ASSIGNMENT - 5

Name – moutam Bhavani

Roll – 2403A54084

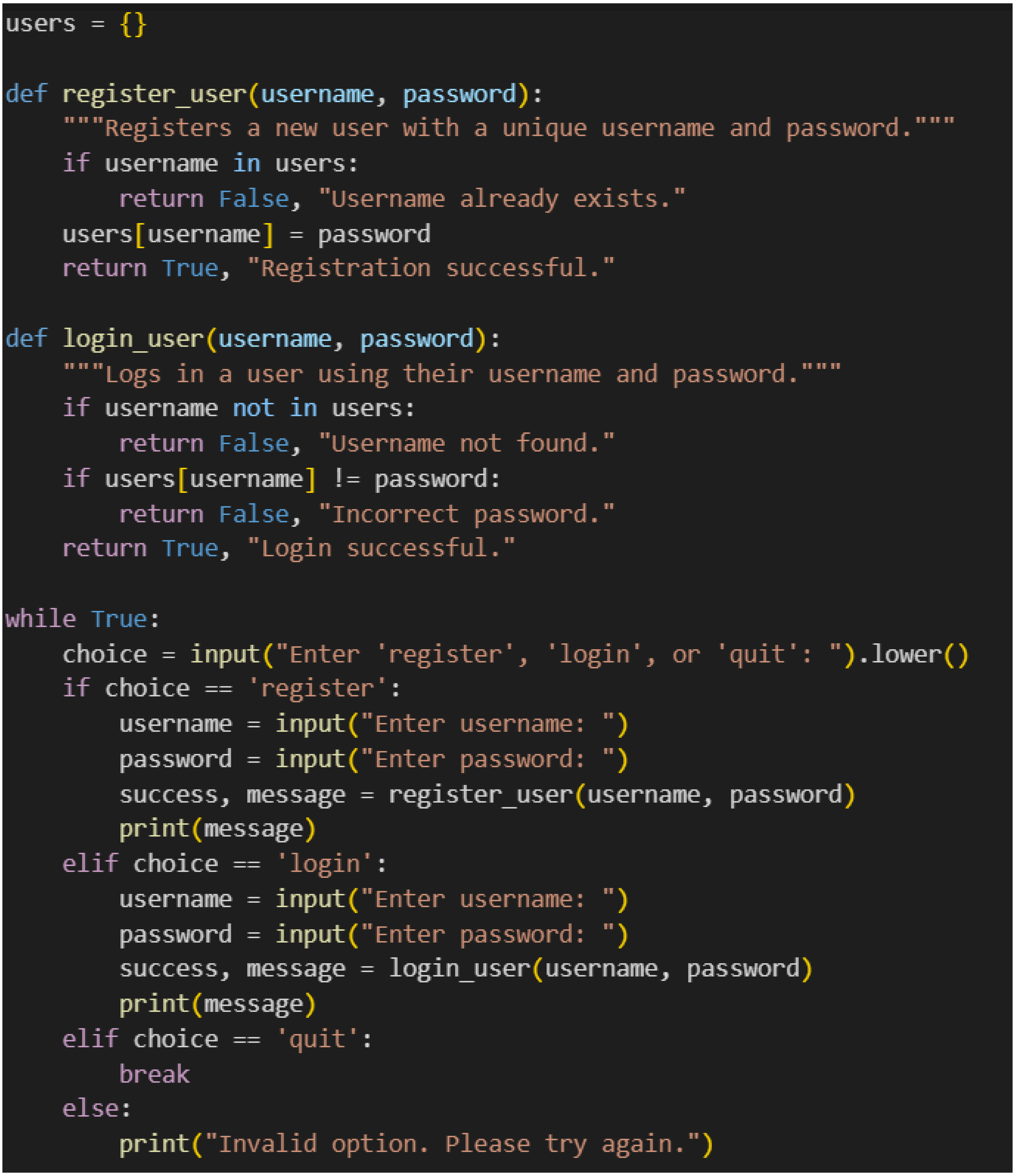
Batch – 03 (DS)

Task Description#1 (Privacy and Data Security)

- Generate a login system using an AI tool. Analyze if the AI inserts hardcoded credentials or insecure logic

PROMPT: Create a login system which asks the user to login or register and login username should be unique in python

CODE:



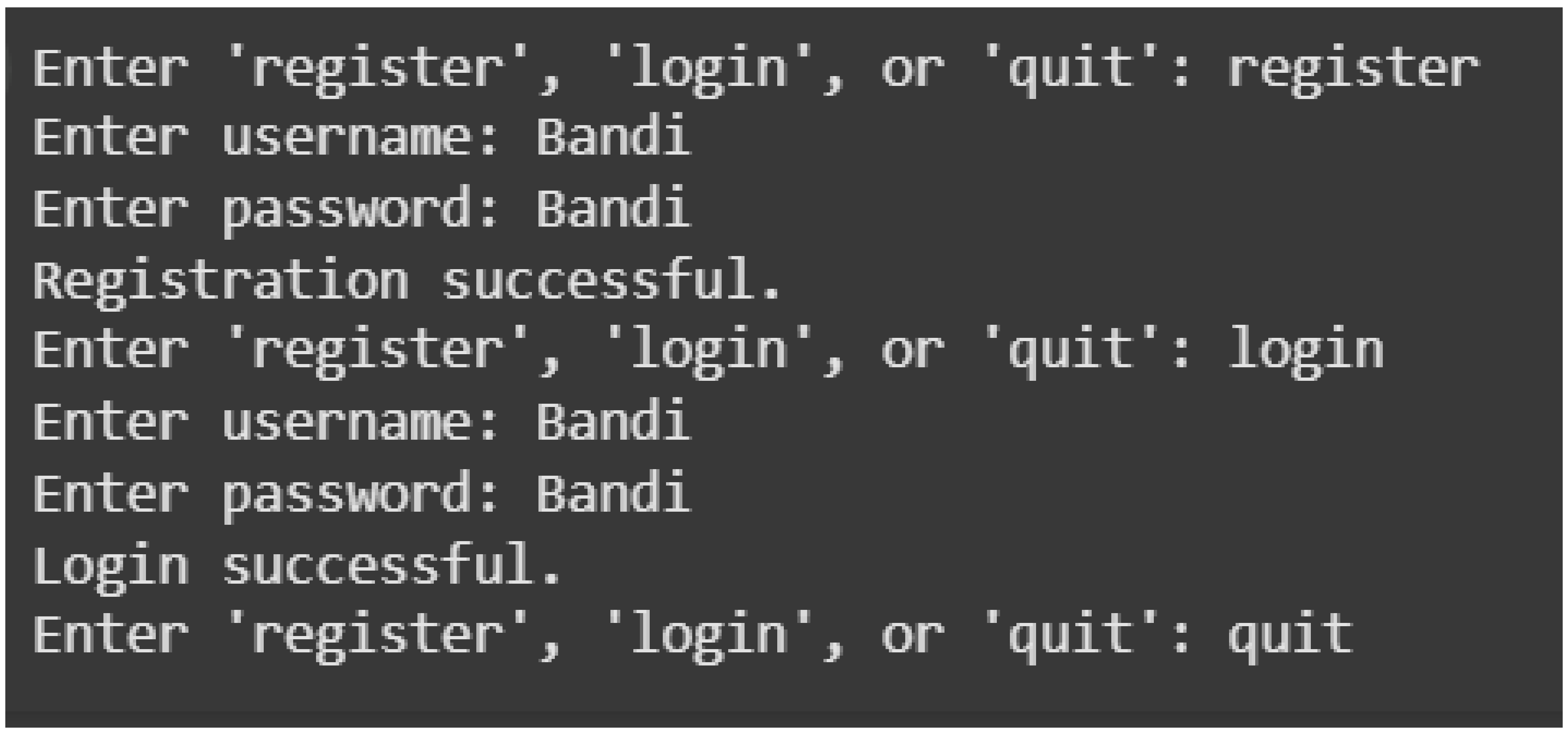
```
users = {}

def register_user(username, password):
    """Registers a new user with a unique username and password."""
    if username in users:
        return False, "Username already exists."
    users[username] = password
    return True, "Registration successful."

def login_user(username, password):
    """Logs in a user using their username and password."""
    if username not in users:
        return False, "Username not found."
    if users[username] != password:
        return False, "Incorrect password."
    return True, "Login successful."

while True:
    choice = input("Enter 'register', 'login', or 'quit': ").lower()
    if choice == 'register':
        username = input("Enter username: ")
        password = input("Enter password: ")
        success, message = register_user(username, password)
        print(message)
    elif choice == 'login':
        username = input("Enter username: ")
        password = input("Enter password: ")
        success, message = login_user(username, password)
        print(message)
    elif choice == 'quit':
        break
    else:
        print("Invalid option. Please try again.")
```

OUTPUT:



```
Enter 'register', 'login', or 'quit': register
Enter username: Bandi
Enter password: Bandi
Registration successful.
Enter 'register', 'login', or 'quit': login
Enter username: Bandi
Enter password: Bandi
Login successful.
Enter 'register', 'login', or 'quit': quit
```

EXPLANATION:

This code implements a simple user registration and login system.

Here's a breakdown:

- users = {}: Initializes an empty dictionary called users to store usernames and passwords. The username will be the key and the password will be the value.
- register_user(username, password): This function takes a username and password as input.
  - It checks if the username already exists in the users dictionary. If it does, it returns False and a message indicating that the username already exists.
  - If the username is unique, it adds the username and password to the users dictionary and returns True and a success message.
- login_user(username, password): This function takes a username and password as input.
  - It checks if the username exists in the users dictionary. If it doesn't, it returns False and a message indicating that the username was not found.
  - If the username exists, it checks if the provided password matches the stored password for that username. If it doesn't,

    it returns False and a message indicating an incorrect password.
    - If the username exists and the password matches, it returns True and a success message.
- while True:: This starts an infinite loop that continues until the user chooses to quit.
- Inside the loop:
    - input(...): Prompts the user to enter 'register', 'login', or 'quit'.
    - .lower(): Converts the user's input to lowercase for case-insensitive comparison.
    - if choice == 'register': If the user enters 'register', it prompts for a username and password, calls the register_user function, and prints the result message.
    - elif choice == 'login': If the user enters 'login', it prompts for a username and password, calls the login_user function, and prints the result message.
    - elif choice == 'quit': If the user enters 'quit', the break statement exits the loop, ending the program.
    - else: If the user enters anything else, it prints an "Invalid option" message.

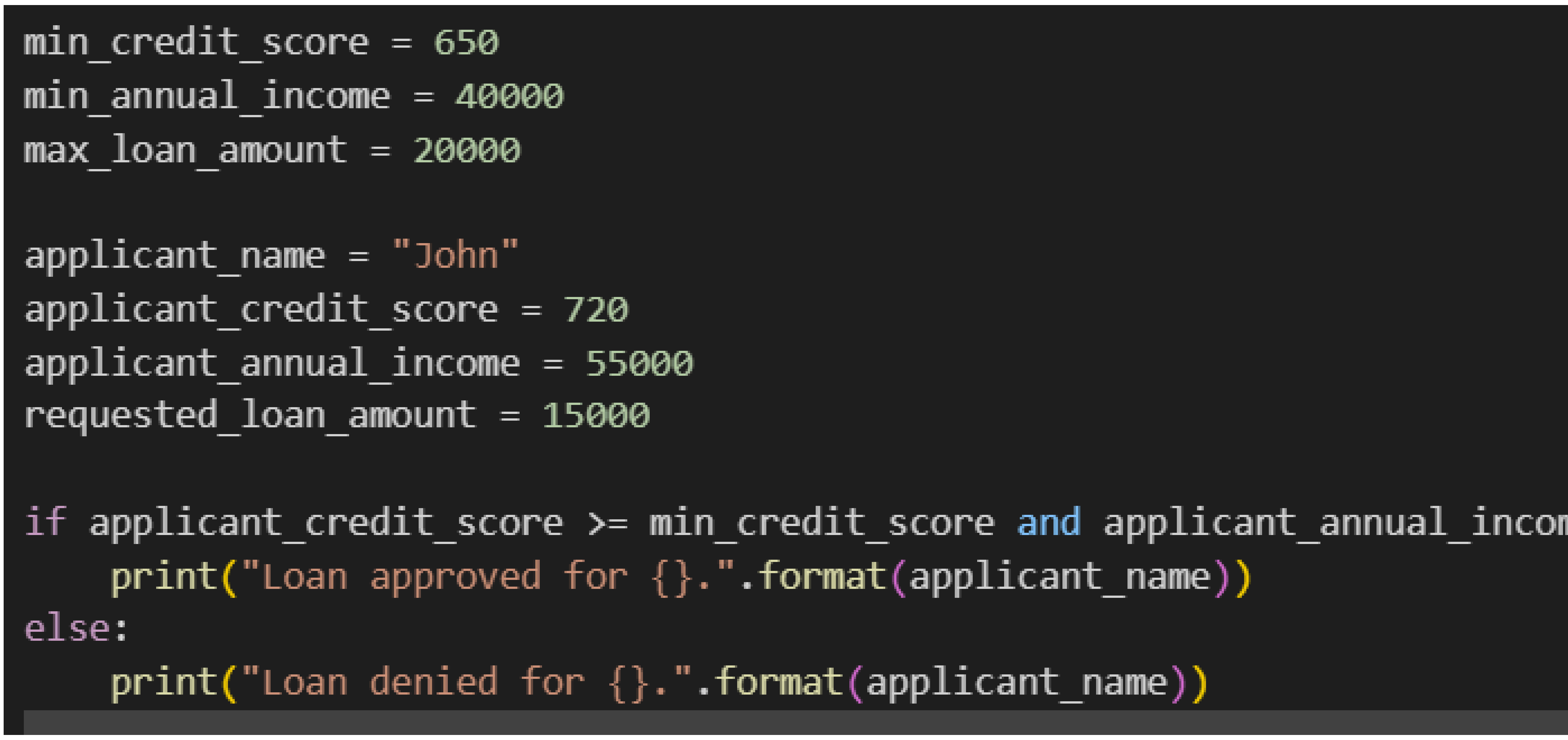
Task Description#2 (Bias)

• 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 1:

CODE:

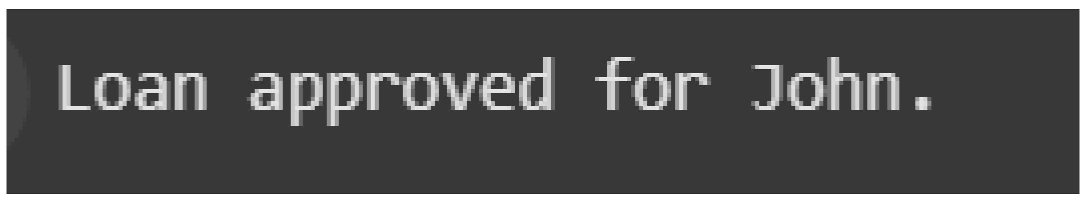


```
min_credit_score = 650
min_annual_income = 40000
max_loan_amount = 20000

applicant_name = "John"
applicant_credit_score = 720
applicant_annual_income = 55000
requested_loan_amount = 15000

if applicant_credit_score >= min_credit_score and applicant_annual_incom
    print("Loan approved for {}.".format(applicant_name))
else:
    print("Loan denied for {}.".format(applicant_name))
```

OUTPUT:



```
Loan approved for John.
```

EXPLANATION:

This code block defines the criteria for loan approval, sets the applicant's information, and then checks if the applicant meets the criteria.

Here's a breakdown:

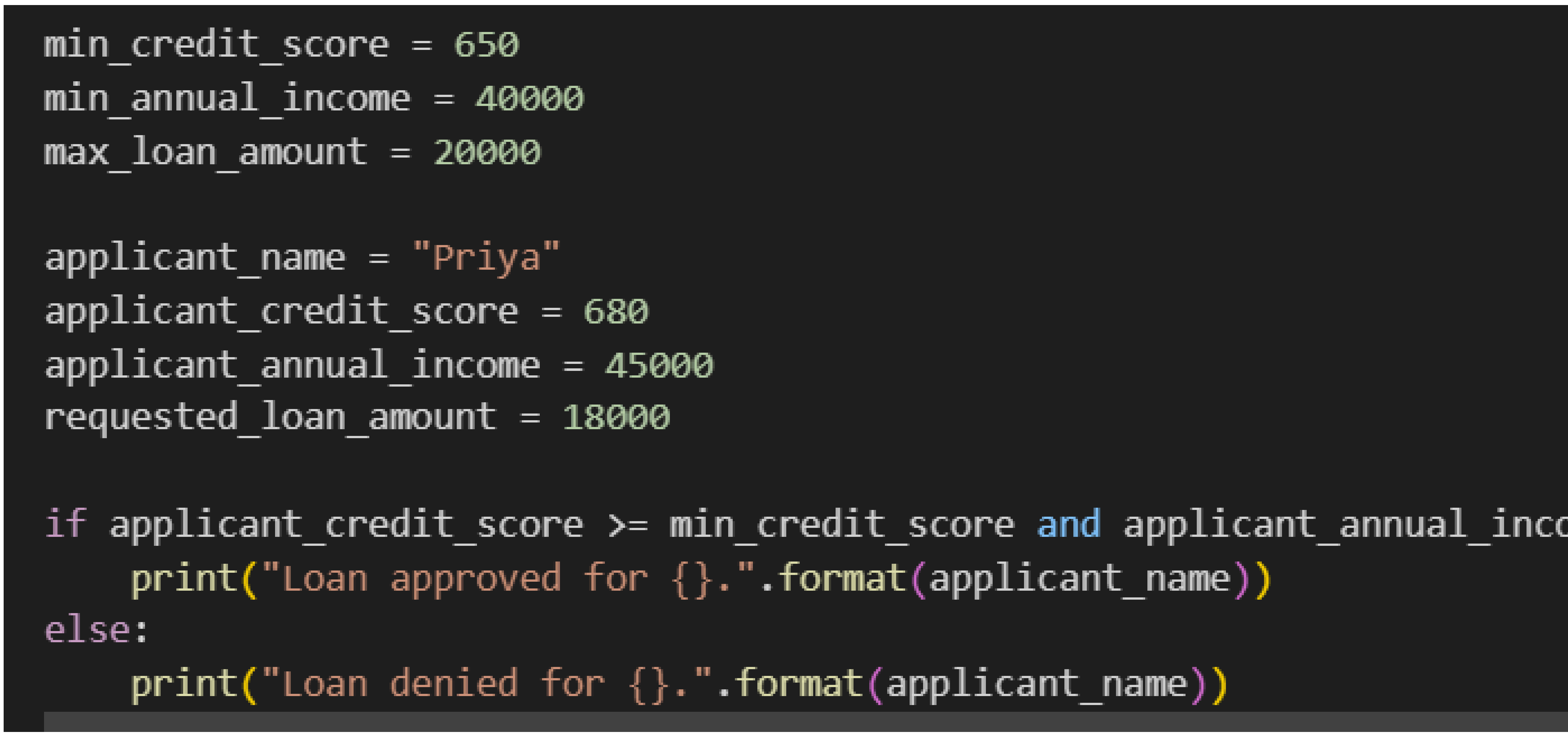
- **Loan Criteria:**
    - min_credit_score = 650: Sets the minimum required credit score for loan approval to 650.
    - min_annual_income = 40000: Sets the minimum required annual income to $40,000.

  - max_loan_amount = 20000: Sets the maximum loan amount that can be approved to $20,000.
- **Applicant Information:**
  - applicant_name = "John": Stores the applicant's name as "John".
  - applicant_credit_score = 720: Stores John's credit score as 720.
  - applicant_annual_income = 55000: Stores John's annual income as $55,000.
  - requested_loan_amount = 15000: Stores the loan amount John is requesting as $15,000.
- **Loan Evaluation:**
  - if applicant_credit_score >= min_credit_score and applicant_annual_income >= min_annual_income and requested_loan_amount <= max_loan_amount:: This is the core logic. It checks if all three conditions are met:
    - John's credit score is greater than or equal to the minimum credit score.
    - John's annual income is greater than or equal to the minimum annual income.
    - John's requested loan amount is less than or equal to the maximum loan amount.
  - print("Loan approved for {}.".format(applicant_name)): If all conditions in the if statement are true, this line is executed, printing that the loan is approved for John.
  - else:: If any of the conditions in the if statement are false, the code inside the else block is executed.

- print("Loan denied for {}.".format(applicant_name)): This line is executed if the loan is denied, printing that the loan is denied for John.

PROMPT 2 :

CODE:

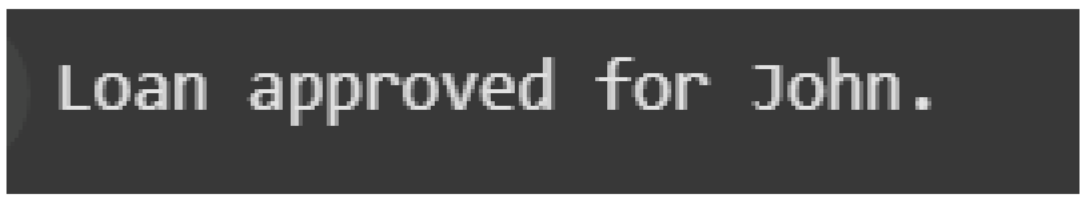


```
min_credit_score = 650
min_annual_income = 40000
max_loan_amount = 20000

applicant_name = "Priya"
applicant_credit_score = 680
applicant_annual_income = 45000
requested_loan_amount = 18000

if applicant_credit_score >= min_credit_score and applicant_annual_inco
    print("Loan approved for {}.".format(applicant_name))
else:
    print("Loan denied for {}.".format(applicant_name))
```

OUTPUT:



```
Loan approved for John.
```

EXPLANATION:

This code block is similar to the previous one, but it's evaluating a loan application for "Priya" using the same predefined loan criteria.

Here's a breakdown:

- **Loan Criteria:**
  - min_credit_score = 650: The minimum required credit score is still 650.
  - min_annual_income = 40000: The minimum required annual income is still $40,000.

    - max_loan_amount = 20000: The maximum loan amount is still $20,000.
- **Applicant Information (Priya):**
    - applicant_name = "Priya": The applicant's name is now "Priya".
    - applicant_credit_score = 680: Priya's credit score is 680.
    - applicant_annual_income = 45000: Priya's annual income is $45,000.
    - requested_loan_amount = 18000: The loan amount Priya is requesting is $18,000.
- **Loan Evaluation:**
    - if applicant_credit_score >= min_credit_score and applicant_annual_income >= min_annual_income and requested_loan_amount <= max_loan_amount:: This checks if Priya meets all three conditions based on the defined criteria.
    - print("Loan approved for {}.".format(applicant_name)): If all conditions are true, it prints that the loan is approved for Priya.
    - else:: If any condition is false.
    - print("Loan denied for {}.".format(applicant_name)): It prints that the loan is denied for Priya.

Essentially, this code is applying the same loan approval rules to a different person, Priya, with her specific financial details.

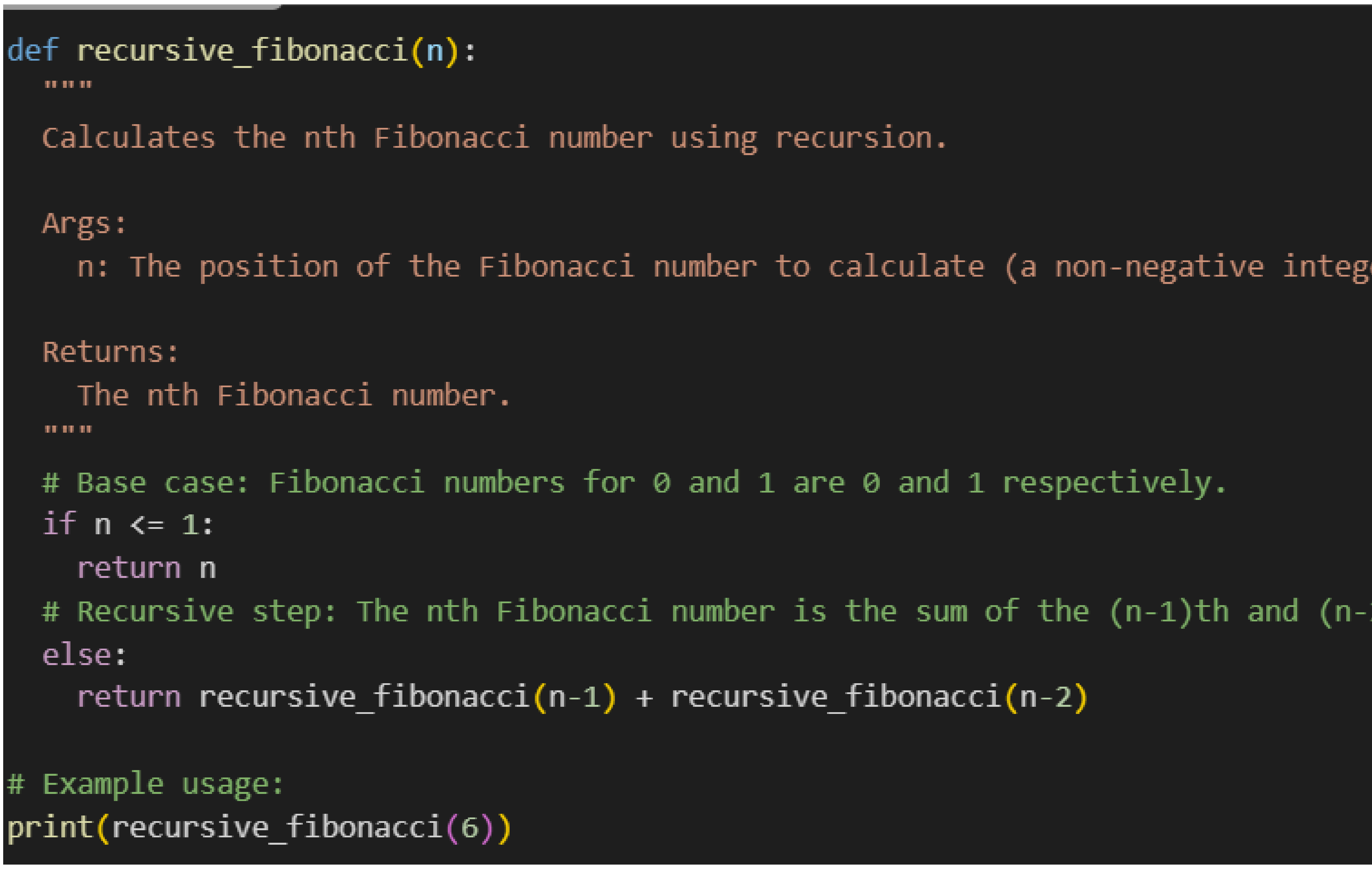
Task Description#3 (Transparency)

• Write prompt to write function calculate

the nth Fibonacci number using recursion and generate comments and explain code document

PROMPT:write a function in pyhton which calculates the nth fibbonocci number using recursion and generate comments

CODE:



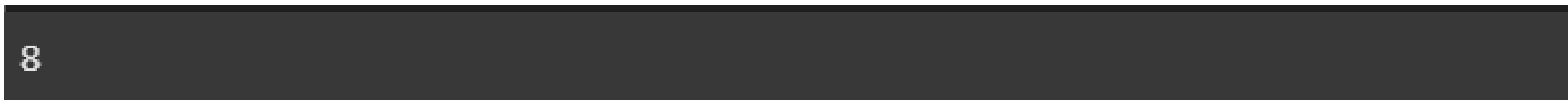
```
def recursive_fibonacci(n):
  """
  Calculates the nth Fibonacci number using recursion.

  Args:
    n: The position of the Fibonacci number to calculate (a non-negative integ

  Returns:
    The nth Fibonacci number.
  """
  # Base case: Fibonacci numbers for 0 and 1 are 0 and 1 respectively.
  if n <= 1:
    return n
  # Recursive step: The nth Fibonacci number is the sum of the (n-1)th and (n-
  else:
    return recursive_fibonacci(n-1) + recursive_fibonacci(n-2)

# Example usage:
print(recursive_fibonacci(6))
```

OUTPUT:



```
8
```

EXPLANATION:

This code defines a Python function called recursive_fibonacci that calculates the nth number in the Fibonacci sequence using recursion.

Here's a breakdown:

- **def recursive_fibonacci(n):**: This line defines the function named recursive_fibonacci that takes one argument, n, which represents the position of the Fibonacci number we want to calculate (starting from 0).
- **Docstring ("""Docstring goes here""")**: This is a docstring that explains what the function does, its arguments (Args), and what it returns (Returns).
- **if n <= 1:**: This is the **base case** of the recursion. The Fibonacci sequence starts with 0 and 1. If n is 0 or 1, the function simply returns n itself, stopping the recursion.
- **else:**: If n is greater than 1, the code enters the **recursive step**.
- **return recursive_fibonacci(n-1) + recursive_fibonacci(n-2)**: This is the core of the recursive calculation. The function calls itself twice: once with n-1 and once with n-2. It then returns the sum of the results of these two recursive calls. This continues until the base case is reached (when n becomes 0 or 1).
- **# Example usage:**: This is a comment indicating the following line is an example of how to use the function.
- **print(recursive_fibonacci(6))**: This line calls the recursive_fibonacci function with n=6 and prints the result, which is the 6th Fibonacci number (starting from 0, the sequence is 0, 1, 1, 2, 3, 5, 8...).

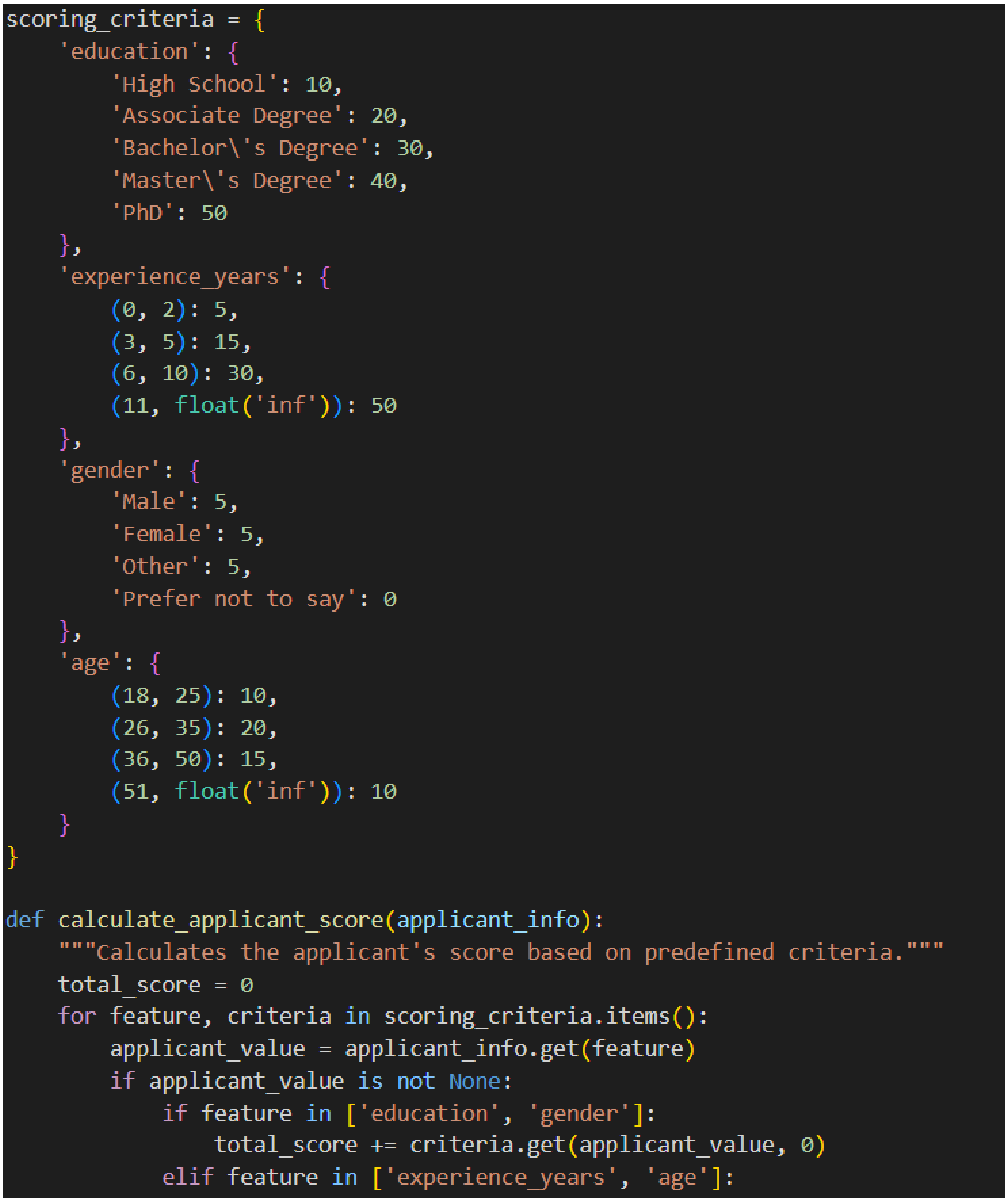
In essence, this function breaks down the problem of finding the nth Fibonacci number into smaller, identical subproblems until it reaches the simplest cases (0 and 1), and then combines the results back up.

Task Description#4 (Bias)

• Ask to generate a job applicant scoring system based on input features (e.g.,

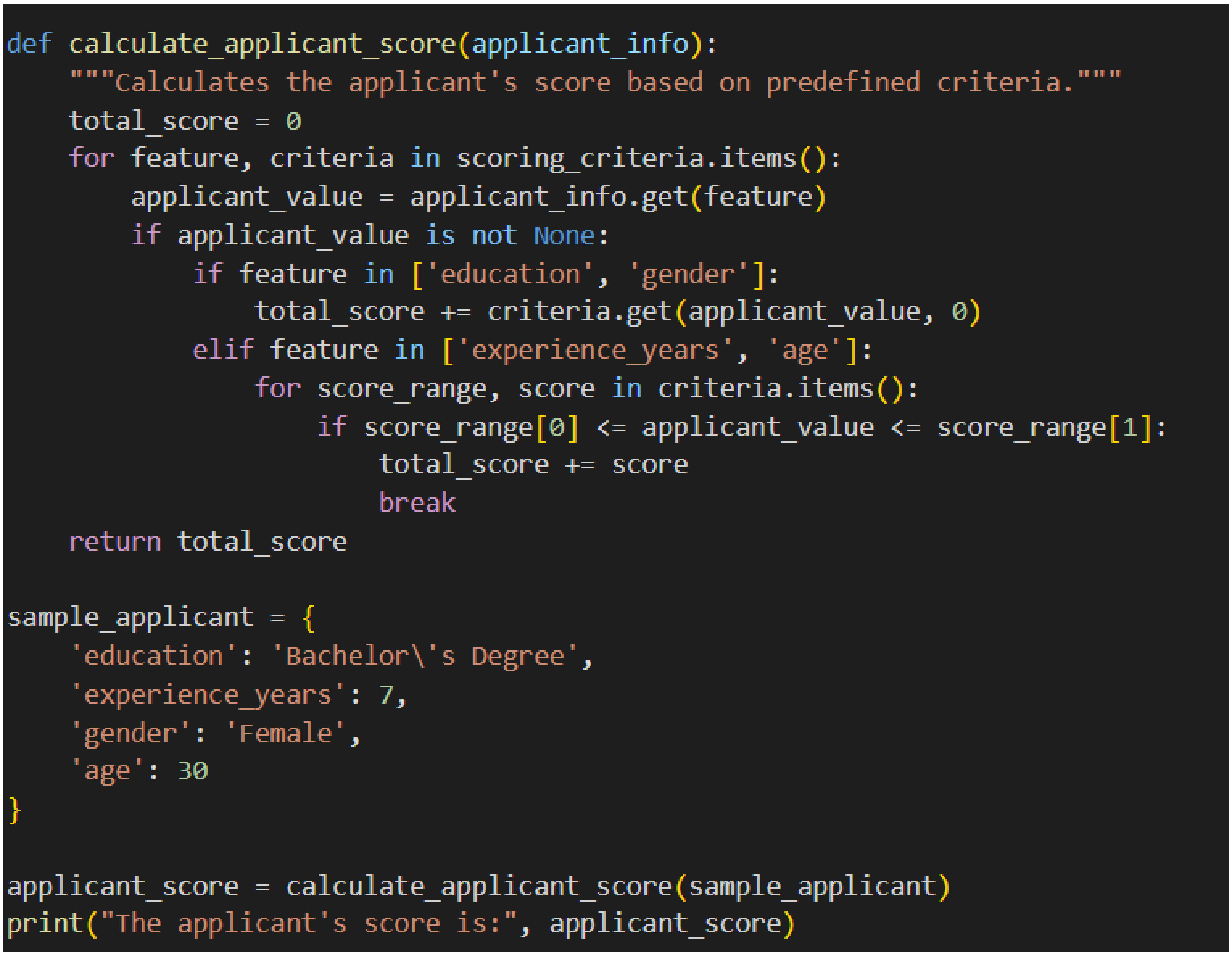
education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings.

PROMPT:
CODE:



```python
scoring_criteria = {
    'education': {
        'High School': 10,
        'Associate Degree': 20,
        'Bachelor\'s Degree': 30,
        'Master\'s Degree': 40,
        'PhD': 50
    },
    'experience_years': {
        (0, 2): 5,
        (3, 5): 15,
        (6, 10): 30,
        (11, float('inf')): 50
    },
    'gender': {
        'Male': 5,
        'Female': 5,
        'Other': 5,
        'Prefer not to say': 0
    },
    'age': {
        (18, 25): 10,
        (26, 35): 20,
        (36, 50): 15,
        (51, float('inf')): 10
    }
}

def calculate_applicant_score(applicant_info):
    """Calculates the applicant's score based on predefined criteria."""
    total_score = 0
    for feature, criteria in scoring_criteria.items():
        applicant_value = applicant_info.get(feature)
        if applicant_value is not None:
            if feature in ['education', 'gender']:
                total_score += criteria.get(applicant_value, 0)
            elif feature in ['experience_years', 'age']:
```

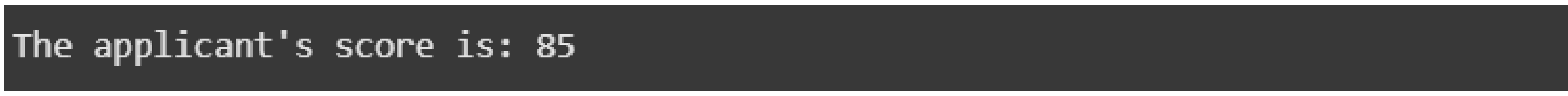


```python
def calculate_applicant_score(applicant_info):
    """Calculates the applicant's score based on predefined criteria."""
    total_score = 0
    for feature, criteria in scoring_criteria.items():
        applicant_value = applicant_info.get(feature)
        if applicant_value is not None:
            if feature in ['education', 'gender']:
                total_score += criteria.get(applicant_value, 0)
            elif feature in ['experience_years', 'age']:
                for score_range, score in criteria.items():
                    if score_range[0] <= applicant_value <= score_range[1]:
                        total_score += score
                        break
    return total_score

sample_applicant = {
    'education': 'Bachelor\'s Degree',
    'experience_years': 7,
    'gender': 'Female',
    'age': 30
}

applicant_score = calculate_applicant_score(sample_applicant)
print("The applicant's score is:", applicant_score)
```

OUTPUT:



```
The applicant's score is: 85
```

EXPLANATION:

This code defines a scoring system for job applicants based on various criteria.

First, the scoring_criteria dictionary is set up. This dictionary holds the rules for how points are awarded for different features like 'education', 'experience_years', 'gender', and 'age'. For example, a PhD in education is worth 50 points, while having 6-10 years of experience is worth 30 points.

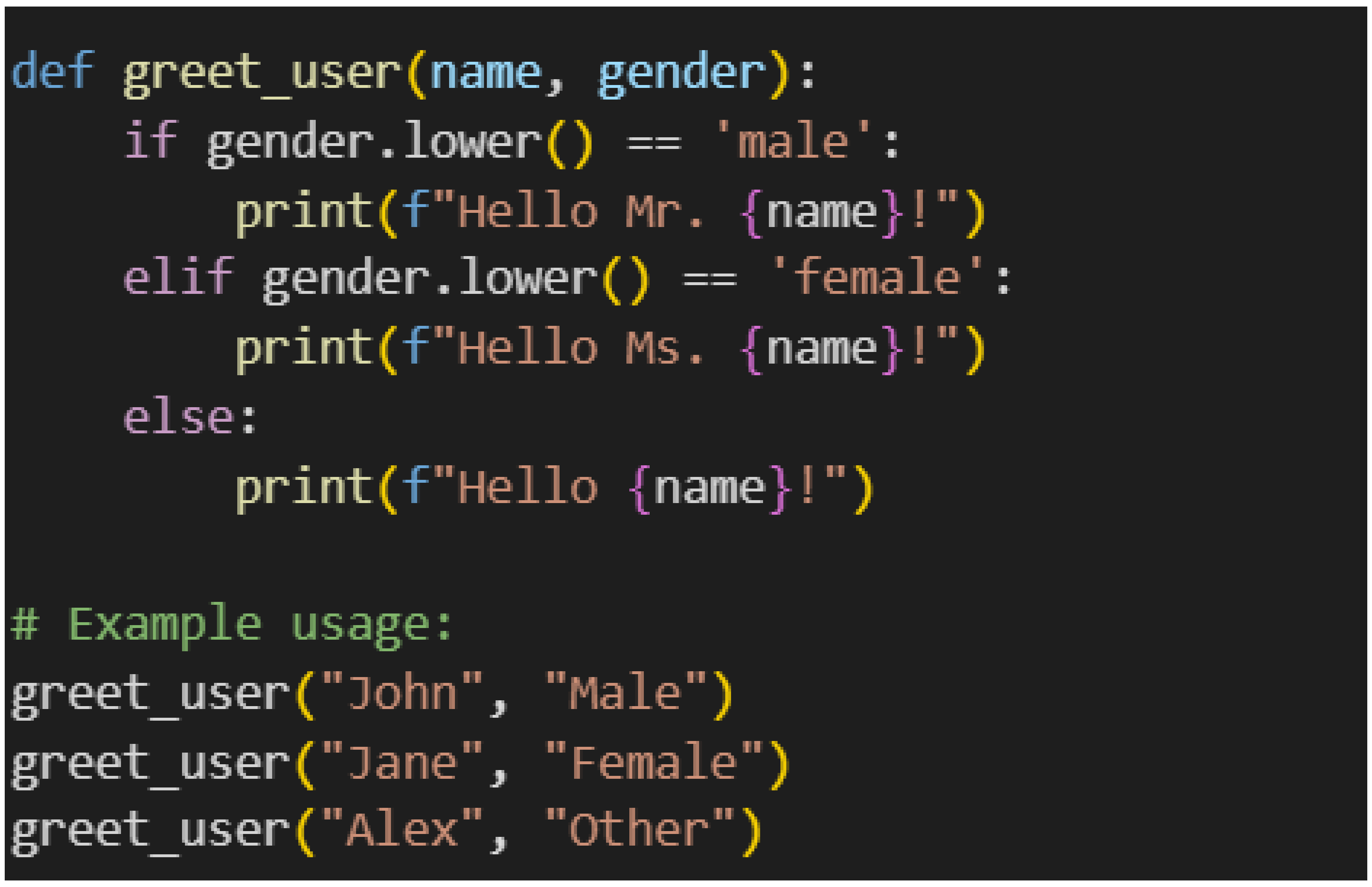
Then, the calculate_applicant_score function takes an applicant's information (as a dictionary) and calculates a total score by iterating through the scoring_criteria. It checks the applicant's value for each feature and adds the corresponding points based on the predefined criteria. For numerical features like experience and age, it checks which range the applicant's value falls into.

Finally, a sample_applicant dictionary is created with example information, and the calculate_applicant_score function is called with this sample data. The resulting score for the sample applicant is then printed.

Task Description#5 (Inclusiveness)

PROMPT: Code snippet: def greet_user(name, gender):

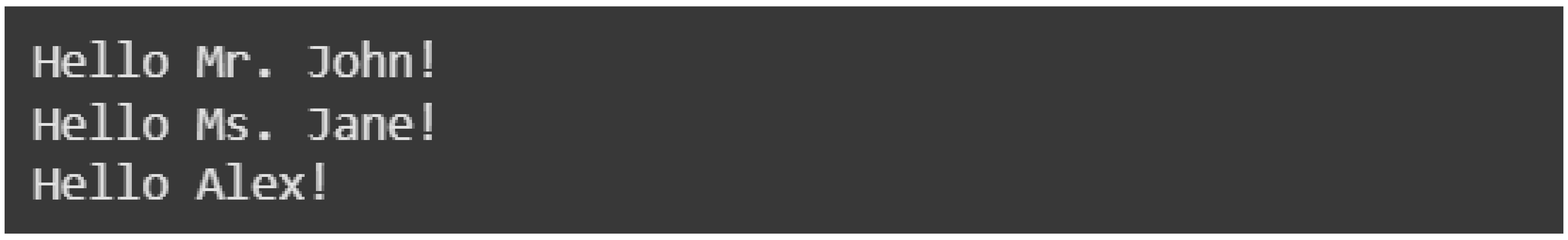
CODE:



```python
def greet_user(name, gender):
    if gender.lower() == 'male':
        print(f"Hello Mr. {name}!")
    elif gender.lower() == 'female':
        print(f"Hello Ms. {name}!")
    else:
        print(f"Hello {name}!")

# Example usage:
greet_user("John", "Male")
greet_user("Jane", "Female")
greet_user("Alex", "Other")
```

OUTPUT:



```
Hello Mr. John!
Hello Ms. Jane!
Hello Alex!
```

EXPLANATION:

This code defines a Python function called greet_user that takes two arguments: name and gender.

Here's how it works:

1. **def greet_user(name, gender):**: This line defines the function named greet_user and specifies that it accepts two parameters: name and gender.
2. **if gender.lower() == 'male':**: This checks if the gender provided, converted to lowercase, is equal to 'male'. If it is, the code inside this if block is executed.
3. **print(f"Hello Mr. {name}!")**: If the gender is 'male', this line prints a greeting using an f-string, addressing the user as "Mr." followed by their name.
4. **elif gender.lower() == 'female':**: If the gender is not 'male', this checks if the gender (in lowercase) is equal to 'female'. If it is, the code inside this elif block is executed.
5. **print(f"Hello Ms. {name}!")**: If the gender is 'female', this line prints a greeting using an f-string, addressing the user as "Ms." followed by their name.
6. **else:**: If the gender is neither 'male' nor 'female', the code inside this else block is executed.
7. **print(f"Hello {name}!")**: For any other gender value, this line prints a general greeting using an f-string, simply saying "Hello" followed by the user's name.

The lines after the function definition show examples of how to use the greet_user function with different names and genders.