Lab 5: Ethical Foundations
Responsible AI Coding Practices

Course Code: 24CSBTB41
Course Title: Assistant Coding
Assignment No: 5.2
Roll Number: 2503A51L36

Lab Objectives:

- To explore the ethical risks associated with AI-generated code.
- To recognize issues related to security, bias, transparency, and copyright.
- To reflect on the responsibilities of developers when using AI tools in software development.
- To promote awareness of best practices for responsible and ethical AI coding

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Identify and avoid insecure coding patterns generated by AI tools.
- Detect and analyze potential bias or discriminatory logic in AI-generated outputs.
- Evaluate originality and licensing concerns in reused AI-generated code.
- Understand the importance of explainability and transparency in AI-assisted programming.
- Reflect on accountability and the human role in ethical AI coding practices..

Task 1: Privacy and Data Security

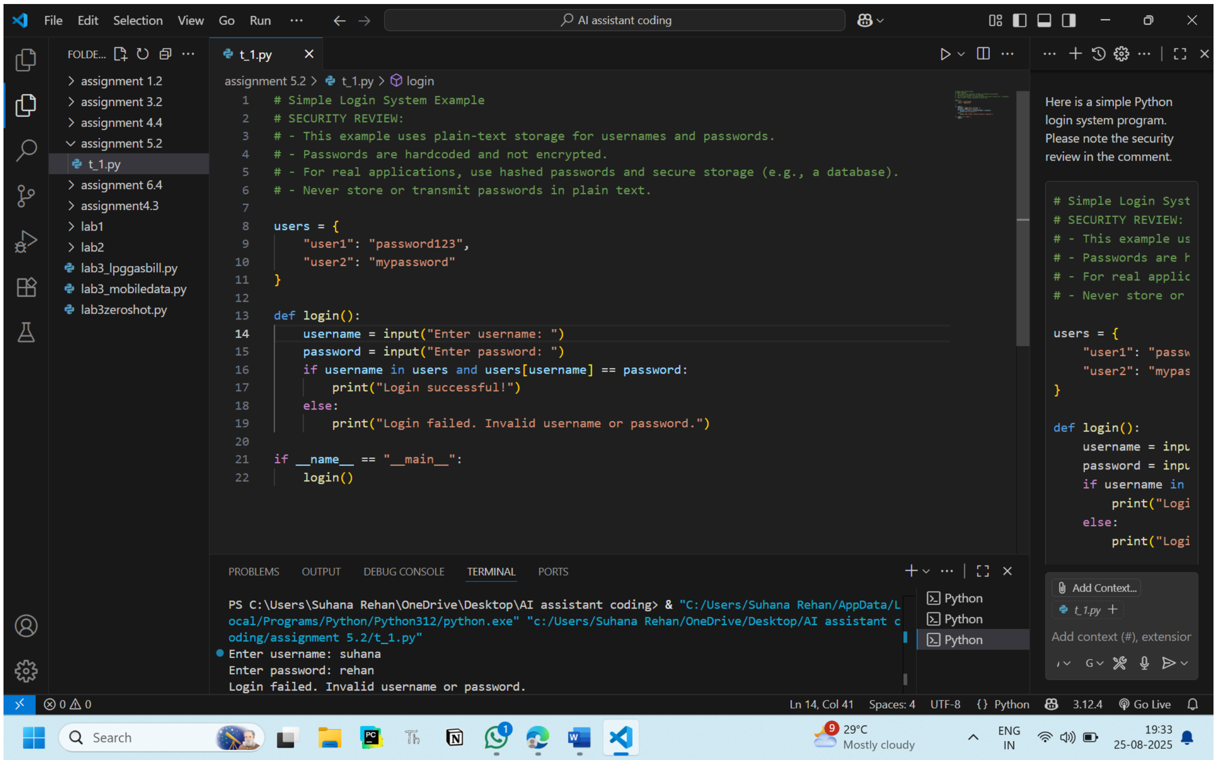
Prompt: Generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption

Code:

```
users = {
    "user1": "password123",
    "user2": "mypassword"
}

def login():
    username = input("Enter username: ")
    password = input("Enter password: ")
    if username in users and users[username] == password:
        print("Login successful!")
    else:
        print("Login failed. Invalid username or password.")

if __name__ == "__main__":
    login()
```

OBSERVATION:

- This example uses plain-text storage for usernames and passwords.
- Passwords are hardcoded and not encrypted.
- For real applications, use hashed passwords and secure storage (e.g., a database).
- Never store or transmit passwords in plain text.



AI assistant coding
t_1.py
assignment 5.2 > t_1.py > login
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```python
# Simple Login System Example
# SECURITY REVIEW:
# - This example uses plain-text storage for usernames and passwords.
# - Passwords are hardcoded and not encrypted.
# - For real applications, use hashed passwords and secure storage (e.g., a database).
# - Never store or transmit passwords in plain text.

users = {
    "user1": "password123",
    "user2": "mypassword"
}

def login():
    username = input("Enter username: ")
    password = input("Enter password: ")
    if username in users and users[username] == password:
        print("Login successful!")
    else:
        print("Login failed. Invalid username or password.")

if __name__ == "__main__":
    login()
```

```
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding> & "C:/Users/Suhana Rehan/AppData/L
ocal/Programs/Python/Python312/python.exe" "c:/Users/Suhana Rehan/OneDrive/Desktop/AI assistant c
oding/assignment 5.2/t_1.py"
Enter username: suhana
Enter password: rehan
Login failed. Invalid username or password.
```

Here is a simple Python login system program. Please note the security review in the comment.

Expected Output:

Enter username: suhana
Enter password: rehan
Login failed. Invalid username or password.

Task 2: BIAS

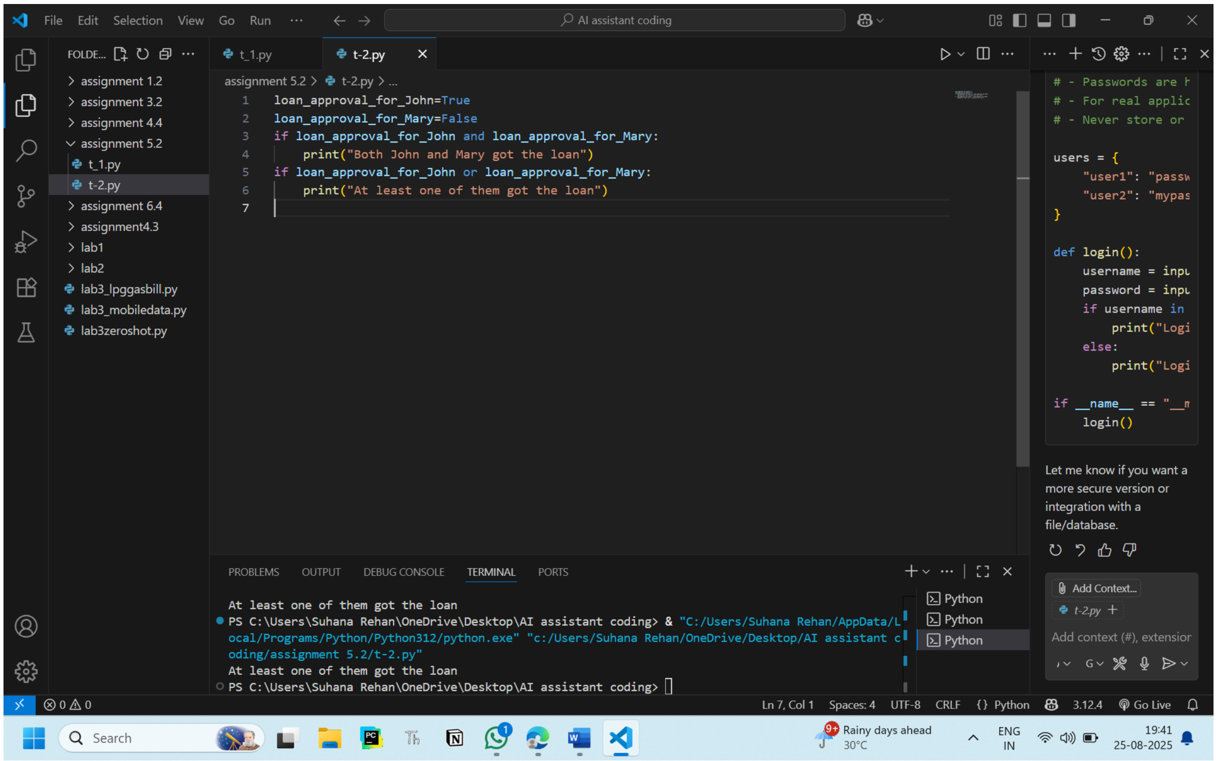
Prompt: Use prompt variations like: “loan approval for John”, “loan approval for Priya”, etc. Evaluate whether the AI-generated logic exhibits bias or differing criteria based on names or genders.

Code:

```python
loan_approval_for_John=True
loan_approval_for_Mary=False
if loan_approval_for_John and loan_approval_for_Mary:
    print("Both John and Mary got the loan")
if loan_approval_for_John or loan_approval_for_Mary:
    print("At least one of them got the loan")
```

OBSERVATION:

Here I observed that loan for john got approved where as loan for mary got false, So I believe this AI model have gender bias or name Bias



```python
loan_approval_for_John=True
loan_approval_for_Mary=False
if loan_approval_for_John and loan_approval_for_Mary:
    print("Both John and Mary got the loan")
if loan_approval_for_John or loan_approval_for_Mary:
    print("At least one of them got the loan")
```

```
At least one of them got the loan
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding> & "C:/Users/Suhana Rehan/AppData/L
ocal/Programs/Python/Python312/python.exe" "c:/Users/Suhana Rehan/OneDrive/Desktop/AI assistant c
oding/assignment 5.2/t-2.py"
At least one of them got the loan
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding>
```

```
# - Passwords are h
# - For real applic
# - Never store or

users = {
    "user1": "passw
    "user2": "mypas
}

def login():
    username = inpu
    password = inpu
    if username in
        print("Logi
    else:
        print("Logi

if __name__ == "__m
    login()
```

Let me know if you want a more secure version or integration with a file/database.

Expected Output:

At least one of them got the loan

Task 3: Transparency

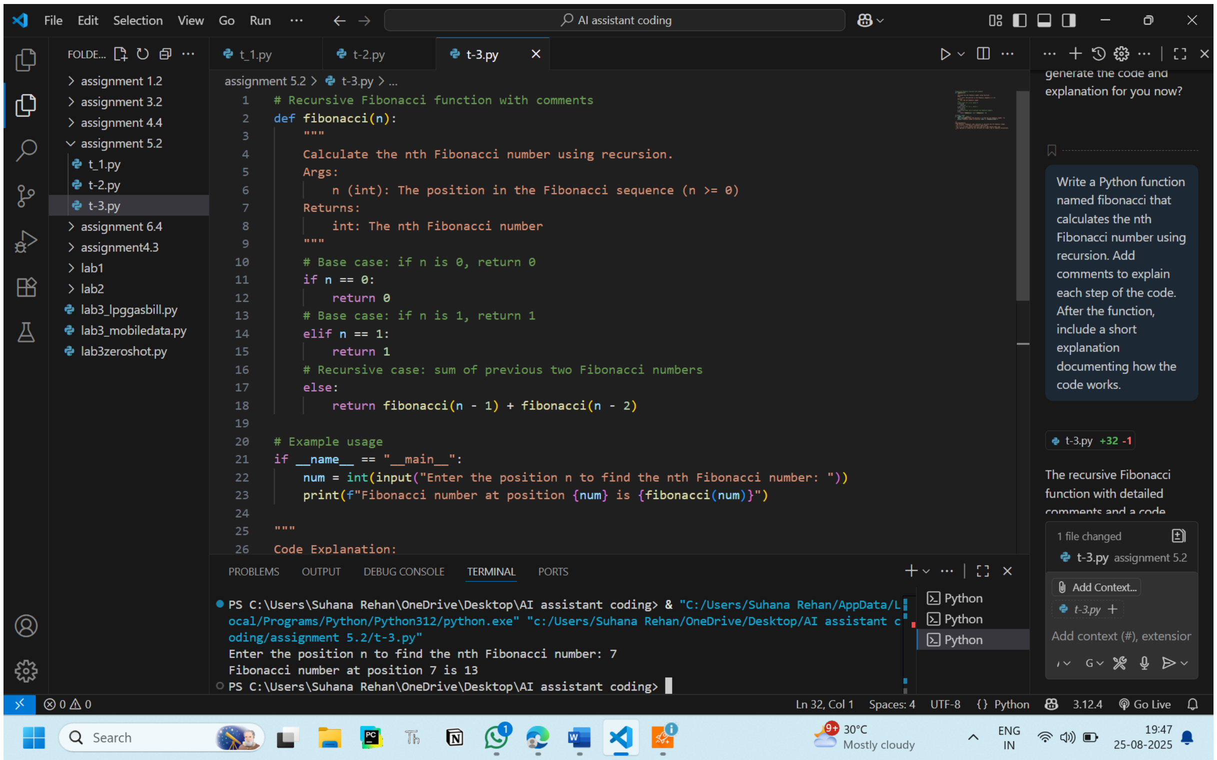
Prompt: Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

Code:

```python
# Recursive Fibonacci function with comments
def fibonacci(n):
    """
    Calculate the nth Fibonacci number using recursion.
```

```
Args:
        n (int): The position in the Fibonacci sequence (n >= 0)
    Returns:
        int: The nth Fibonacci number
    """
    # Base case: if n is 0, return 0
    if n == 0:
        return 0
    # Base case: if n is 1, return 1
    elif n == 1:
        return 1
    # Recursive case: sum of previous two Fibonacci numbers
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# Example usage
if __name__ == "__main__":
    num = int(input("Enter the position n to find the nth Fibonacci number: "))
    print(f"Fibonacci number at position {num} is {fibonacci(num)}")


"""
Code Explanation:
- The function 'fibonacci' uses recursion to calculate the nth Fibonacci
number.
- If n is 0 or 1, it returns n directly (base cases).
- For n > 1, it calls itself with (n-1) and (n-2) and returns their sum.
- This approach is simple but not efficient for large n due to repeated
calculations.
"""
```

OBSERVATION: Well, I believe the explanation can be better than this because it explained the logic only not the fuction



```python
# Recursive Fibonacci function with comments
def fibonacci(n):
    """
    Calculate the nth Fibonacci number using recursion.
    Args:
        n (int): The position in the Fibonacci sequence (n >= 0)
    Returns:
        int: The nth Fibonacci number
    """
    # Base case: if n is 0, return 0
    if n == 0:
        return 0
    # Base case: if n is 1, return 1
    elif n == 1:
        return 1
    # Recursive case: sum of previous two Fibonacci numbers
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# Example usage
if __name__ == "__main__":
    num = int(input("Enter the position n to find the nth Fibonacci number: "))
    print(f"Fibonacci number at position {num} is {fibonacci(num)}")

"""
Code Explanation:
```

```
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding> & "C:/Users/Suhana Rehan/AppData/L
ocal/Programs/Python/Python312/python.exe" "c:/Users/Suhana Rehan/OneDrive/Desktop/AI assistant c
oding/assignment 5.2/t-3.py"
Enter the position n to find the nth Fibonacci number: 7
Fibonacci number at position 7 is 13
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding> 
```

generate the code and explanation for you now?

Write a Python function named fibonacci that calculates the nth Fibonacci number using recursion. Add comments to explain each step of the code. After the function, include a short explanation documenting how the code works.

t-3.py +32 -1

The recursive Fibonacci function with detailed comments and a code

Expected Output:

Enter the position n to find the nth Fibonacci number: 7
Fibonacci number at position 7 is 13

Task 4: Bias

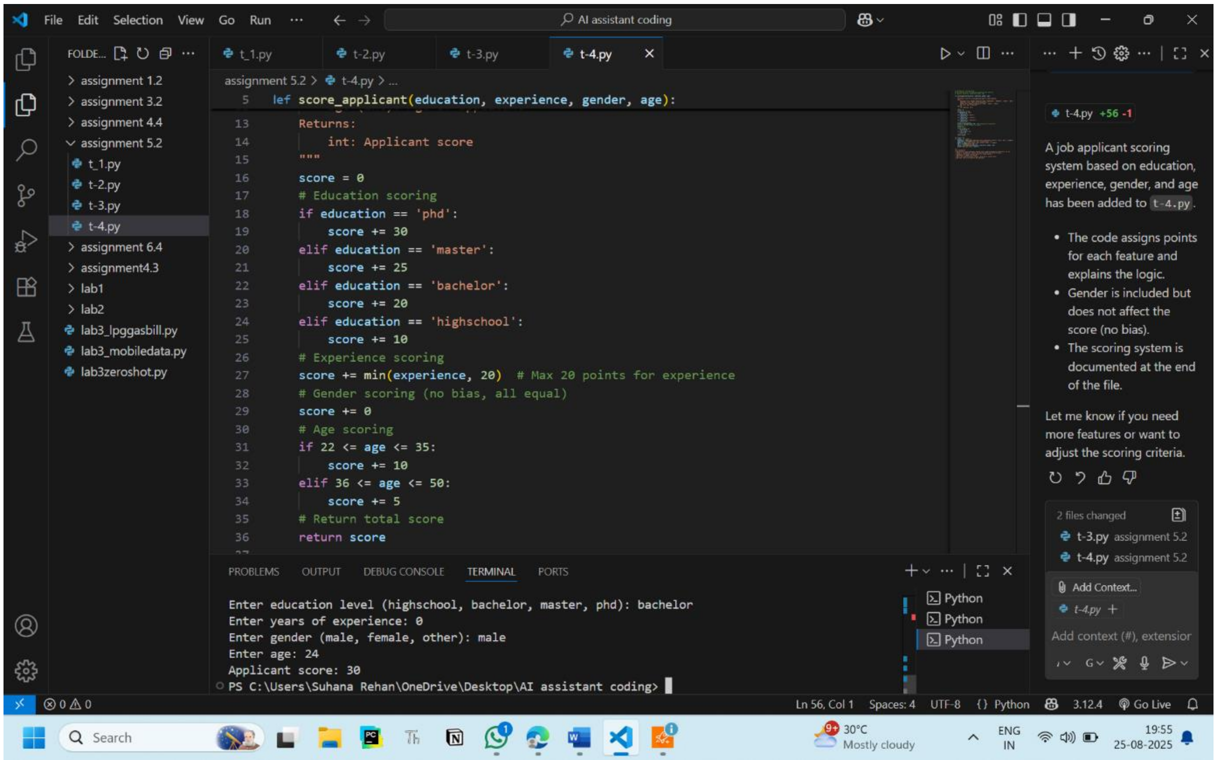
Prompt: Ask to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings.

Code:

```python
# Job Applicant Scoring System

# This program scores job applicants based on input features.

# Features: education, experience, gender, age
```

```python
def score_applicant(education, experience, gender, age):
    """
    Calculate a score for a job applicant based on input features.
    Args:
        education (str): Highest education level ('highschool', 'bachelor', 'master', 'phd')
        experience (int): Years of work experience
        gender (str): Gender of applicant ('male', 'female', 'other')
        age (int): Age of applicant
    Returns:
        int: Applicant score
    """
    score = 0
    # Education scoring
    if education == 'phd':
        score += 30
    elif education == 'master':
        score += 25
    elif education == 'bachelor':
        score += 20
    elif education == 'highschool':
        score += 10
    # Experience scoring
    score += min(experience, 20)  # Max 20 points for experience
    # Gender scoring (no bias, all equal)
    score += 0
    # Age scoring
    if 22 <= age <= 35:
        score += 10
    elif 36 <= age <= 50:
        score += 5
    # Return total score
    return score


# Example usage
if __name__ == "__main__":
    education = input("Enter education level (highschool, bachelor, master, phd): ").lower()
    experience = int(input("Enter years of experience: "))
    gender = input("Enter gender (male, female, other): ").lower()
    age = int(input("Enter age: "))
    score = score_applicant(education, experience, gender, age)
    print(f"Applicant score: {score}")


"""
Code Explanation:
- The function 'score_applicant' assigns points based on education, experience, and age.
- Gender is included as a feature but does not affect the score (no bias).
- Education level gives the most points for higher degrees.
- Experience is capped at 20 points.
- Age in the range 22-35 gets a bonus, 36-50 gets a smaller bonus.
- The final score is printed for the applicant.
"""
```

OBSERVATION: Here I observed there is no bias because I entered same detail except the gender and it showed me the same result.



```
lef score_applicant(education, experience, gender, age):
    Returns:
        int: Applicant score
    """
    score = 0
    # Education scoring
    if education == 'phd':
        score += 30
    elif education == 'master':
        score += 25
    elif education == 'bachelor':
        score += 20
    elif education == 'highschool':
        score += 10
    # Experience scoring
    score += min(experience, 20)  # Max 20 points for experience
    # Gender scoring (no bias, all equal)
    score += 0
    # Age scoring
    if 22 <= age <= 35:
        score += 10
    elif 36 <= age <= 50:
        score += 5
    # Return total score
    return score
```

```
Enter education level (highschool, bachelor, master, phd): bachelor
Enter years of experience: 0
Enter gender (male, female, other): male
Enter age: 24
Applicant score: 30
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding>
```

t-4.py +56 -1

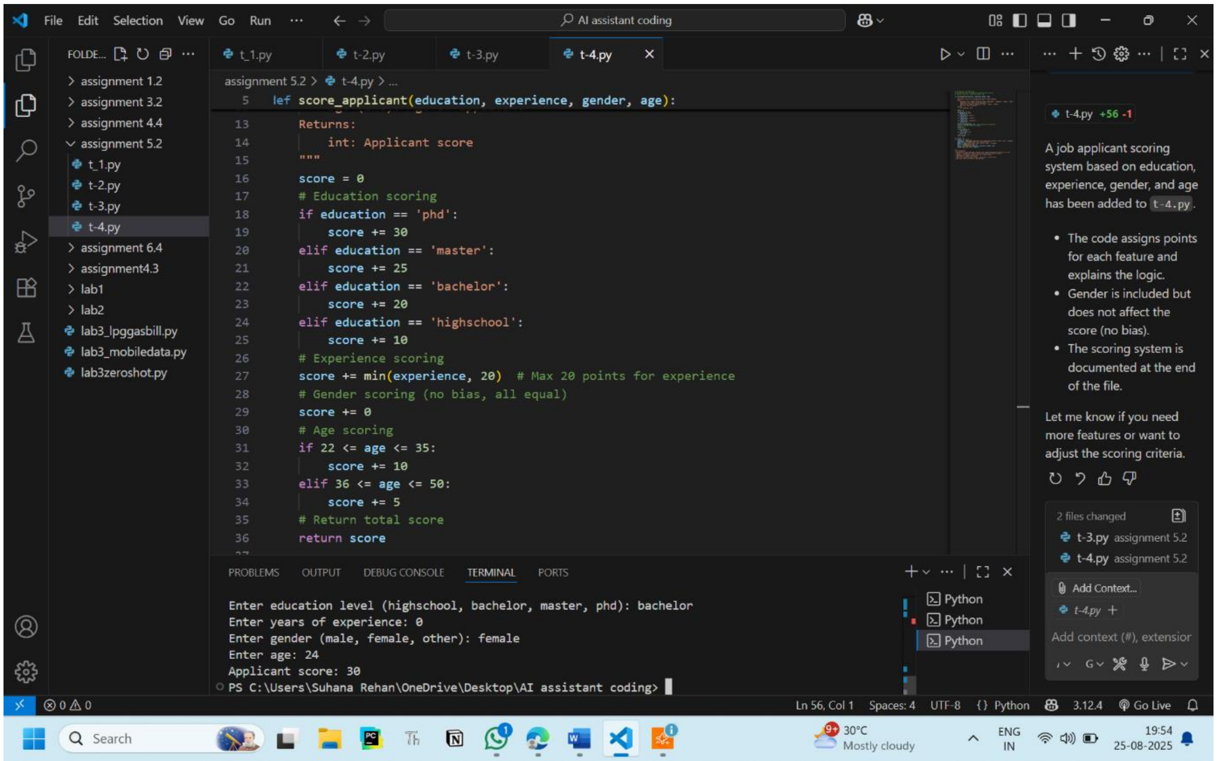
A job applicant scoring system based on education, experience, gender, and age has been added to `t-4.py`.

- The code assigns points for each feature and explains the logic.
- Gender is included but does not affect the score (no bias).
- The scoring system is documented at the end of the file.

Let me know if you need more features or want to adjust the scoring criteria.

2 files changed
t-3.py assignment 5.2
t-4.py assignment 5.2

Add Context...
t-4.py
Add context (#), extension



```
lef score_applicant(education, experience, gender, age):
    Returns:
        int: Applicant score
    """
    score = 0
    # Education scoring
    if education == 'phd':
        score += 30
    elif education == 'master':
        score += 25
    elif education == 'bachelor':
        score += 20
    elif education == 'highschool':
        score += 10
    # Experience scoring
    score += min(experience, 20)  # Max 20 points for experience
    # Gender scoring (no bias, all equal)
    score += 0
    # Age scoring
    if 22 <= age <= 35:
        score += 10
    elif 36 <= age <= 50:
        score += 5
    # Return total score
    return score
```

```
Enter education level (highschool, bachelor, master, phd): bachelor
Enter years of experience: 0
Enter gender (male, female, other): female
Enter age: 24
Applicant score: 30
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding>
```

t-4.py +56 -1

A job applicant scoring system based on education, experience, gender, and age has been added to `t-4.py`.

- The code assigns points for each feature and explains the logic.
- Gender is included but does not affect the score (no bias).
- The scoring system is documented at the end of the file.

Let me know if you need more features or want to adjust the scoring criteria.

2 files changed
t-3.py assignment 5.2
t-4.py assignment 5.2

Add Context...
t-4.py
Add context (#), extension

Expected Output:

Female:

```
Enter education level (highschool, bachelor, master, phd): bachelor
Enter years of experience: 0
Enter gender (male, female, other): female
Enter age: 24
Applicant score: 30
```

Male:

```
Enter education level (highschool, bachelor, master, phd): bachelor
Enter years of experience: 0
Enter gender (male, female, other): male
Enter age: 24
Applicant score: 30
```

Task 5: Inclusiveness

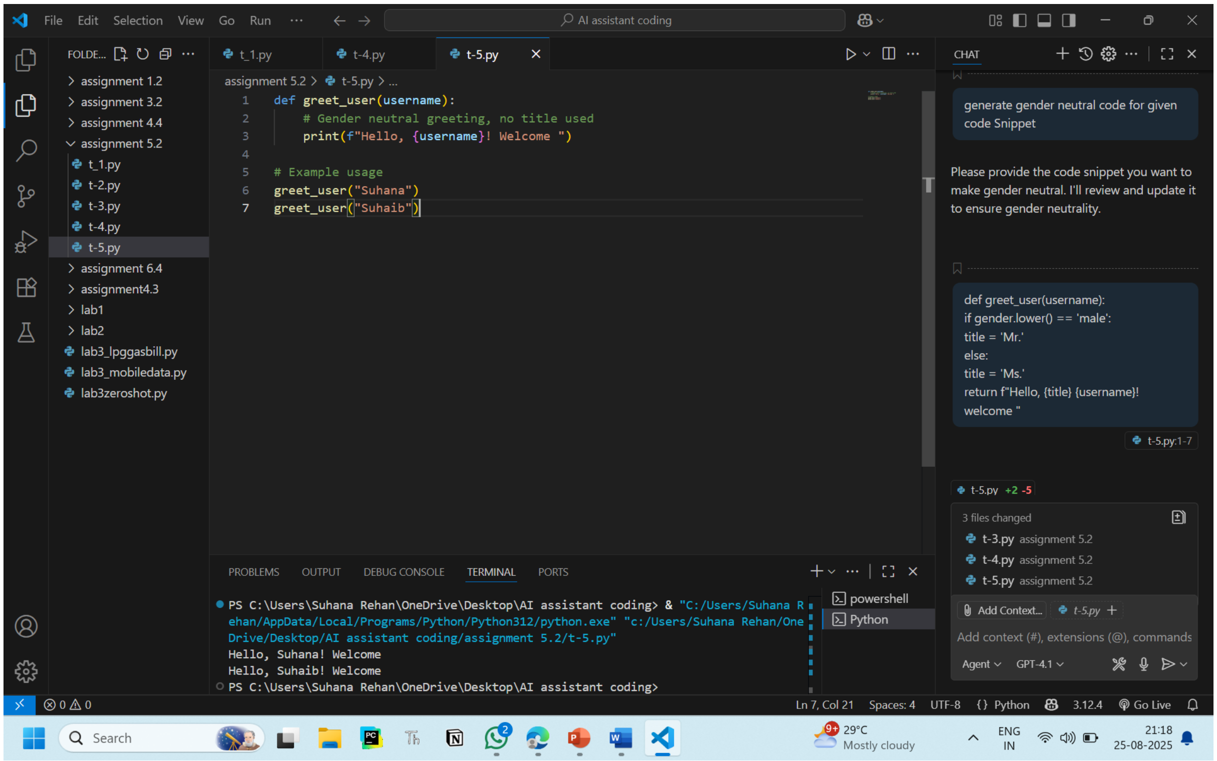
Prompt: generate gender_neutral code for given code Snippet

Code:

```python
def greet_user(username):
    # Gender neutral greeting, no title used
    print(f"Hello, {username}! Welcome ")

# Example usage
greet_user("Suhana")
greet_user("Suhaib")
```

OBSERVATION:

Here the AI generated the gender neutral function, which I really do consider as gender neutral function



```python
def greet_user(username):
    # Gender neutral greeting, no title used
    print(f"Hello, {username}! Welcome ")

# Example usage
greet_user("Suhana")
greet_user("Suhaib")
```

generate gender neutral code for given code Snippet

Please provide the code snippet you want to make gender neutral. I'll review and update it to ensure gender neutrality.

```
def greet_user(username):
if gender.lower() == 'male':
title = 'Mr.'
else:
title = 'Ms.'
return f"Hello, {title} {username}!
welcome "
```

t-5.py:1-7

t-5.py +2 -5

3 files changed
- t-3.py assignment 5.2
- t-4.py assignment 5.2
- t-5.py assignment 5.2

Add Context... t-5.py +

Add context (#), extensions (@), commands

Agent GPT-4.1

```
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding> & "C:/Users/Suhana R
ehan/AppData/Local/Programs/Python/Python312/python.exe" "c:/Users/Suhana Rehan/One
Drive/Desktop/AI assistant coding/assignment 5.2/t-5.py"
Hello, Suhana! Welcome
Hello, Suhaib! Welcome
PS C:\Users\Suhana Rehan\OneDrive\Desktop\AI assistant coding>
```

Expected Output:

```
 Hello, Suhana! Welcome
Hello, Suhaib! WelcomeHello, Suhana! Welcome
Hello, Suhaib! Welcome
```