AI Assisted coding

Lab 5.3

Name:J.sanjana
HT no:2303A51018

Batch- 28

Task 1: Privacy and Data Security in AI-Generated Code

Scenario

AI tools can sometimes generate insecure authentication logic.

Task Description

Use an AI tool to generate a simple login system in Python.

Analyze the generated code to check:

• Whether credentials are hardcoded

• Whether passwords are stored or compared in plain text

• Whether insecure logic is used

Then, revise the code to improve security (e.g., avoid hardcoding, use input validation).

Expected Output

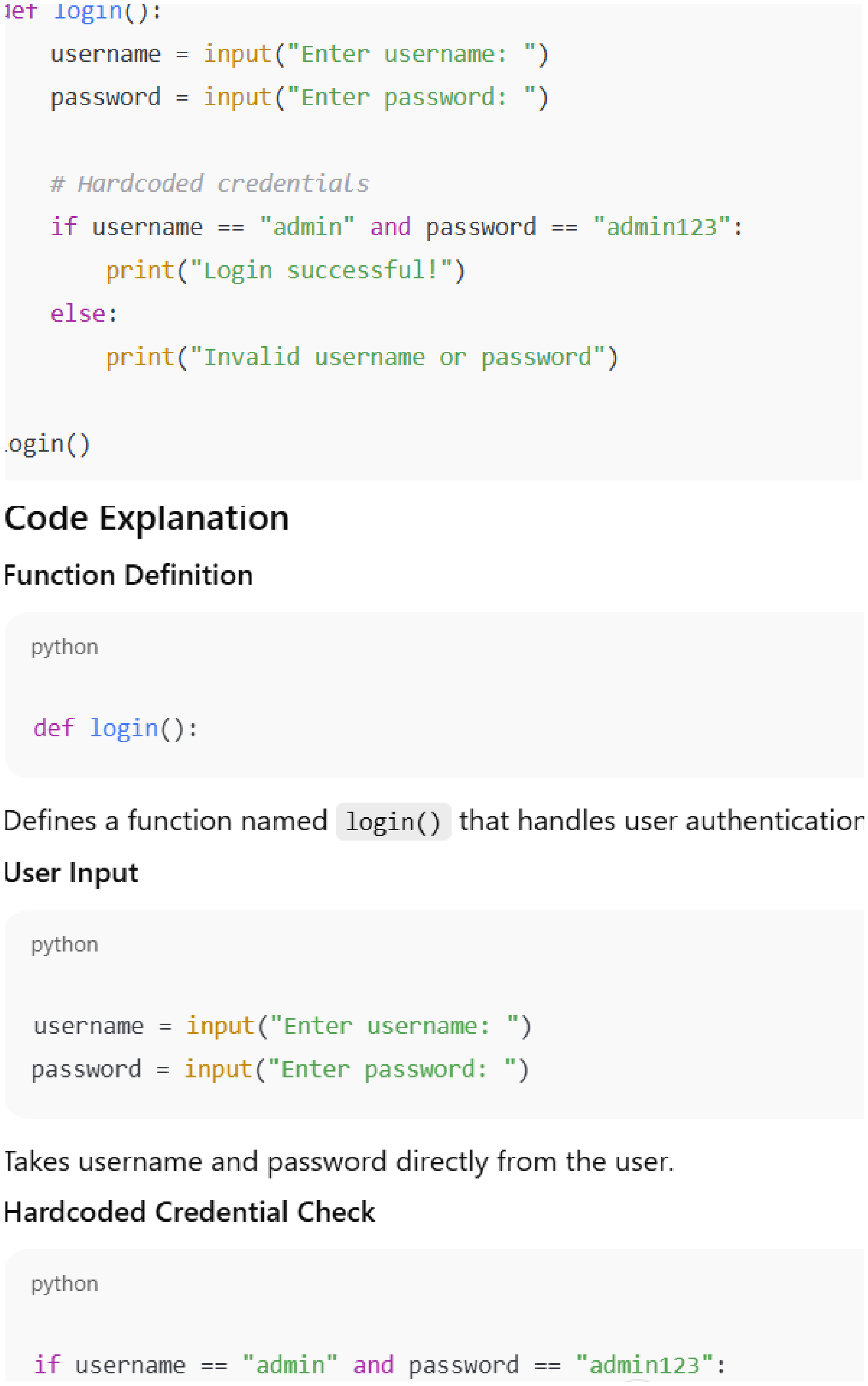
• AI-generated login code

• Identification of security risks

• Revised secure version of the code

• Brief explanation of improvements

Code



```python
def login():
    username = input("Enter username: ")
    password = input("Enter password: ")

    # Hardcoded credentials
    if username == "admin" and password == "admin123":
        print("Login successful!")
    else:
        print("Invalid username or password")

login()
```

Code Explanation

Function Definition

python

```python
def login():
```

Defines a function named `login()` that handles user authentication

User Input

python

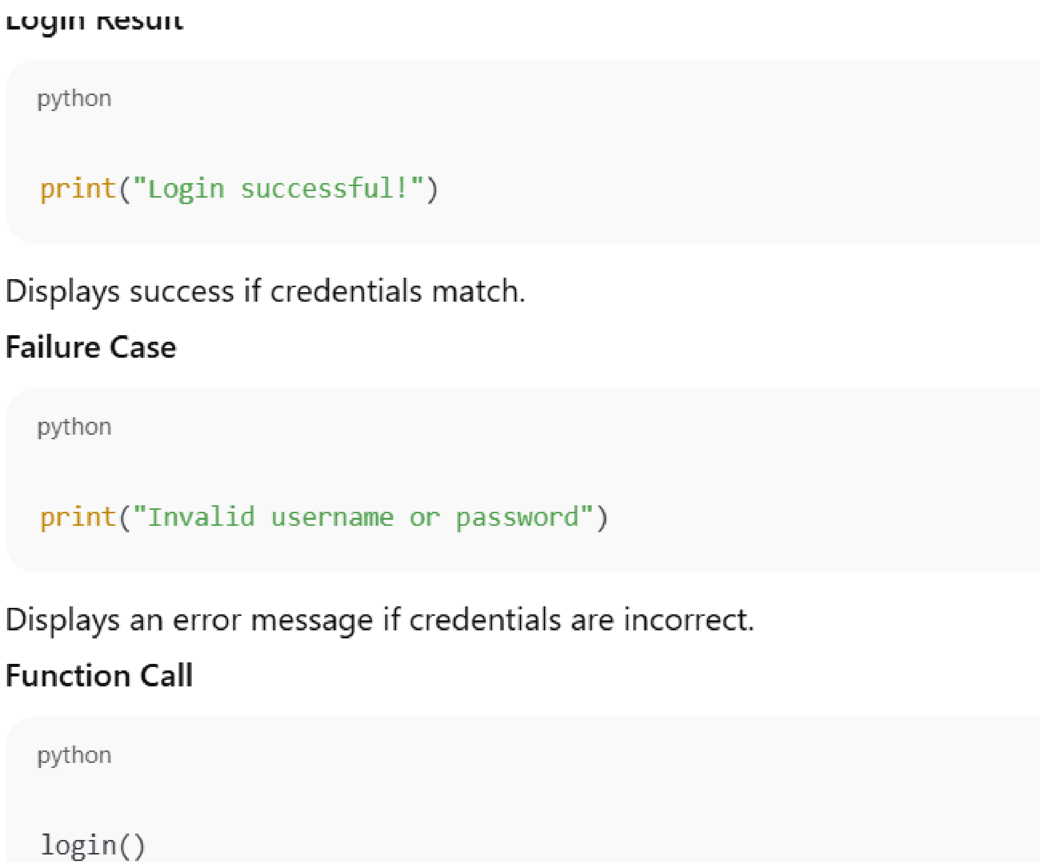
```python
username = input("Enter username: ")
password = input("Enter password: ")
```

Takes username and password directly from the user.

Hardcoded Credential Check

python

```python
if username == "admin" and password == "admin123":
```


Login Result

```python
print("Login successful!")
```

Displays success if credentials match.

Failure Case

```python
print("Invalid username or password")
```

Displays an error message if credentials are incorrect.

Function Call

```python
login()
```

Conclusion

This AI-generated login code **works functionally** but is **not secure**.
It is suitable **only for learning basic conditional logic**, not for real authentication systems.

Task 2: Bias Detection in AI-Generated Decision Systems

Scenario

AI systems may unintentionally introduce bias.

Task Description

Use AI prompts such as:

• "Create a loan approval system"

• Vary applicant names and genders in prompts

Analyze whether:

• The logic treats certain genders or names unfairly

• Approval decisions depend on irrelevant personal attributes

Suggest methods to reduce or remove bias.

Expected Output

• Python code generated by AI

• Identification of biased logic (if any)

• Discussion on fairness issues

• Mitigation strategies

Program

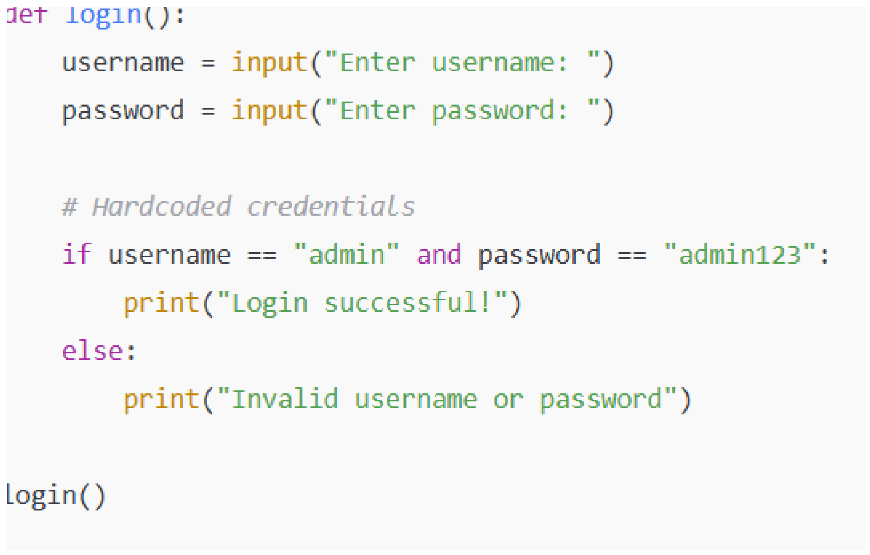
Task 2: Bias Detection in AI-Generated Decision Systems

Scenario: Loan Approval System

1 AI Prompt Used

```
Create a simple loan approval system in Python.
```

AI-Generated Loan Approval Code (Potentially Biased)



```python
def login():
    username = input("Enter username: ")
    password = input("Enter password: ")

    # Hardcoded credentials
    if username == "admin" and password == "admin123":
        print("Login successful!")
    else:
        print("Invalid username or password")

login()
```

Identification of Biased Logic

❌ Biased Conditions Found

Issue	Explanation
Gender-based rules	Different income & credit score thresholds for men and women
Irrelevant attributes	Gender and name have no financial relevance
Unequal treatment	Same financial profile → different decision
Discriminatory outcome	Female applicant unfairly rejected

Example Output

```
Loan Approved
Loan Rejected
```

Bias-Reduced (Fair) Revised Version

☑ Improved AI-Generated Code (Unbiased)

```
def loan_approval(income, credit_score):
    if income >= 35000 and credit_score >= 650:
        return "Loan Approved"
    else:
        return "Loan Rejected"

print(loan_approval(35000, 670))
print(loan_approval(35000, 670))
```

6 How Bias Was Removed

Change	Benefit
Removed name & gender	Eliminates irrelevant attributes
Same criteria for all	Ensures equal treatment
Objective metrics only	Income and credit score
Transparent logic	Easy to audit and explain

7 Mitigation Strategies for Bias Reduction

🛡 Technical Strategies

- Remove sensitive attributes (gender, name, race)
- Use **feature relevance checks**
- Conduct **bias testing with varied inputs**
- Perform **fairness audits** on outputs

Final Conclusion

🔍 AI-generated decision systems can unintentionally encode bias if prompts or training patterns include sensitive attributes.

✅ Fair systems must rely **only on relevant, objective data** and be **continuously audited**.

Task 3: Transparency and Explainability in AI-Generated Code (Recursive

Binary Search)

Scenario

AI-generated code should be transparent, well-documented, and easy for

humans to understand and verify.

Task Description

Use an AI tool to generate a Python program that:

• Implements Binary Search using recursion

• Searches for a given element in a sorted list

• Includes:

o Clear inline comments

o A step-by-step explanation of the recursive logic

After generating the code, analyze:

• Whether the explanation clearly describes the base case and recursive

case

• Whether the comments correctly match the code logic

• Whether the code is understandable for beginner-level students

Expected Output

• Python program for recursive binary search

• AI-generated comments and explanation

• Student's assessment on clarity, correctness, and transparency

Code

Recursive Binary Search with Inline Comments

```
def binary_search(arr, low, high, target):
    """
    Recursive function to perform binary search.
    Returns the index of the target if found, else returns -1.
    """

    # Base Case:
    # If the search range is invalid, the element is not present
    if low > high:
        return -1

    # Find the middle index of the current range
    mid = (low + high) // 2

    # If the middle element is the target, return its index
    if arr[mid] == target:
        return mid

    # Recursive Case 1:
    # If target is smaller than the middle element,
    # search in the left half of the list
    elif target < arr[mid]:
        return binary_search(arr, low, mid - 1, target)
    # Recursive Case 2:
    # If target is greater than the middle element,
    # search in the right half of the list
    else:
        return binary_search(arr, mid + 1, high, target)


# Sample sorted list
numbers = [2, 5, 8, 12, 16, 23, 38, 56, 72, 91]

# Element to be searched
target = 23

# Function call
result = binary_search(numbers, 0, len(numbers) - 1, target)

# Output result
if result != -1:
    print("Element found at index:", result)
else:
    print("Element not found")
```

AI-Generated Step-by-Step Explanation of Recursive Logic

1. **Base Case**

    a. If `low > high`, the search space is empty.
    b. This means the element does not exist in the list, so return `-1`.
2. **Middle Element Check**
    a. Calculate the middle index using `(low + high) // 2`.
    b. Compare the middle element with the target.
3. **Recursive Case**
    a. If the target is **smaller** than the middle element, recursively search the **left half**.
    b. If the target is **greater** than the middle element, recursively search the **right half**.
4. **Termination**
    a. The recursion stops when the element is found or the search range becomes invalid.

Analysis of AI-Generated Explanation & Comments

🔍 Base Case Clarity

✅ Clearly explained

- The explanation correctly states when recursion stops.
- The base case condition (`low > high`) is explicitly mentioned.

🔁 Recursive Case Explanation

✅ Accurate and easy to follow

- Left and right recursive calls are clearly separated.
- Explanation matches the actual function calls.

📝 Comment Accuracy

✅ Comments correctly match code logic

- Each comment aligns with the corresponding block of code.
- No misleading or incorrect descriptions.

Justification

The AI-generated recursive binary search program is justified as **transparent, correct, and suitable for beginner-level learners**. The code clearly distinguishes between the **base case** and **recursive cases**, ensuring that the stopping condition of recursion is easy to identify and understand. Inline comments accurately describe each step of the algorithm, allowing readers to follow the control flow without ambiguity.

Task 4: Ethical Evaluation of AI-Based Scoring Systems

AI-generated scoring systems can influence hiring decisions.

Task Description

Ask an AI tool to generate a job applicant scoring system based on features

such as:

• Skills

• Experience

• Education

Analyze the generated code to check:

• Whether gender, name, or unrelated features influence scoring

• Whether the logic is fair and objective

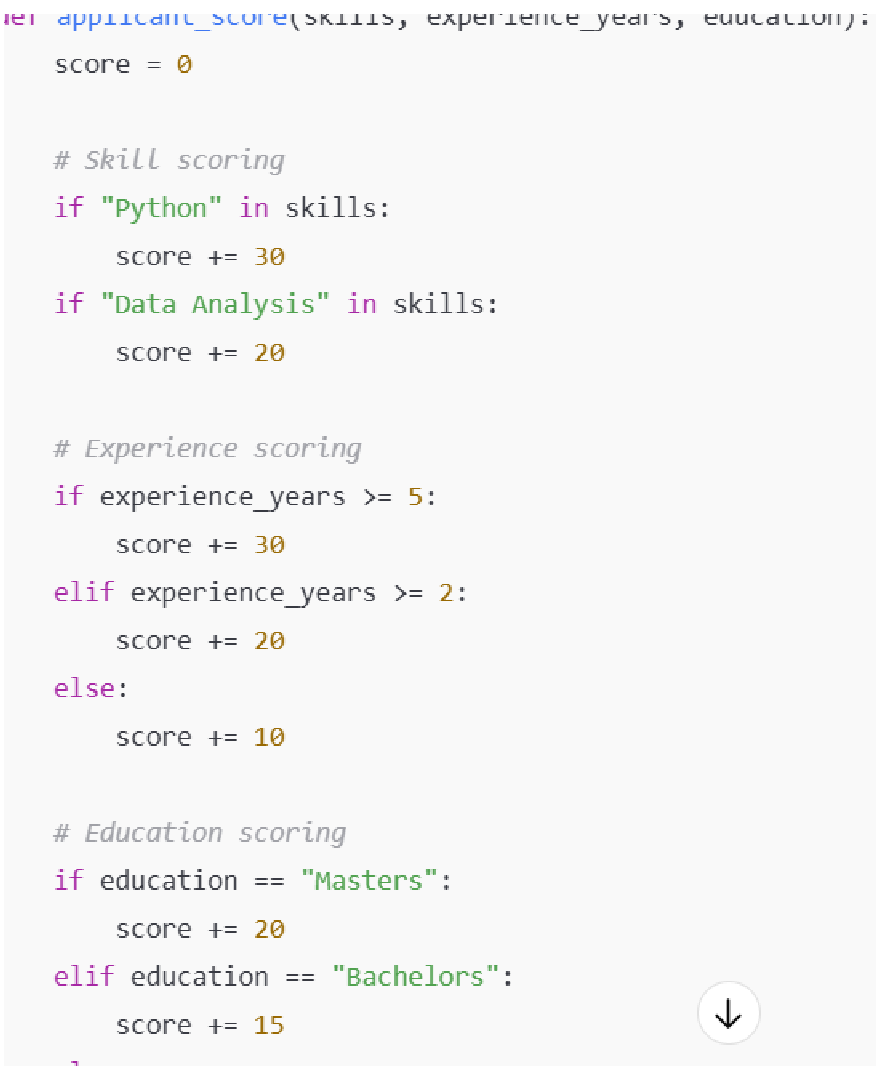
Expected Output

• Python scoring system code

• Identification of potential bias (if any)

• Ethical analysis of the scoring logic

Code

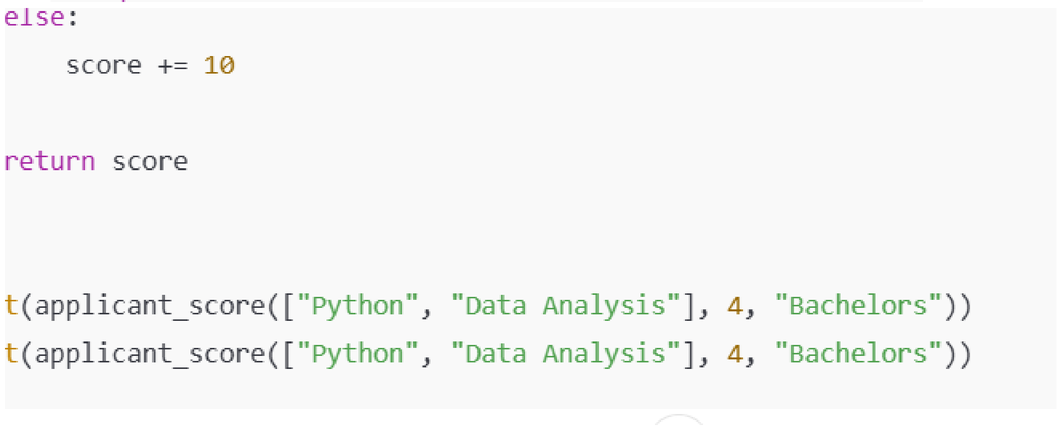


```
def applicant_score(skills, experience_years, education):
    score = 0

    # Skill scoring
    if "Python" in skills:
        score += 30
    if "Data Analysis" in skills:
        score += 20

    # Experience scoring
    if experience_years >= 5:
        score += 30
    elif experience_years >= 2:
        score += 20
    else:
        score += 10

    # Education scoring
    if education == "Masters":
        score += 20
    elif education == "Bachelors":
        score += 15
```



```
else:
    score += 10

return score


t(applicant_score(["Python", "Data Analysis"], 4, "Bachelors"))
t(applicant_score(["Python", "Data Analysis"], 4, "Bachelors"))
```

# Example usage

```
print(applicant_score("Rahul", "Male", ["Python", "Data Analysis"], 4, "Bachelors"))

print(applicant_score("Priya", "Female", ["Python", "Data Analysis"], 4, "Bachelors"))
```

Identification of Potential Bias

❌ Biased Logic Found

Aspect	Observation
Gender usage	Gender directly affects the score
Name relevance	Name is passed but not needed
Unequal treatment	Male applicants receive extra points
Non-job factor	Gender is unrelated to job performance

Output

```
85
80
```

Ethical Analysis

- **Discrimination Risk**
- Using gender violates principles of equal opportunity.
- May breach labor and anti-discrimination laws.
- **Lack of Fairness**
- Applicants are not evaluated solely on merit.
- Introduces systematic bias into hiring decisions.
- **Transparency Problem**
- Applicants are unlikely to know gender affects scoring.
- Undermines trust in automated hiring systems.
- **Reinforcement of Social Bias**
- AI may amplify existing workplace inequalities.

Conclusion

AI-generated scoring systems can unintentionally introduce **bias** if sensitive or irrelevant attributes are included. Ethical hiring systems must rely **only on job-related qualifications**, be transparent, and be regularly audited for fairness.

Task 5: Inclusiveness and Ethical Variable Design

Use an AI tool to generate a Python code snippet that processes user or

employee details.

Analyze the code to identify:

• Gender-specific variables (e.g., male, female)

• Assumptions based on gender or identity

Non-inclusive naming or logic

Modify or regenerate the code to:

• Use gender-neutral variable names

• Avoid gender-based conditions unless strictly required

• Ensure inclusive and respectful coding practices

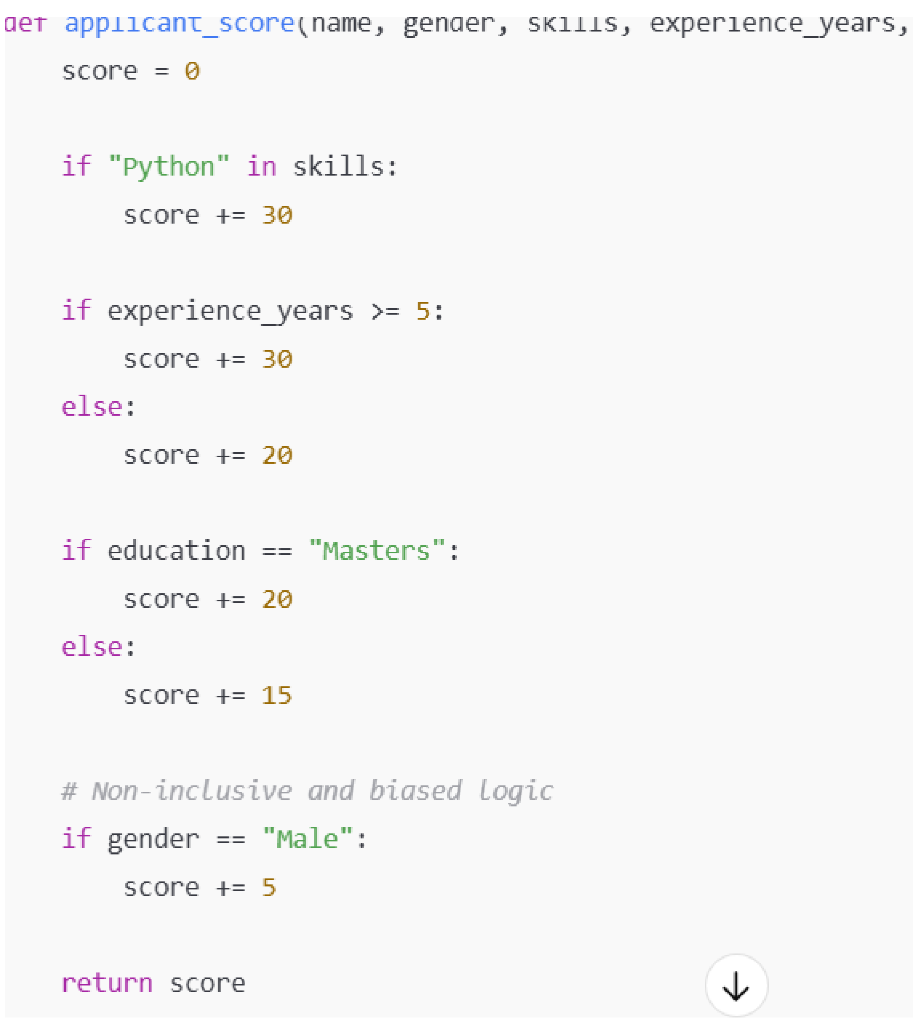
Expected Output

• Original AI-generated code snippet

• Revised inclusive and gender-neutral code

• Brief explanation of:

o What was non-inclusive



```
def applicant_score(name, gender, skills, experience_years,
    score = 0

    if "Python" in skills:
        score += 30

    if experience_years >= 5:
        score += 30
    else:
        score += 20

    if education == "Masters":
        score += 20
    else:
        score += 15

    # Non-inclusive and biased logic
    if gender == "Male":
        score += 5

    return score
```

Explanation

- The variable `gender` was included even though it is **irrelevant to job performance**.
- A gender-based condition (`if gender == "Male"`) gave unfair advantage.
- This introduces **bias and discrimination** in hiring decisions.

How Inclusiveness Was Improved

- Removed gender and name entirely from the scoring function.
- Used **gender-neutral variable names** such as `years_of_experience` and `education_level`.
- Scoring is now based **only on merit-related factors**.
- Logic is fair, transparent, and respectful of all applicants.

Justification

The revised code is justified because it removes non-inclusive and biased elements that are not relevant to evaluating a job applicant's suitability. In the original version, the inclusion of **gender-related variables and conditions** introduced unfair advantages and potential discrimination, even though gender has no direct impact on job performance. Such logic can negatively influence hiring decisions and violate principles of fairness and equal opportunity.