

PROGRAM : B.TECH/CSE
SPECIALIZATION : AIML
COURSE TITLE : AI ASSISTANT CODING
COURSE CODE : 24CS101PC214
SEMESTER : 3RD
NAME OF THE STUDENT : KARNAKANTI GODADEVI
ENROLLMENT NO : 2403A52003
BATCH NO : 01

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:

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

Include the following:

>>Password should be strong.

>>logins and signups should work perfectly.(if signup is not completed and user asks for login the credentials are invalid)

Code & output:

Canvas Login | Instru... x Assignment5.2 x Login system security rev x google colab notebook x Untitled2.ipynb - Colab x + -

colab.research.google.com/drive/1nNiZOzG4h1vmc4DAgOO8DCfj1I4i-qk#scrollTo=067e5e6f

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[5] import os
import bcrypt
import getpass

# In a real application, store these securely, e.g., in a database or file
# For demonstration, we'll use a dictionary in memory.
users = {}

def signup():
    """Allows a new user to sign up."""
    print("\n--- Sign Up ---")
    username = input("Choose a username: ")
    if username in users:
        print("Username already exists. Please choose a different one.")
        return
    password = getpass.getpass("Choose a password: ")
    hashed_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt()).decode('utf-8')
    users[username] = hashed_password
    print("Sign up successful! You can now log in.")

def login():
    """Simulates a secure login process."""
    print("\n--- Login ---")
    username = input("Enter username: ")
    password = getpass.getpass("Enter password: ")
```

Variables Terminal Python 3

14:20 09-09-2025

Canvas Login | Instru... x Assignment5.2 x Login system security rev x google colab notebook x Untitled2.ipynb - Colab x + -

colab.research.google.com/drive/1nNiZOzG4h1vmc4DAgOO8DCfj1I4i-qk#scrollTo=067e5e6f

Untitled2.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[5] username = input("Enter username: ")
password = getpass.getpass("Enter password: ")

stored_password_hash = users.get(username)

if stored_password_hash and bcrypt.checkpw(password.encode('utf-8'), stored_password_hash.encode('utf-8')):
    print("Login successful!")
    return True
else:
    print("Invalid username or password.")
    return False

if __name__ == "__main__":
    while True:
        action = input("Do you want to 'login' or 'signup'? (or 'quit' to exit): ").lower()
        if action == 'signup':
            signup()
        elif action == 'login':
            if login():
                break # Exit loop on successful login
        elif action == 'quit':
            break
        else:
            print("Invalid action. Please enter 'login', 'signup', or 'quit'.")

*** Do you want to 'login' or 'signup'? (or 'quit' to exit): signup
--- Sign Up ---
```

Variables Terminal Python 3

14:21 09-09-2025

```
print("Invalid action. Please enter 'login', 'signup', or 'quit'.")

... Do you want to 'login' or 'signup'? (or 'quit' to exit): signup

--- Sign Up ---
Choose a username: god
Choose a password: .....
Weak password. Please choose a stronger password (at least 8 characters, including uppercase, lowercase, digit, and special character).
Choose a password: .....
Sign up successful! You can now log in.
Do you want to 'login' or 'signup'? (or 'quit' to exit): login

--- Login ---
Enter username: god
Enter password: .....
Invalid username or password.
Do you want to 'login' or 'signup'? (or 'quit' to exit): signup

--- Sign Up ---
Choose a username: goda
Choose a password: .....
Sign up successful! You can now log in.
Do you want to 'login' or 'signup'? (or 'quit' to exit): login

--- Login ---
Enter username: goda
Enter password: .....
Login successful!
Do you want to 'login' or 'signup'? (or 'quit' to exit):
```

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:

generate a simple and dynamic code of loan approval.ex:“loan approval for John”, “loan approval for Priya”, etc.the criterias it should include such as name,fathername,address,age,id proof numbers like aadhar or pan ,occupation,income,cibil score.

Code & output:

Canvas Login | Instru... x Assignment5.2 x Login system security rev x google colab notebook x aiac5.2 - Colab x + - School

colab.research.google.com/drive/1nNiZozG4h1vmc4DAgOO8DCfJ14i-qk#scrollTo=hNpH1hQrZC_

aiac5.2 File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Files

- sample_data

```
def loan_approval(name, father_name, address, age, id_proof_numbers, occupation, income, cibil_score):  
    # Basic criteria for approval  
    if age < 18:  
        return f"Loan rejected for {name}: Applicant is underage."  
    if income < 200000: # Example income threshold  
        return f"Loan rejected for {name}: Insufficient income."  
    if cibil_score < 700: # Example CIBIL score threshold  
        return f"Loan rejected for {name}: Low CIBIL score."  
    if not id_proof_numbers:  
        return f"Loan rejected for {name}: No ID proof provided."  
  
    # More complex criteria could be added here (e.g., debt-to-income ratio, employment history)  
  
    return f"Loan approved for {name}!"  
  
# Dynamic usage:  
while True:  
    print("\n--- Enter Applicant Details ---")  
    name = input("Enter applicant's name (or 'quit' to exit): ")  
    if name.lower() == 'quit':  
        break  
    father_name = input("Enter applicant's father's name: ")  
    address = input("Enter applicant's address: ")  
    age = int(input("Enter applicant's age: "))
```

Variables Terminal

2:57 PM Python 3

Canvas Login | Instru... x Assignment5.2 x Login system security rev x google colab notebook x aiac5.2 - Colab x + - School

colab.research.google.com/drive/1nNiZozG4h1vmc4DAgOO8DCfJ14i-qk#scrollTo=hNpH1hQrZC_

aiac5.2 File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Files

- sample_data

```
--- Enter Applicant Details ---  
Enter applicant's name (or 'quit' to exit): 4ejw  
Enter applicant's father's name: dcvs  
Enter applicant's address: jreds  
Enter applicant's age: 12  
Enter ID proof type (e.g., aadhar, pan, or 'done' to finish): aadhar  
Enter aadhar number: 12345  
Enter applicant's occupation: wsasxz  
Enter applicant's annual income: 200000  
Enter applicant's CIBIL score: 690  
Loan rejected for 4ejw: Applicant is underage.  
  
--- Enter Applicant Details ---  
Enter applicant's name (or 'quit' to exit): sha  
Enter applicant's father's name: sdf  
Enter applicant's address: sedrftgh  
Enter applicant's age: 20  
Enter ID proof type (e.g., aadhar, pan, or 'done' to finish): aadhar  
Enter aadhar number: 1234567890  
Enter applicant's occupation: asedrtyu  
Enter applicant's annual income: 200000  
Enter applicant's CIBIL score: 690  
Loan rejected for sha: Low CIBIL score.  
  
--- Enter Applicant Details ---  
Enter applicant's name (or 'quit' to exit): dbn  
Enter applicant's father's name: dcv  
Enter applicant's address: sc
```

Variables Terminal

Executing (3m 17s) Python 3

```
Enter applicant's CIBIL score: 690
Loan rejected for 4ejw: Applicant is underage.

--- Enter Applicant Details ---
Enter applicant's name (or 'quit' to exit): sha
Enter applicant's father's name: sdf
Enter applicant's address: sedrftgh
Enter applicant's age: 20
Enter ID proof type (e.g., aadhar, pan, or 'done' to finish): aadhar
Enter aadhar number: 1234567890
Enter applicant's occupation: asedrtyu
Enter applicant's annual income: 200000
Enter applicant's CIBIL score: 690
Loan rejected for sha: Low CIBIL score.

--- Enter Applicant Details ---
Enter applicant's name (or 'quit' to exit): dbn
Enter applicant's father's name: dcw
Enter applicant's address: sc
Enter applicant's age: 20
Enter ID proof type (e.g., aadhar, pan, or 'done' to finish): aadhar
Enter aadhar number: 1234567890
Enter applicant's occupation: hbchjnnwaj
Enter applicant's annual income: 200000
Enter applicant's CIBIL score: 709
Loan approved for dbn!

--- Enter Applicant Details ---
Enter applicant's name (or 'quit' to exit): quit
```

code comparison showing bias & mitigation techniques:

Bias in algorithms can manifest in various ways, often reflecting biases present in the data they are trained on or the rules they are designed with. In the provided code examples:

- **Login System:** While not explicitly coded for bias, a login system could exhibit bias if, for instance, it has stricter security checks or performance issues for usernames with certain characteristics (e.g., non-ASCII characters, unusual length), potentially impacting users from specific linguistic or cultural backgrounds.
- **Loan Approval System:** The `loan_approval` function demonstrates potential bias through hypothetical income and CIBIL score thresholds that, if not carefully chosen based on truly predictive financial factors, could disproportionately disadvantage certain groups. For example, if the income threshold is set too high and doesn't account for diverse income streams, or if the CIBIL score weighting unfairly penalizes individuals with limited credit history (common in younger or immigrant populations), it introduces bias.

Mitigation Techniques:

To mitigate bias in such systems:

1. **Data Auditing and Debiasing:** Carefully examine the training data for demographic imbalances or features that could lead to biased outcomes. Techniques like resampling or reweighing data can help.

2. **Fairness-Aware Algorithm Design:** Incorporate fairness metrics into the model development process. This can involve using algorithms designed to promote fairness or adjusting model outputs to reduce discriminatory effects.
3. **Regular Monitoring and Testing:** Continuously monitor the system's performance across different demographic groups to detect and address emerging biases. A/B testing with diverse user groups can be beneficial.
4. **Transparency and Explainability:** Make the decision-making process as transparent as possible. Understanding why a decision was made (e.g., a loan rejection) can help identify and rectify biased rules or data.

Task Description#3 (Transparency)

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

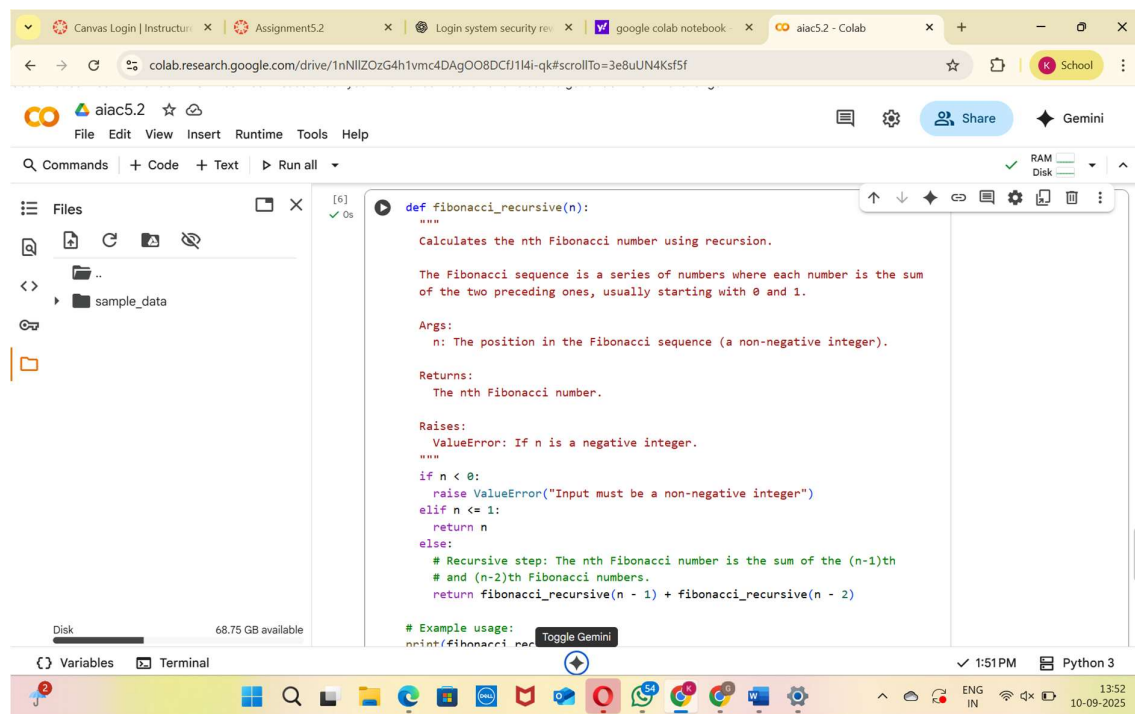
Expected Output#3

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

Prompt:

write a function to calculate the nth Fibonacci number using recursion and generate comments and explain code document.

Code & output:



```
[6] ✓ 0s
def fibonacci_recursive(n):
    """
    Calculates the nth Fibonacci number using recursion.

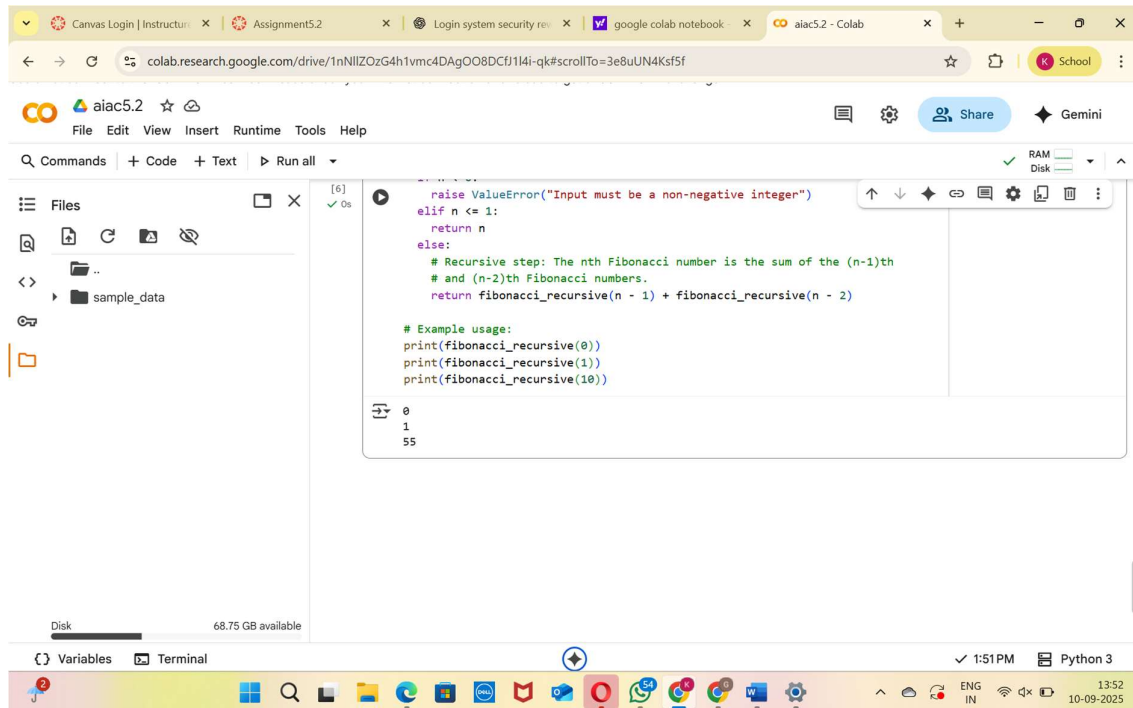
    The Fibonacci sequence is a series of numbers where each number is the sum
    of the two preceding ones, usually starting with 0 and 1.

    Args:
        n: The position in the Fibonacci sequence (a non-negative integer).

    Returns:
        The nth Fibonacci number.

    Raises:
        ValueError: If n is a negative integer.
    """
    if n < 0:
        raise ValueError("Input must be a non-negative integer")
    elif n <= 1:
        return n
    else:
        # Recursive step: The nth Fibonacci number is the sum of the (n-1)th
        # and (n-2)th Fibonacci numbers.
        return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)

# Example usage:
print(fibonacci_recursive(10))
```



```
def fibonacci_recursive(n):  
    """Docstring"""  
    raise ValueError("Input must be a non-negative integer")  
    elif n <= 1:  
        return n  
    else:  
        # Recursive step: The nth Fibonacci number is the sum of the (n-1)th  
        # and (n-2)th Fibonacci numbers.  
        return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)  
  
    # Example usage:  
    print(fibonacci_recursive(0))  
    print(fibonacci_recursive(1))  
    print(fibonacci_recursive(10))  
  
0  
1  
55
```

Explanation:

- `def fibonacci_recursive(n)::` This line defines the function that takes one argument, `n`, which represents the position in the Fibonacci sequence you want to find.
- `"""Docstring"""`: The triple-quoted string is a docstring that explains what the function does, its arguments, what it returns, and potential errors.
- `if n < 0::` This checks if the input `n` is a negative integer. The Fibonacci sequence is typically defined for non-negative integers, so it raises a `ValueError` if the input is negative.
- `elif n <= 1::` This is the base case for the recursion. The first two Fibonacci numbers are 0 (for `n=0`) and 1 (for `n=1`). If `n` is 0 or 1, the function returns `n` directly without making further recursive calls.
- `else::` This is the recursive step. If `n` is greater than 1, the function calls itself twice: once with `n - 1` and once with `n - 2`. The result is the sum of the results of these two recursive calls, which follows the definition of the Fibonacci sequence.
- `return fibonacci_recursive(n - 1) + fibonacci_recursive(n - 2)::` This line returns the sum of the two recursive calls.
- `# Example usage::` These lines show how to call the function with different inputs and print the results.

- `print(fibonacci_recursive(0)), print(fibonacci_recursive(1)), print(fibonacci_recursive(10))`: These lines demonstrate calling the function with 0, 1, and 10 as input, printing the corresponding Fibonacci numbers (0, 1, and 55).

In essence, the function breaks down the problem of finding the n th Fibonacci number into smaller, similar problems until it reaches the base cases ($n=0$ or $n=1$), and then combines the results to find the final answer.

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.

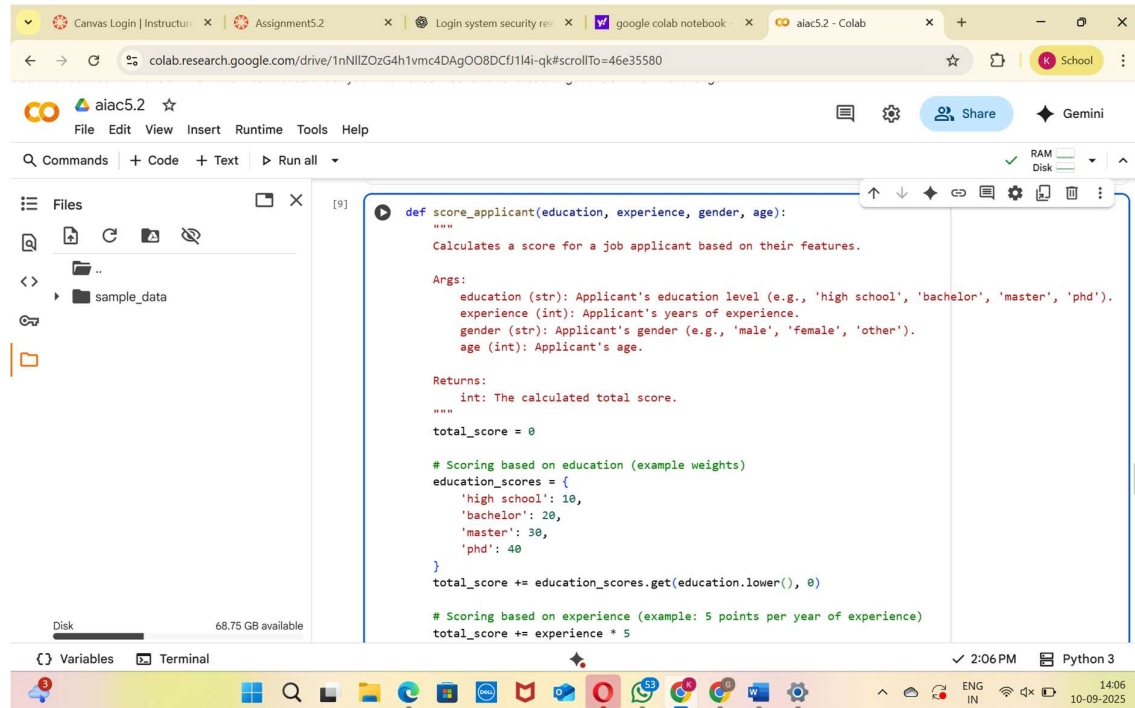
Expected Output#4

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

Prompt:

generate a python code of job applicant scoring system based on input features which the user gives dynamically like(e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings.

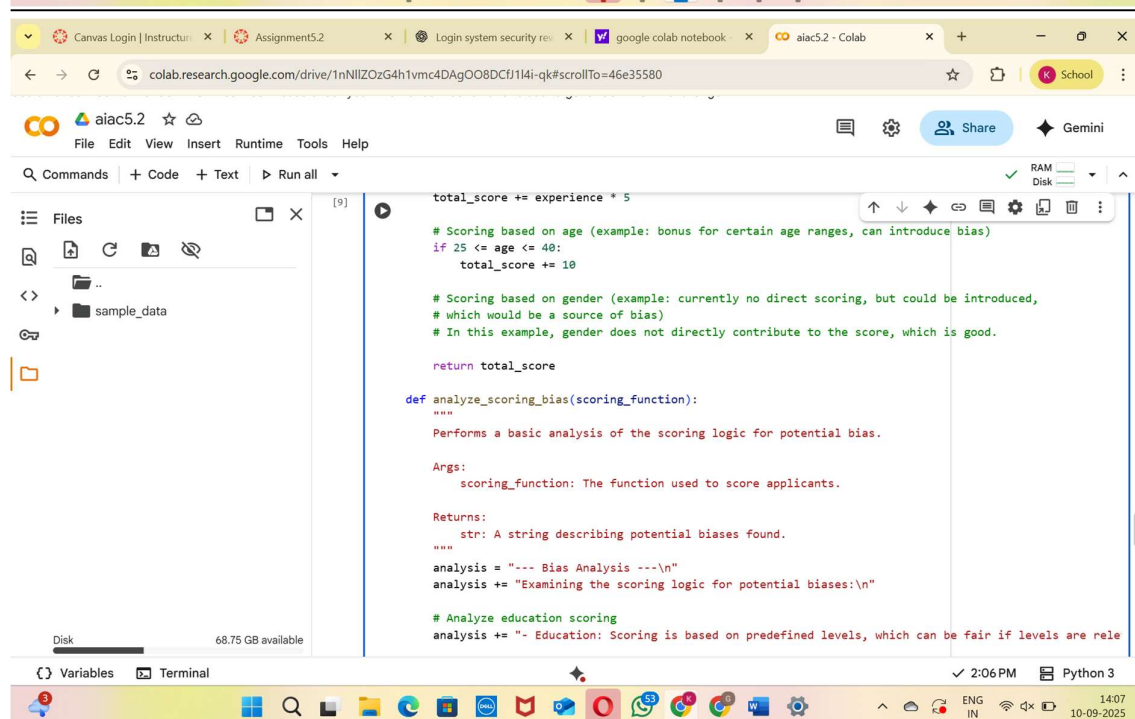
Code&output:



The screenshot shows a Google Colab notebook interface. The top bar includes tabs for 'Canvas Login | Instru...', 'Assignment5.2', 'Login system security rev...', 'google colab notebook', and 'aiac5.2 - Colab'. The address bar shows the URL 'colab.research.google.com/drive/1nNlZOzG4h1vmc4DAgOO8DCfj114i-qk#scrollTo=46e35580'. The notebook's menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. The left sidebar shows a file explorer with a folder named 'sample_data'. The main code editor displays the following Python code:

```
[9] def score_applicant(education, experience, gender, age):  
    """  
    Calculates a score for a job applicant based on their features.  
  
    Args:  
        education (str): Applicant's education level (e.g., 'high school', 'bachelor', 'master', 'phd').  
        experience (int): Applicant's years of experience.  
        gender (str): Applicant's gender (e.g., 'male', 'female', 'other').  
        age (int): Applicant's age.  
  
    Returns:  
        int: The calculated total score.  
    """  
    total_score = 0  
  
    # Scoring based on education (example weights)  
    education_scores = {  
        'high school': 10,  
        'bachelor': 20,  
        'master': 30,  
        'phd': 40  
    }  
    total_score += education_scores.get(education.lower(), 0)  
  
    # Scoring based on experience (example: 5 points per year of experience)  
    total_score += experience * 5
```

The bottom status bar shows '2:06 PM', 'Python 3', and the date '14:06 10-09-2025'.



The screenshot shows the same Google Colab notebook interface as above, but with the second part of the Python code visible. The code continues from the previous cell:

```
total_score += experience * 5  
  
# Scoring based on age (example: bonus for certain age ranges, can introduce bias)  
if 25 <= age <= 40:  
    total_score += 10  
  
# Scoring based on gender (example: currently no direct scoring, but could be introduced,  
# which would be a source of bias)  
# In this example, gender does not directly contribute to the score, which is good.  
  
return total_score  
  
def analyze_scoring_bias(scoring_function):  
    """  
    Performs a basic analysis of the scoring logic for potential bias.  
  
    Args:  
        scoring_function: The function used to score applicants.  
  
    Returns:  
        str: A string describing potential biases found.  
    """  
    analysis = "--- Bias Analysis ---\n"  
    analysis += "Examining the scoring logic for potential biases:\n"  
  
    # Analyze education scoring  
    analysis += "- Education: Scoring is based on predefined levels, which can be fair if levels are rele
```

The bottom status bar shows '2:06 PM', 'Python 3', and the date '14:07 10-09-2025'.

```
# Analyze experience scoring
analysis += "- Experience: Scoring is linear with experience, which is generally fair, assuming relevant experience."

# Analyze age scoring
analysis += "- Age: The current age bonus (25-40) introduces potential age bias. Applicants outside this range are penalized."

# Analyze gender scoring
analysis += "- Gender: Gender does not directly affect the score in this function, which is good for preventing direct gender bias in the score calculation itself."

analysis += "\nOverall: The current scoring system has a potential age bias due to the age bonus. The other factors seem more neutral in the scoring logic itself."

return analysis

# Dynamic Usage:
while True:
    print("\n--- Enter Applicant Details ---")
    education = input("Enter applicant's education level (high school, bachelor, master, phd, or 'quit' to exit): ")
    if education.lower() == 'quit':
        break
    try:
        experience = int(input("Enter applicant's years of experience: "))
        age = int(input("Enter applicant's age: "))
    except ValueError:
        print("Invalid input. Please enter numbers for experience and age.")
        continue
    gender = input("Enter applicant's gender: ")

    applicant_score = score_applicant(education, experience, gender, age)
    print(f"Applicant Score: {applicant_score}")

    # Analyze the scoring system after getting some inputs (you can analyze it once or after each applicant)
    bias_analysis = analyze_scoring_bias(score_applicant)
    print(bias_analysis)
```

```
gender = input("Enter applicant's gender: ")

applicant_score = score_applicant(education, experience, gender, age)
print(f"Applicant Score: {applicant_score}")

# Analyze the scoring system after getting some inputs (you can analyze it once or after each applicant)
bias_analysis = analyze_scoring_bias(score_applicant)
print(bias_analysis)

--- Enter Applicant Details ---
Enter applicant's education level (high school, bachelor, master, phd, or 'quit' to exit): bachelor
Enter applicant's years of experience: 0
Enter applicant's age: 21
Enter applicant's gender: female
Applicant Score: 20

--- Enter Applicant Details ---
Enter applicant's education level (high school, bachelor, master, phd, or 'quit' to exit): quit
--- Bias Analysis ---
Examining the scoring logic for potential biases:
- Education: Scoring is based on predefined levels, which can be fair if levels are relevant to the job.
- Experience: Scoring is linear with experience, which is generally fair, assuming relevant experience.
- Age: The current age bonus (25-40) introduces potential age bias. Applicants outside this range are penalized.
- Gender: Gender does not directly affect the score in this function, which is good for preventing direct gender bias in the score calculation itself.

Overall: The current scoring system has a potential age bias due to the age bonus. The other factors seem more neutral in the scoring logic itself.
```

Canvas Login | Instructu... Assignment5.2 Login system security rev... google.colab notebook aia5.2 - Colab

canvas.instructure.com/courses/12513912/assignments/57523475

Task Description#5 (Inclusiveness)

- Code Snippet

```
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

ENG IN 14:15 10-09-2025

Prompt:

snippet code:

```
def greet_user(name,gender):
```

```
if gender.lower()=="male":
```

```
title="mr."
```

```
else:title="mrs."
```

```
return f"hello,{title}{name}!welcome."
```

now regenerate code for gender neutral also

Canvas Login | Instructional Technology | Assignment5.2 | aiac5.2 - Colab | Login system security review | google colab notebook - x | + | - | X

colab.research.google.com/drive/1nNlIZOG4h1vmc4DAgOO8DCfj1I4i-qk#scrollTo=67109b57

aiac5.2

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Files

- sample_data
- Screenshot 2025-09-10 141549.png

Disk 68.75 GB available

Gemini

```
[11] ✓ Bs
def greet_user_neutral(name):
    """
    Greet a user in a completely gender-neutral way using only their name.

    Args:
        name (str): The name of the user.
    """
    return f"Hello, {name}! Welcome."

# Get user input for the name
user_name = input("Please enter your name: ")

# Example Usage:
print(greet_user_neutral(user_name))
```

Please enter your name: amamamamama
Hello, amamamamama! Welcome.

Variables Terminal

2:27 PM Python 3

ENG IN 14:27 10-09-2025

