SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName:B. Tech		**Assignment Type: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week3 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:5.2(Present assignment number)/**24**(Total number of assignments)			

Q.No.	**Question**	***ExpectedTime to complete***
1	Lab 5: Ethical Foundations – Responsible AI Coding Practices **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.	Week3 - Wednesday

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 Description#1 (Privacy and Data Security)

- 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.

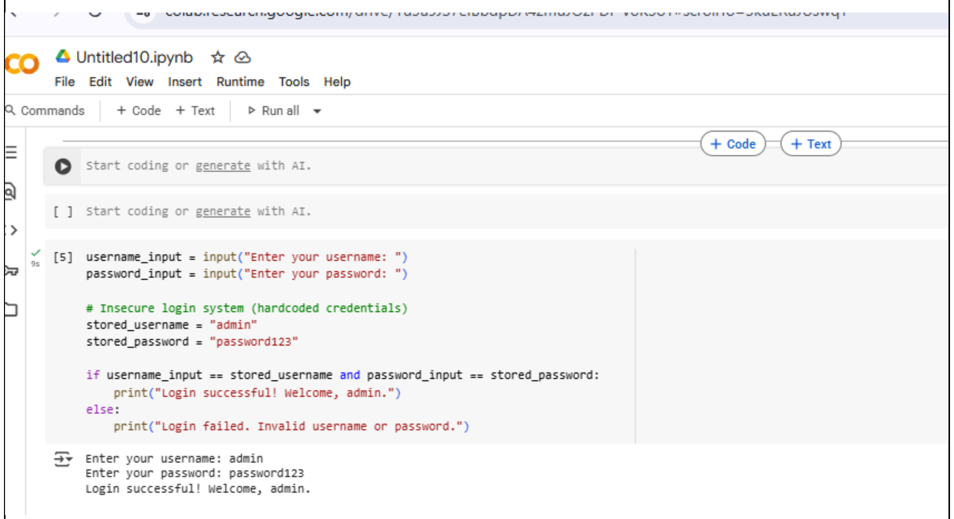
Expected Output#1

- Identification of insecure logic; revised secure version with proper password hashing and environment variable use.

Prompt:

1.Write a Python program that asks the user for a username and password.

Code#1


Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all
+ Code + Text
Start coding or generate with AI.
Start coding or generate with AI.

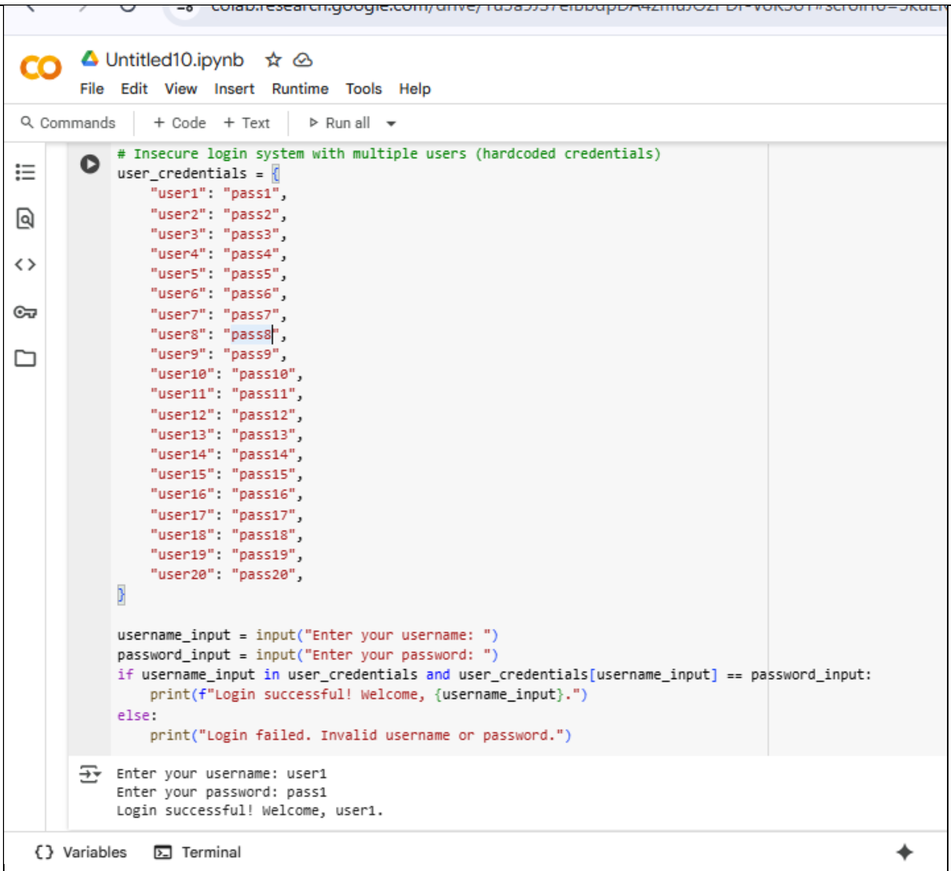
```
username_input = input("Enter your username: ")
password_input = input("Enter your password: ")

# Insecure login system (hardcoded credentials)
stored_username = "admin"
stored_password = "password123"

if username_input == stored_username and password_input == stored_password:
    print("Login successful! Welcome, admin.")
else:
    print("Login failed. Invalid username or password.")
```

```
Enter your username: admin
Enter your password: password123
Login successful! Welcome, admin.
```

Prompt:

1.Create a Python program for a login system that stores 20 username-password pairs inside a dictionary.

Code#1:


Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Variables Terminal

```python
# Insecure login system with multiple users (hardcoded credentials)
user_credentials = {
    "user1": "pass1",
    "user2": "pass2",
    "user3": "pass3",
    "user4": "pass4",
    "user5": "pass5",
    "user6": "pass6",
    "user7": "pass7",
    "user8": "pass8",
    "user9": "pass9",
    "user10": "pass10",
    "user11": "pass11",
    "user12": "pass12",
    "user13": "pass13",
    "user14": "pass14",
    "user15": "pass15",
    "user16": "pass16",
    "user17": "pass17",
    "user18": "pass18",
    "user19": "pass19",
    "user20": "pass20",
}

username_input = input("Enter your username: ")
password_input = input("Enter your password: ")
if username_input in user_credentials and user_credentials[username_input] == password_input:
    print(f"Login successful! Welcome, {username_input}.")
else:
    print("Login failed. Invalid username or password.")
```

```
Enter your username: user1
Enter your password: pass1
Login successful! Welcome, user1.
```

Observations#1:

1. The program stores usernames and passwords in a dictionary.
2. The system works for 20 users.
3. If the entered username exists and the password matches, login is successful.

Code Explanation#1:

1. A dictionary named users is created to store 20 username–password pairs.
2. The user is asked to input a username and a password.
3. The program checks whether the entered username exists in the dictionary.
4. If both username and password are correct, it prints a Login Success.

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.

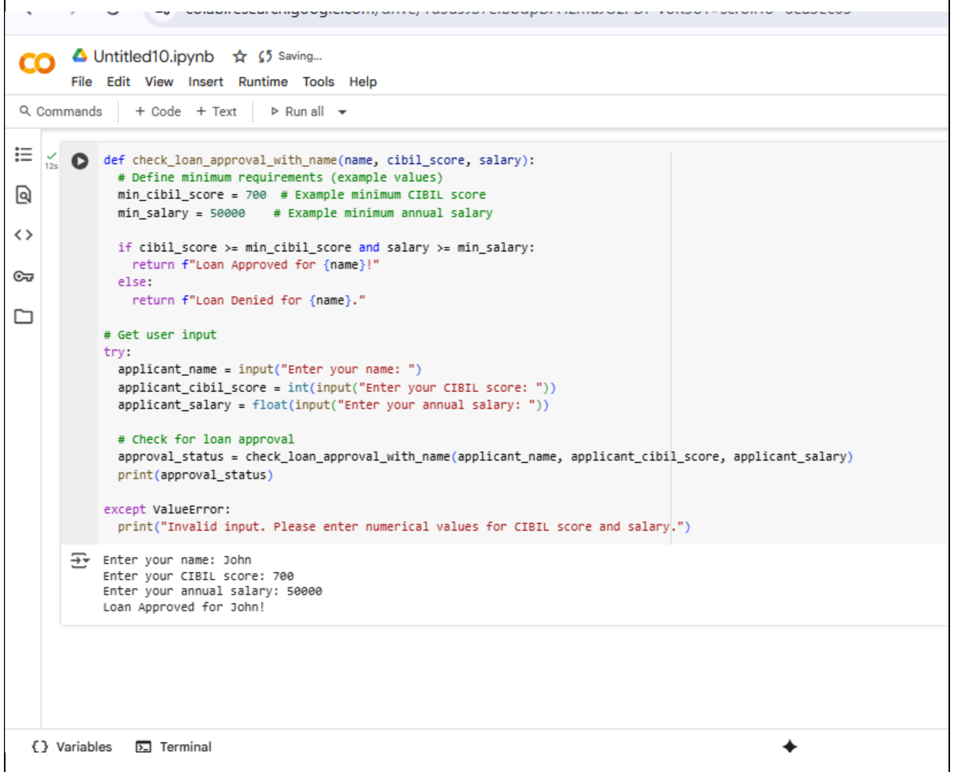
Expected Output#2

- Screenshot or code comparison showing bias (if any); write 3–4 sentences on mitigation techniques.

Prompt #2:

1.Write a python code for loan approval of a person based on the three inputs: the person's name, credit score, and annual income.

Code#2:


Untitled10.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Variables Terminal

```python
def check_loan_approval_with_name(name, cibil_score, salary):
  # Define minimum requirements (example values)
  min_cibil_score = 700  # Example minimum CIBIL score
  min_salary = 50000    # Example minimum annual salary

  if cibil_score >= min_cibil_score and salary >= min_salary:
    return f"Loan Approved for {name}!"
  else:
    return f"Loan Denied for {name}."

# Get user input
try:
  applicant_name = input("Enter your name: ")
  applicant_cibil_score = int(input("Enter your CIBIL score: "))
  applicant_salary = float(input("Enter your annual salary: "))

  # Check for loan approval
  approval_status = check_loan_approval_with_name(applicant_name, applicant_cibil_score, applicant_salary)
  print(approval_status)

except ValueError:
  print("Invalid input. Please enter numerical values for CIBIL score and salary.")
```

```
Enter your name: John
Enter your CIBIL score: 700
Enter your annual salary: 50000
Loan Approved for John!
```

Observations#2:

1. The function takes three inputs: name, credit_score, and salary.
2. Credit score and income are treated as numeric values, while name is a string.
3. If either the credit score or income is below the threshold, the loan is denied.
4. The program is simple and rule-based.
5. The output clearly distinguishes between loan approved and loan denied.

Code Explanation#2:

1.The function is defined to take three inputs.

2.The if statement checks whether the credit_score and salary is above the minimum threshold.

3.If both conditions are satisfied, the function **returns a message** indicating the loan is approved for that person.

Task Description#3 (Transparency)

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

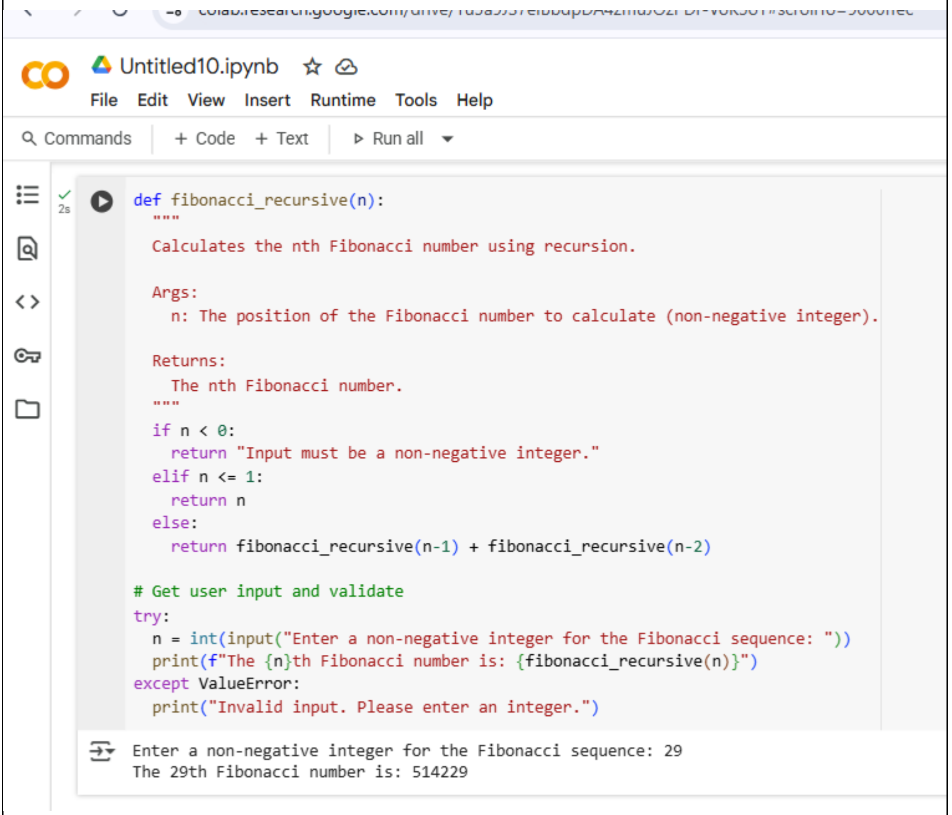
Expected Output#3

- Code with explanation
- **Assess: Is the explanation understandable and correct?**

Prompt #3:

1.Write a Python function to calculate the nth Fibonacci number using recursion.

Code#3:


Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

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

  Args:
    n: The position of the Fibonacci number to calculate (non-negative integer).

  Returns:
    The nth Fibonacci number.
  """
  if n < 0:
    return "Input must be a non-negative integer."
  elif n <= 1:
    return n
  else:
    return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

# Get user input and validate
try:
  n = int(input("Enter a non-negative integer for the Fibonacci sequence: "))
  print(f"The {n}th Fibonacci number is: {fibonacci_recursive(n)}")
except ValueError:
  print("Invalid input. Please enter an integer.")
```

```
Enter a non-negative integer for the Fibonacci sequence: 29
The 29th Fibonacci number is: 514229
```

Observations#3:

1. The function calculates the nth Fibonacci number using recursion.
2. The function adds the two previous Fibonacci numbers to get the current one.
3. Works correctly for small values but may be slow for large numbers.

Code Explanation#3:

1. def Fibonacci_recursive(n): defines the function that calculates the nth Fibonacci number.
2. If the input is not an integer, the function raises an error or invalid input.
3. The next step checks for negative numbers.
4. If the number is valid, it prints the nth Fibonacci number.

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.

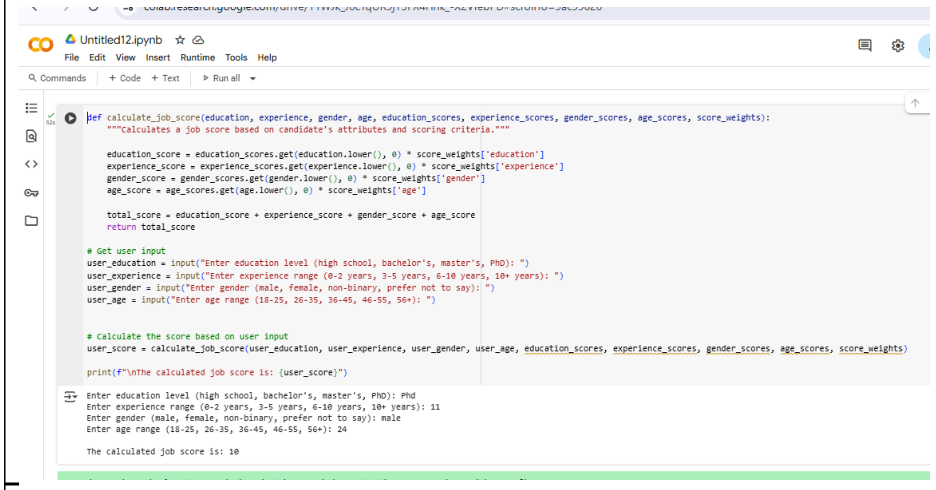
Expected Output#4

- Python code
- Analyze is there any bias with respect to gender or any

Prompt#4:

1.Write a Python function that calculates a score for a job applicant based on their education, years of experience, gender, and age. Return the total score.

Code#4:


Untitled12.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```python
def calculate_job_score(education, experience, gender, age, education_scores, experience_scores, gender_scores, age_scores, score_weights):
    """Calculates a job score based on candidate's attributes and scoring criteria."""

    education_score = education_scores.get(education.lower(), 0) * score_weights['education']
    experience_score = experience_scores.get(experience.lower(), 0) * score_weights['experience']
    gender_score = gender_scores.get(gender.lower(), 0) * score_weights['gender']
    age_score = age_scores.get(age.lower(), 0) * score_weights['age']

    total_score = education_score + experience_score + gender_score + age_score
    return total_score

# Get user input
user_education = input("Enter education level (high school, bachelor's, master's, PhD): ")
user_experience = input("Enter experience range (0-2 years, 3-5 years, 6-10 years, 10+ years): ")
user_gender = input("Enter gender (male, female, non-binary, prefer not to say): ")
user_age = input("Enter age range (18-25, 26-35, 36-45, 46-55, 56+): ")


# Calculate the score based on user input
user_score = calculate_job_score(user_education, user_experience, user_gender, user_age, education_scores, experience_scores, gender_scores, age_scores, score_weights)

print(f"\nThe calculated job score is: {user_score}")
```

```
Enter education level (high school, bachelor's, master's, PhD): Phd
Enter experience range (0-2 years, 3-5 years, 6-10 years, 10+ years): 11
Enter gender (male, female, non-binary, prefer not to say): male
Enter age range (18-25, 26-35, 36-45, 46-55, 56+): 24

The calculated job score is: 10
```

Observations#4:

1. Scores applicants based on education, experience, gender, and age.
2. Age contributes points based on preferred age ranges.
3. Age contributes points based on preferred age ranges.

Code Explanation#4:

1. def calculate_job_score(...) defines the function with four inputs.
2. Experience adds 5 points per year.
3. Age gives extra points.
4. Gender is taken as input but not used in scoring.
5. Finally, the total score is returned and printed.

Task Description#5 (Inclusiveness)

- Code Snippet

```python
def greet_user(name, gender):
    if gender.lower() == "male":
        title = "Mr."
    else:
        title = "Mrs."
    return f"Hello, {title} {name}! Welcome."
```

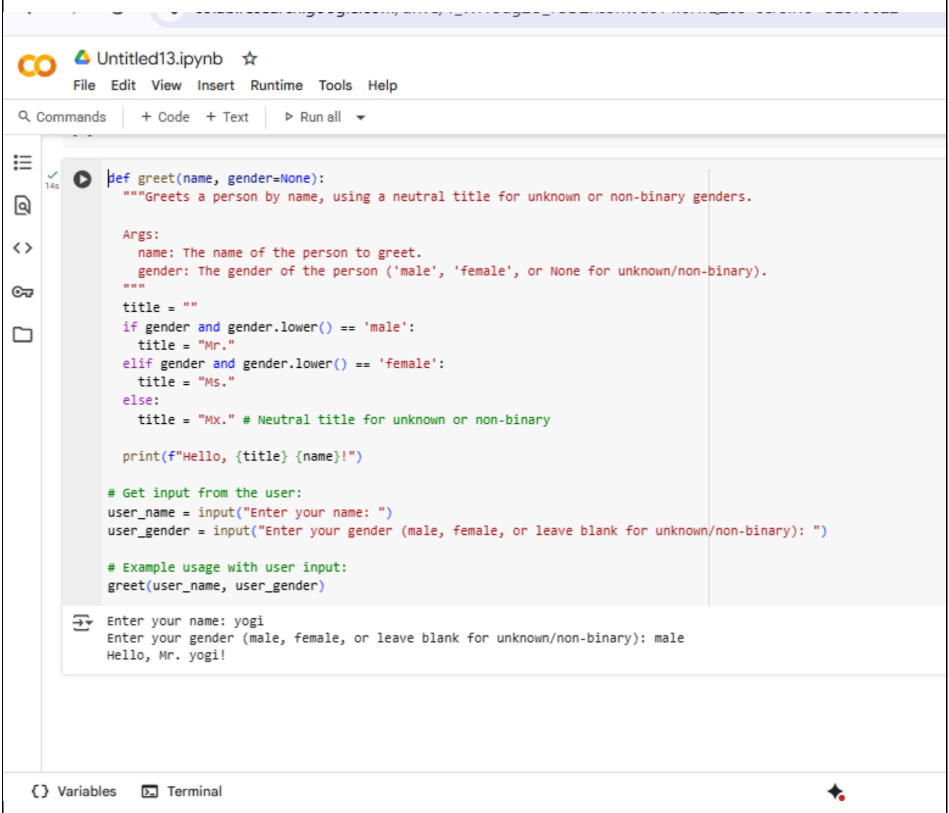
Expected Output#5

- Regenerate code that includes **gender-neutral** also

Prompt#5:

1.Write a Python program that asks for a person's name and gender, then greets them. Use neutral title for unknown or non-binary genders.

Code#5:


Untitled13.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Variables Terminal

```
def greet(name, gender=None):
  """Greets a person by name, using a neutral title for unknown or non-binary genders.

  Args:
    name: The name of the person to greet.
    gender: The gender of the person ('male', 'female', or None for unknown/non-binary).
  """
  title = ""
  if gender and gender.lower() == 'male':
    title = "Mr."
  elif gender and gender.lower() == 'female':
    title = "Ms."
  else:
    title = "Mx." # Neutral title for unknown or non-binary

  print(f"Hello, {title} {name}!")

# Get input from the user:
user_name = input("Enter your name: ")
user_gender = input("Enter your gender (male, female, or leave blank for unknown/non-binary): ")

# Example usage with user input:
greet(user_name, user_gender)
```

```
Enter your name: yogi
Enter your gender (male, female, or leave blank for unknown/non-binary): male
Hello, Mr. yogi!
```

Observations#5:

1. Greets the user by their name.
2. Uses a neutral title Mx for unknown or non-binary gender.
3. Greeting with Mr for male and Ms for female.

Code Explanation#5:

1. The function greet_person(name, gender=None) greets people by name.
2. The title Mx is used if the gender belongs to unknown.
3. If the gender is male the program uses "Mr".
4. If the gender is female , it uses "Ms".
5. Function works for any input, including unknown or non-binary gender.
6. Finally, it prints a greeting

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	Max Marks
Transparency	0.5
Bias	1.0
Inclusiveness	0.5
Data security and Privacy	0.5
Total	**2.5 Marks**