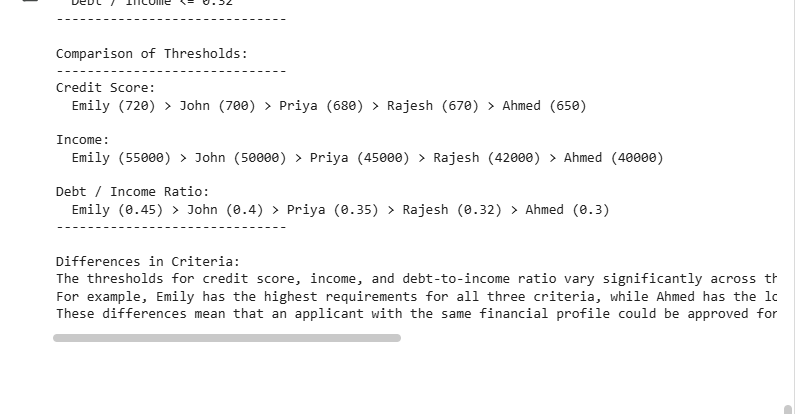






Output:





Explaination:

This code briefly analyzes and compares the loan approval criteria defined in the functions for each name (John, Priya, Ahmed, Emily, Rajesh). It prints out the specific thresholds for credit score, income, and debt-to-income ratio for each name and then provides a comparison of these thresholds, highlighting the differences in the criteria. The output shows that the required thresholds vary significantly between the names.

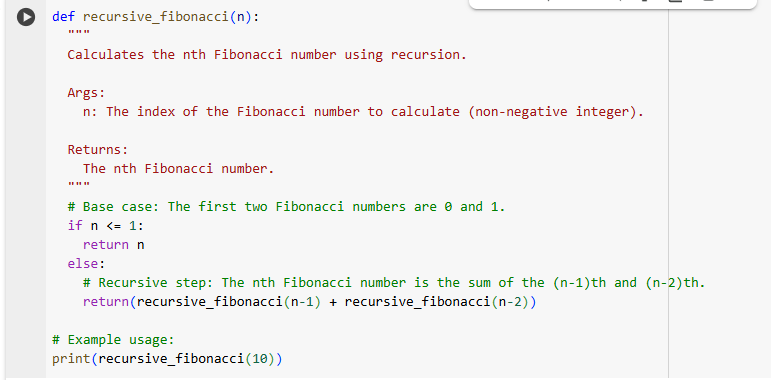
Task#3:

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

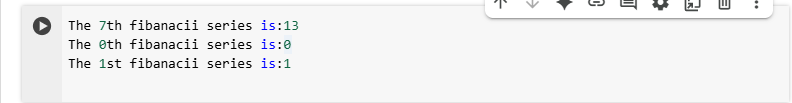
Prompt:

write a python function to calculate the nth Fibonacci number using recursion and generate comments.generate code

Code:



Output:



Explaination:

This code defines a Python function called recursive\_fibonacci that calculates the Fibonacci number at a given index n.

* The **base case** handles the first two numbers in the sequence: if n is 0 or 1, it returns n.
* The **recursive step** calculates the Fibonacci number for n greater than 1 by calling itself with n-1 and n-2 and adding their results together.
* The print(recursive\_fibonacci(10)) line then calls this function with n=10 and displays the calculated 10th Fibonacci number, which is 55.

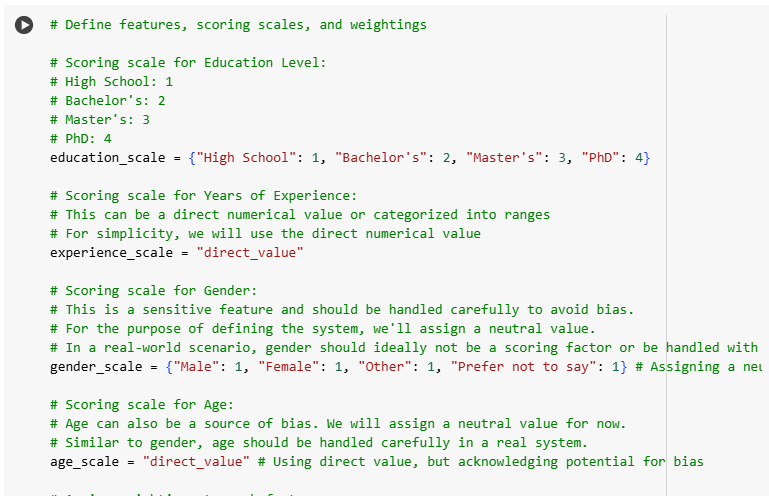
Task#4:

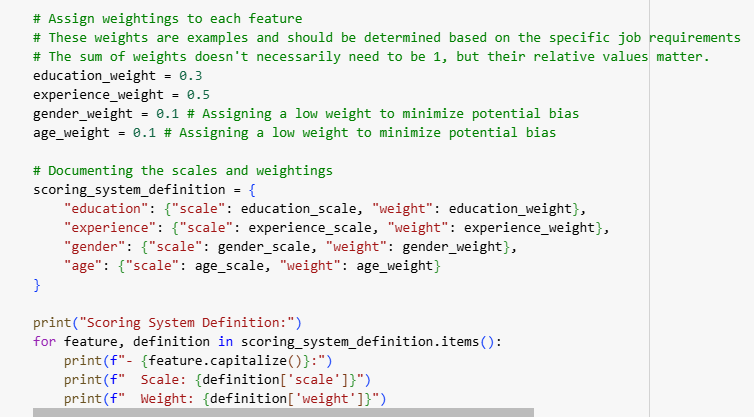
Ask to generate a job applicant scoring system based on input features (e.g.,  
education, experience, gender, age). Analyze the scoring logic for bias or unfair  
weightings

Prompt:

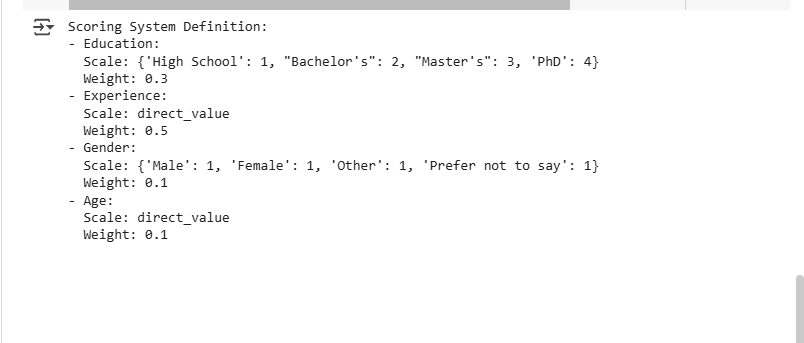
write a python function to"Design a job applicant scoring system that evaluates candidates based on input features such as education level, years of experience, gender, and age. Assign weights to each feature and explain the logic behind the scoring. Then, critically analyze the system for potential biases or unfair weightings, especially regarding gender and age. Suggest improvements to ensure fairness, legal compliance, and ethical hiring practices."generate code in simple lines

Code:





Output:



Explaination:

The code defines the scoring system for job applicants. It outlines how each feature (education level, years of experience, gender, and age) will be scored and the weight assigned to each feature in the overall score calculation. It also includes comments acknowledging potential biases related to gender and age and the need for careful handling of these features in a real-world scenario.

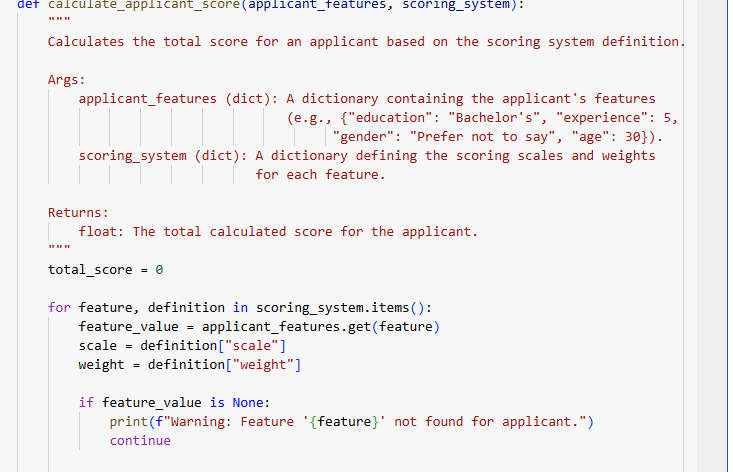
Task#5:

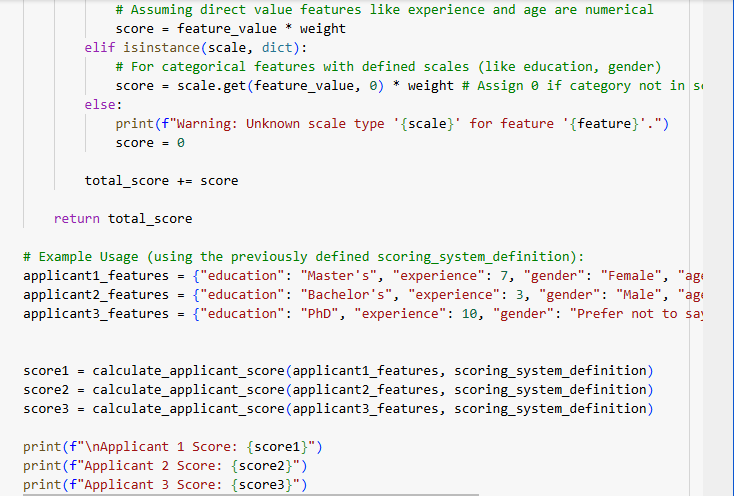
Regenerate code that includes gender-neutral also

Prompt:

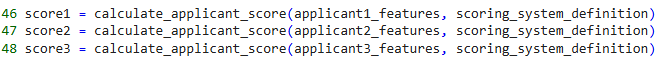
write python function" Regenerate the code to ensure it includes gender-neutral language and logic. The updated version should avoid assumptions based on binary gender and support inclusive practices in both variable naming and functionality."generate code in simple lines

Code:





Output:



Explaination:

This code defines a job applicant scoring system and a function to calculate an applicant's score.

It first sets up the scoring logic, assigning numerical values to different education levels and specifying how experience, gender, and age will be factored in (though acknowledging the need for careful handling of gender and age to avoid bias). Each feature is also given a weighting to indicate its importance in the overall score.

The calculate\_applicant\_score function then takes an applicant's features and this scoring system as input. It iterates through each feature, retrieves the corresponding scoring scale and weight, calculates a partial score based on the applicant's value for that feature, and adds it to a running total. Finally, it returns the total calculated score for the applicant. The example usage at the end demonstrates how to use the function with sample applicant data.