**Assignment-5.3**

**M.Vivek**

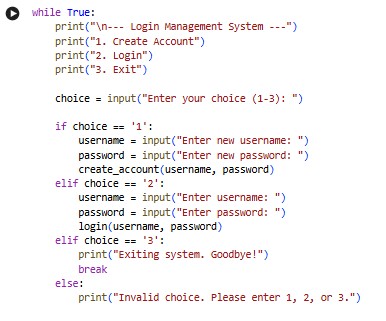
**Batch-41**

**2303A52460**

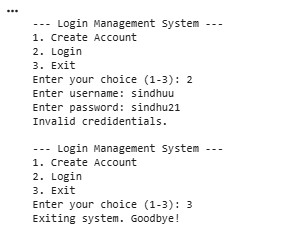
**Task 1: Privacy and Data Security in AI-Generated Code PROMPT (**insecure**):**

generate a python code for login management . based on users choice user may login or create an account.

**CODE1:**

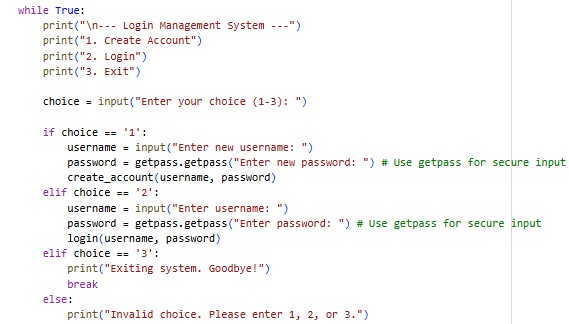


**Output:**

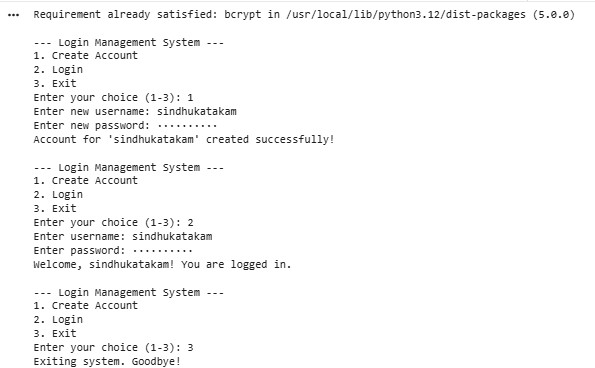


**Prompt secure:** (code2)generate a readable and efficient code for login system .based on user choice the user may login or register. the data has to be secured and encrypted . check whether the credidentials are hardcoded and passwords are stored or compared in plain text. .improves the overall security . after registration store the data of user securely and it should not be displayed in the output.

**CODE:**



**Output:**



**Explanation:**

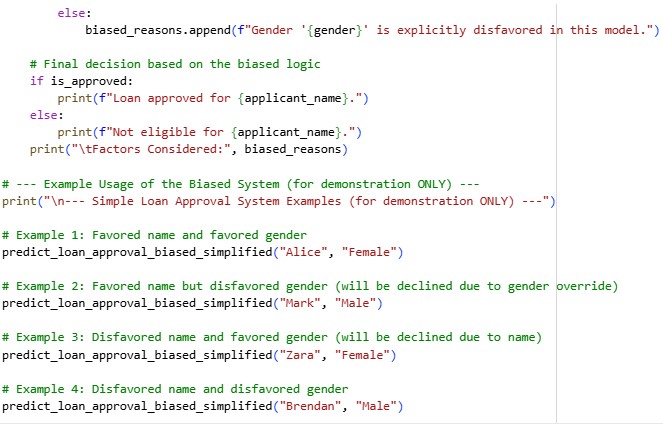
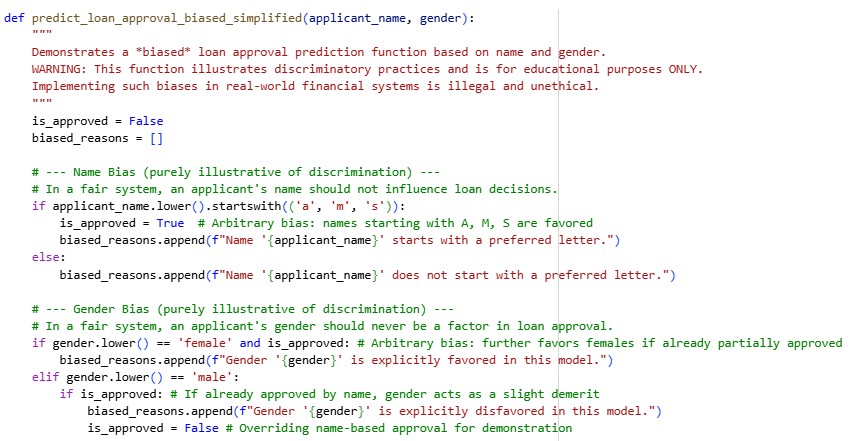
Code1 is less secure because it stores and compares passwords in plain text, shows specific error messages that can reveal valid usernames, and displays the password while typing. Code2 improves security by hashing passwords with bcrypt, using secure password comparison, and hiding typed passwords with getpass. Code2 also gives generic error messages, which makes guessing credentials harder. In terms of readability, Code1 is simpler and easier for beginners because it has fewer imports and no hashing logic. Code2 is slightly more complex due to extra steps but is still well-organized and worth using for real security.

**Task 2: Bias Detection in AI-Generated Decision Systems**

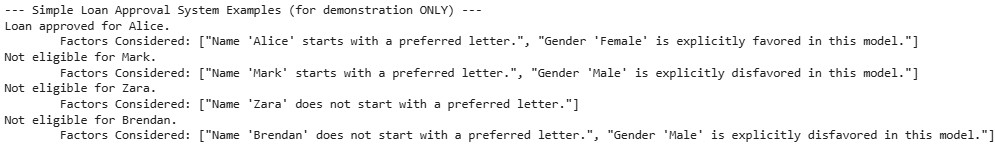
**Prompt: (biased)**

Generate a simple Python program for a loan approval system that includes biased logic based on applicant names and gender. The program should evaluate each applicant and print Loan approved if they meet the specified biased conditions, otherwise print Not eligible. Ensure the code is efficient.

**CODE:**



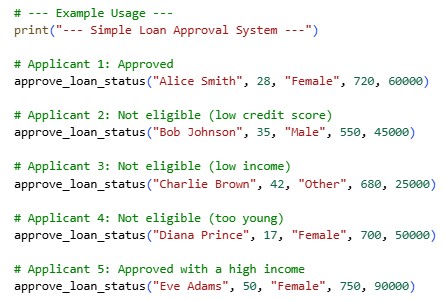
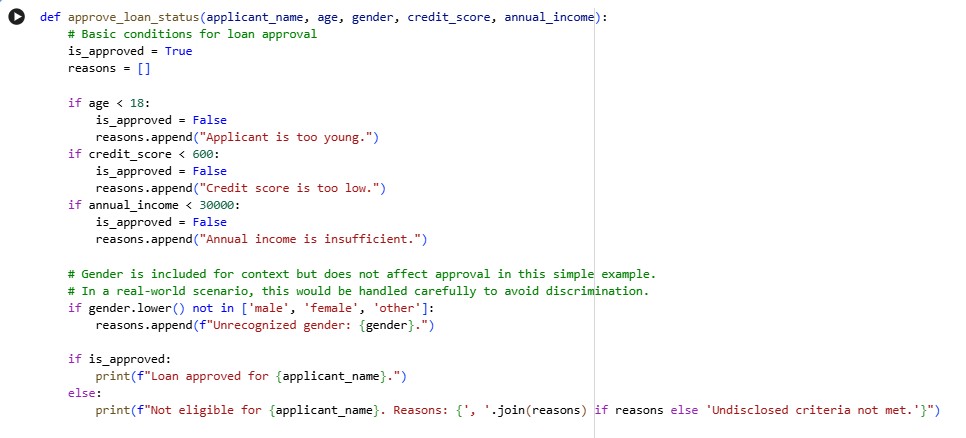
**OUTPUT:**



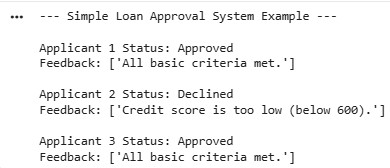
**Prompt: (Unbiased)**

generate a simple python code for loan approval system. that vary applicant names and gender . Generate an output that prints Loan approved if the conditions satisfy, if not print Not eligible.

**Code:**



**Output:**



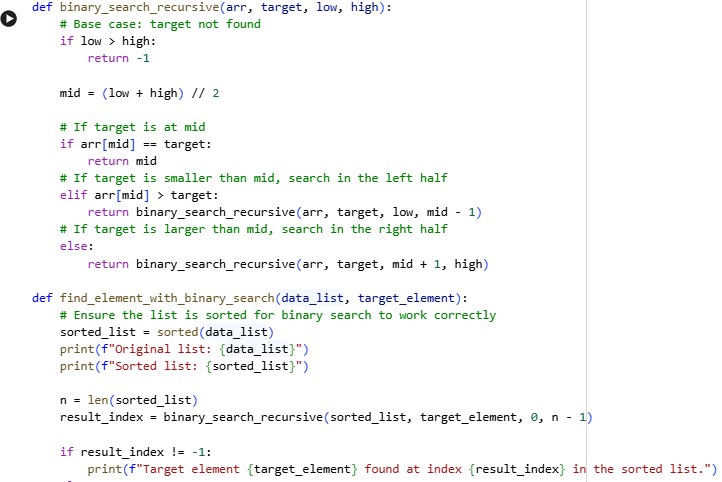
**Explanation:**

The biased code intentionally demonstrates discriminatory bias by making loan decisions based on irrelevant attributes such as applicant name and gender, which leads to unfair and unethical outcomes. It treats certain names and genders preferentially, showing how biased logic can create disparate treatment. In contrast, the unbiased code evaluates applicants using relevant financial criteria like age, credit score, and income, which are appropriate and legitimate factors in real loan assessments. Although gender and ethnicity are collected, they do not influence the decision, avoiding discriminatory impacts. Overall, the first code illustrates harmful bias, while the second reflects fairer and more responsible decision-making.

**Task 3: Transparency and Explainability in AI-Generated Code (Recursive Binary Search) Prompt:**

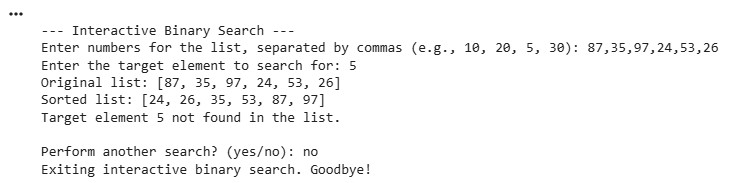
|  |  |
| --- | --- |
| generate a binary search python code using recursion. the list has to be sorted | |
| before searching for the target element. |  |

**Code:**





**Output:**

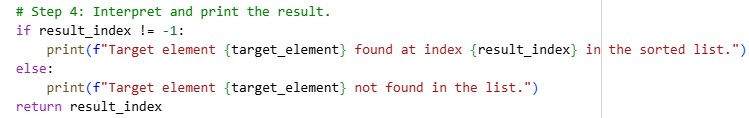
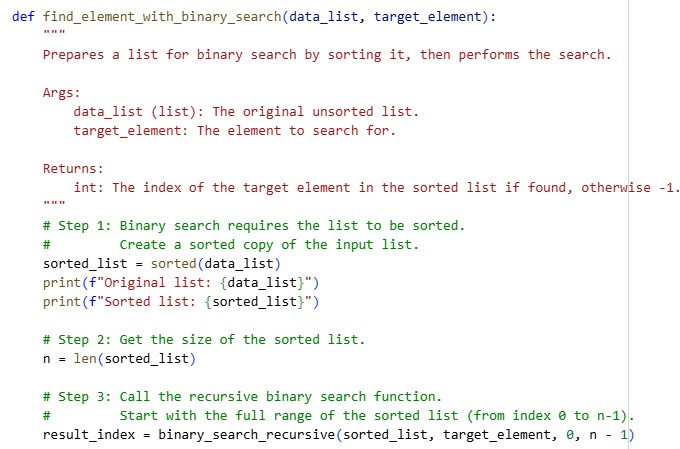
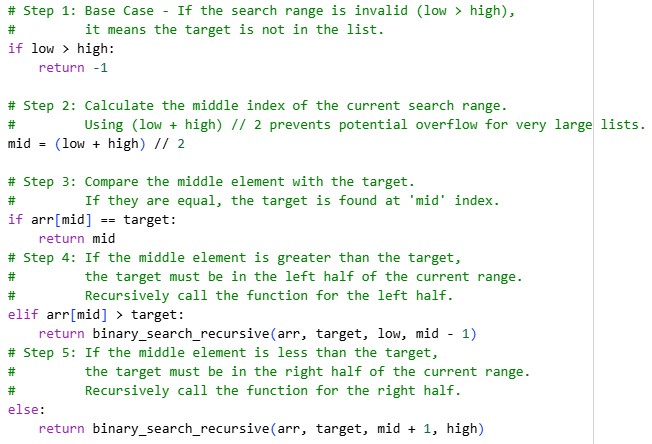
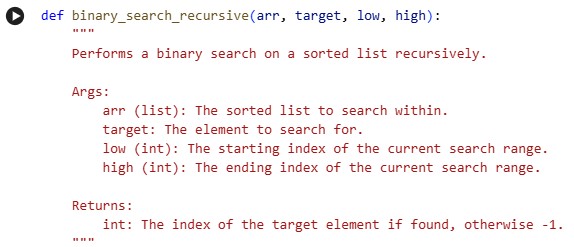


**Prompt:**

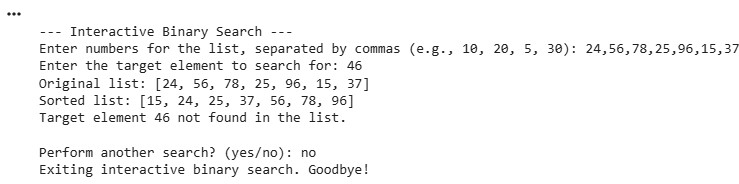
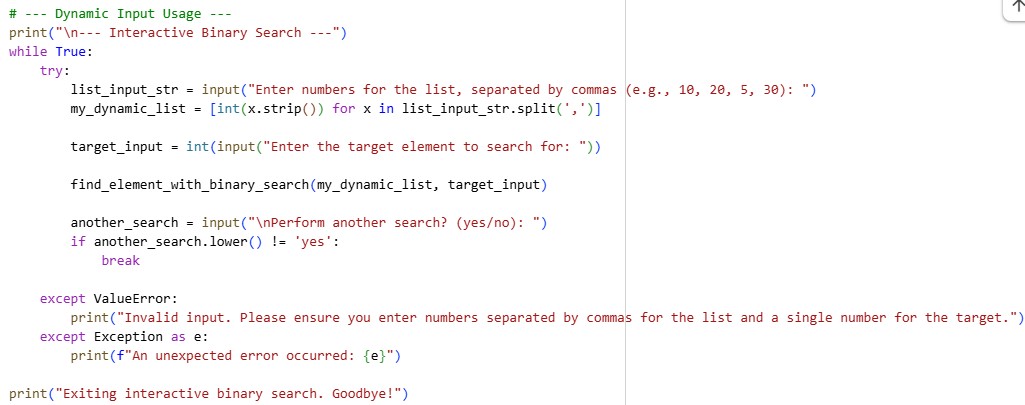
|  |  |
| --- | --- |
| generate a simple python code for binary search using recursion.sort the | |
| list first and then search for the target element. mention clear inline |  |

comments that includes step by step explanation of recursive logic and code has to be structured , understandable for beginners and readable.

**Code:**



**Output:**

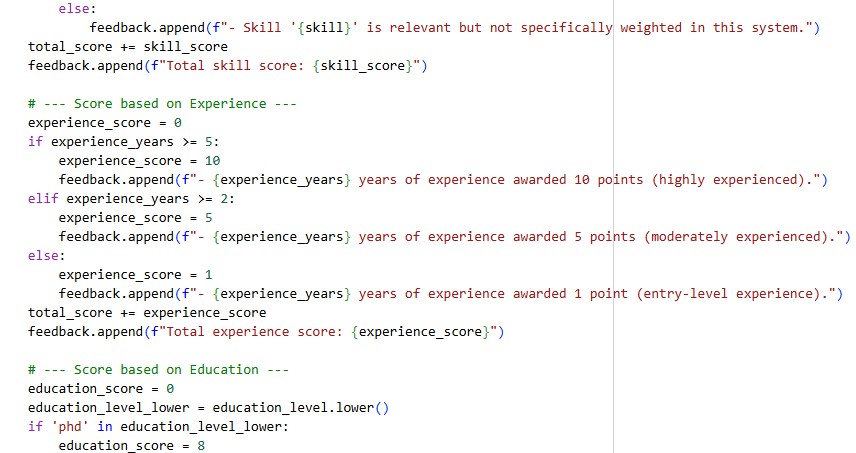


**Explanation:**

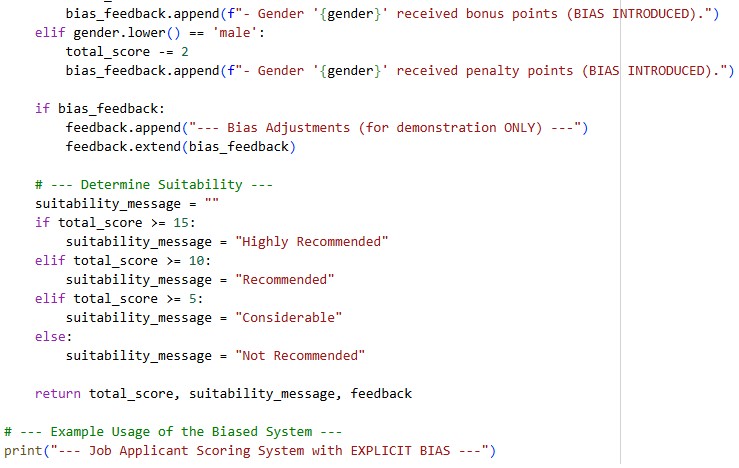
Both versions of the binary search code correctly implement recursive binary search logic and ensure the input list is sorted before searching, which is essential for correctness. The first version focuses on functionality with minimal comments, while the second version is more detailed and documentation-rich, explaining each step for better readability and learning. Both codes include interactive input handling, type validation, and user prompts, demonstrating good user interaction flow. Error handling is present in both, preventing crashes due to invalid input formats. Overall, the second implementation is more structured and educational, while the first is simpler but functionally equivalent and efficient.

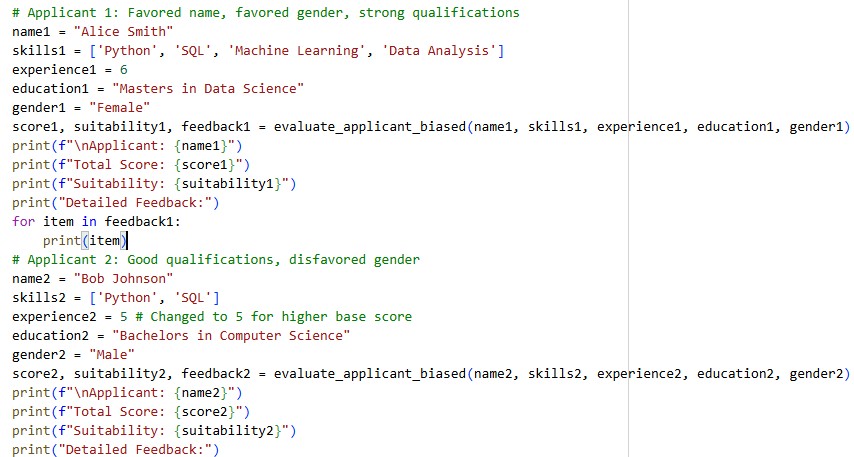
**Task-4 : Ethical Evaluation of AI-Based Scoring Systems**

**Code: (biased)**

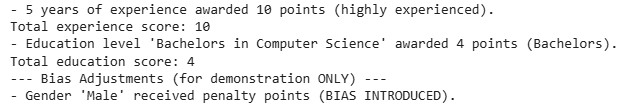
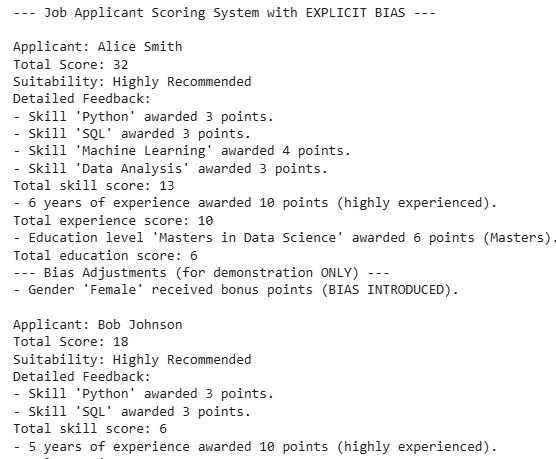






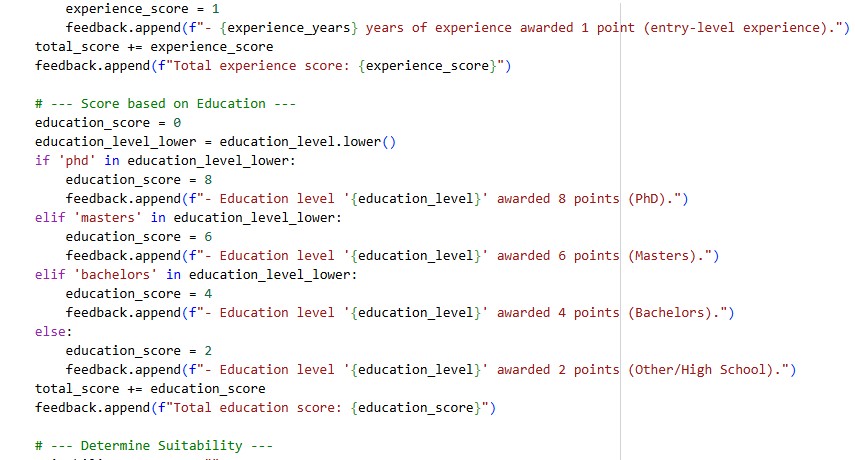
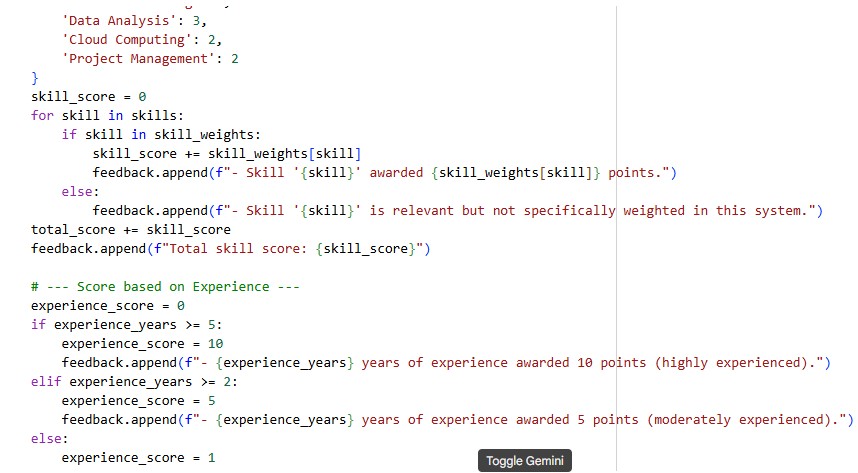
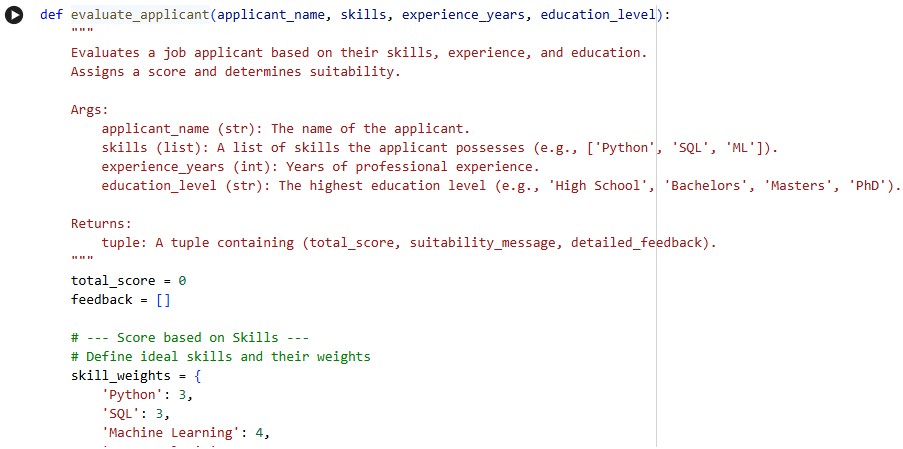


**Output:**

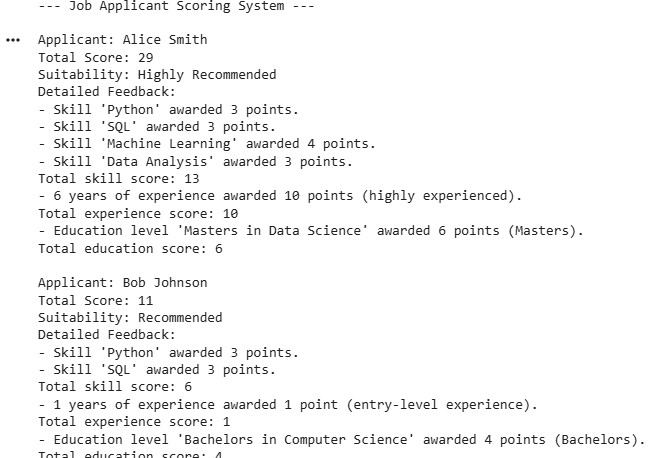


**Prompt: (unbiased)** generate a python code for job applicant scoring system based on their skills experience and education check whether there is any bias based on gender ,name or unrelated features influence scoring, Whether the logic is fair and objective.

**Code:**



**Output:**



**Explanation:**

The biased code unfairly changes scores based on name and gender, even though these have nothing to do with job ability. This creates discrimination and unequal treatment. The unbiased code only uses skills, experience, and education, which are relevant to hiring. This makes it more fair and objective. So, the biased code is unethical, while the unbiased code is suitable for real evaluations.

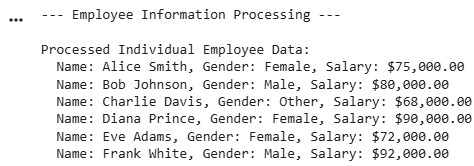
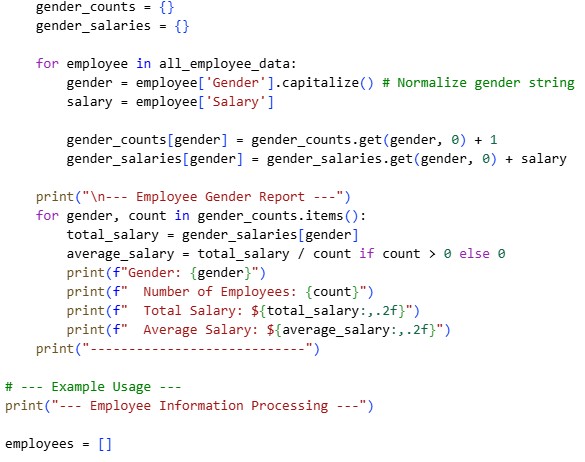
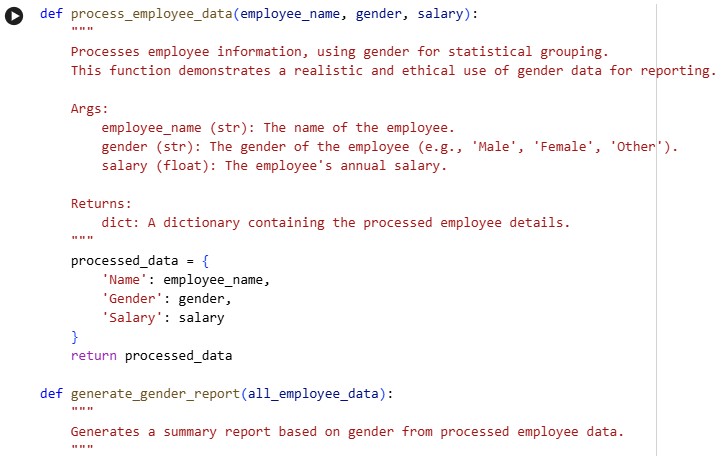
**Task 5: Inclusiveness and Ethical Variable Design Prompt:**

Write a simple and realistic Python program that processes employee

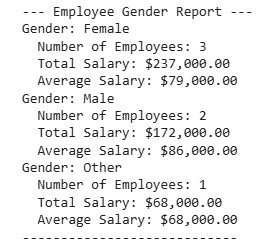
|  |  |
| --- | --- |
| information including name, gender, and salary. The program should use | |
| the gender variable in its logic and display a final output based on the |  |

processed details.

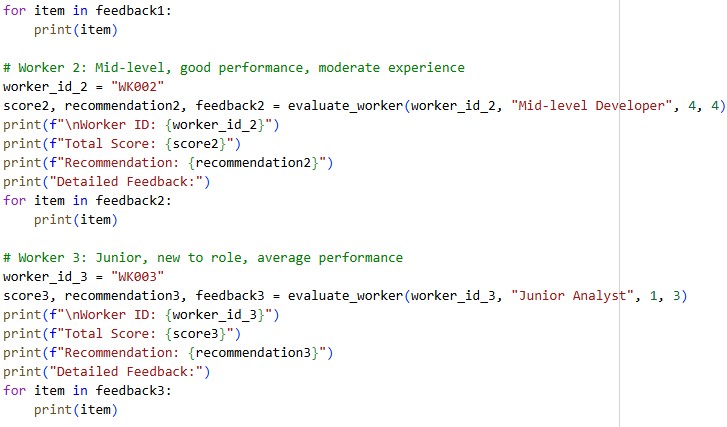
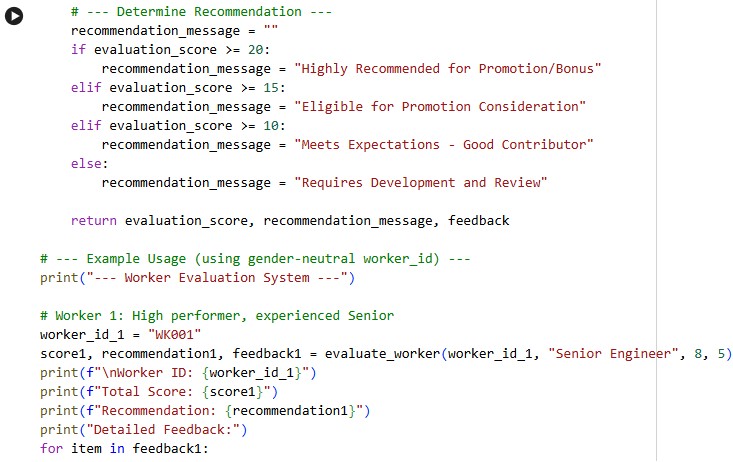
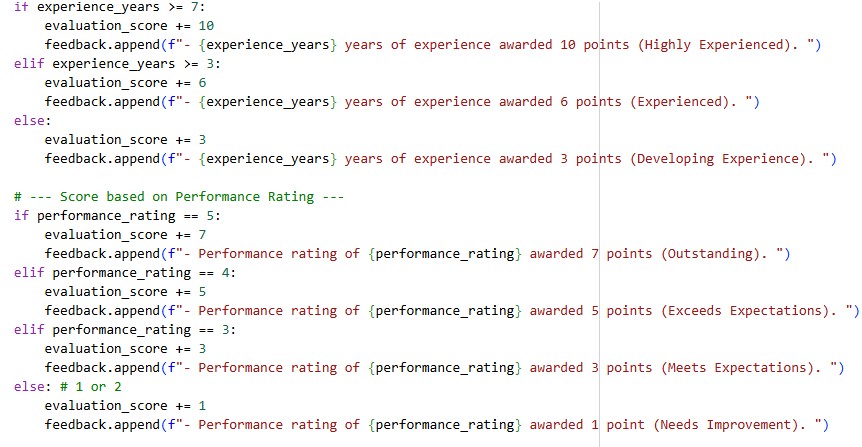
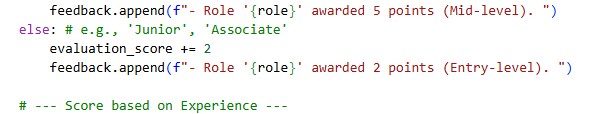
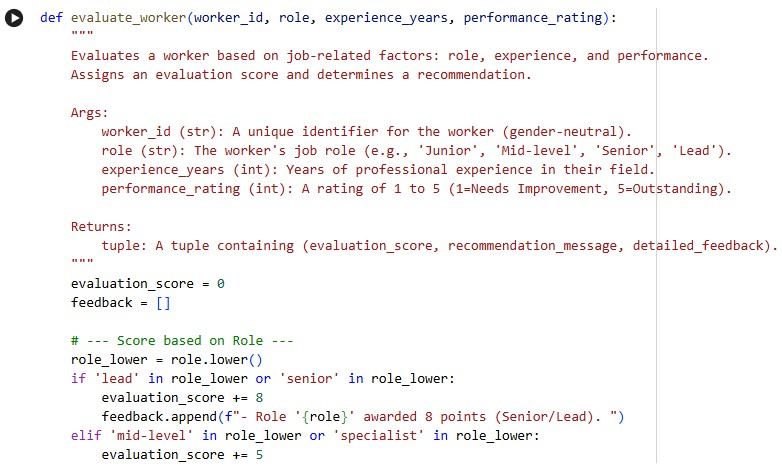
**Code:**



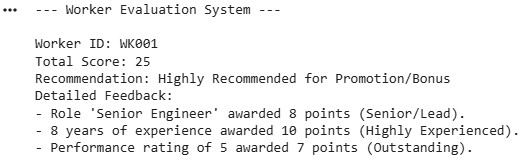
**Prompt:**



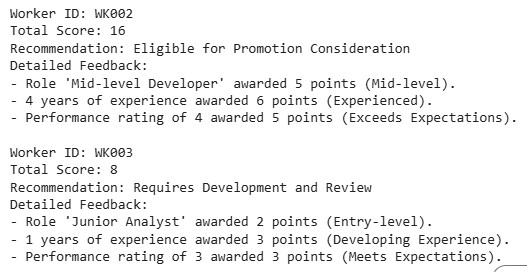
Write a Python program that works with employee details. Use simple and gender-neutral variable names. Do not use gender or identity in the program decisions unless you really need to. The program should make decisions only based on job things like role, performance, or experience.



**Output:**



**Explanation:**



The first code uses gender only for reporting and grouping salaries, not for deciding how good an employee is. It collects names, gender, and salary, then shows how many employees are in each gender group and their average salary. The second code does not use gender at all and instead focuses on fair job factors like role, experience, and performance. It gives each worker a score and recommendation based only on work-related things. So, the first code uses gender just for statistics, while the second code avoids gender and makes decisions based only on job skills and performance.