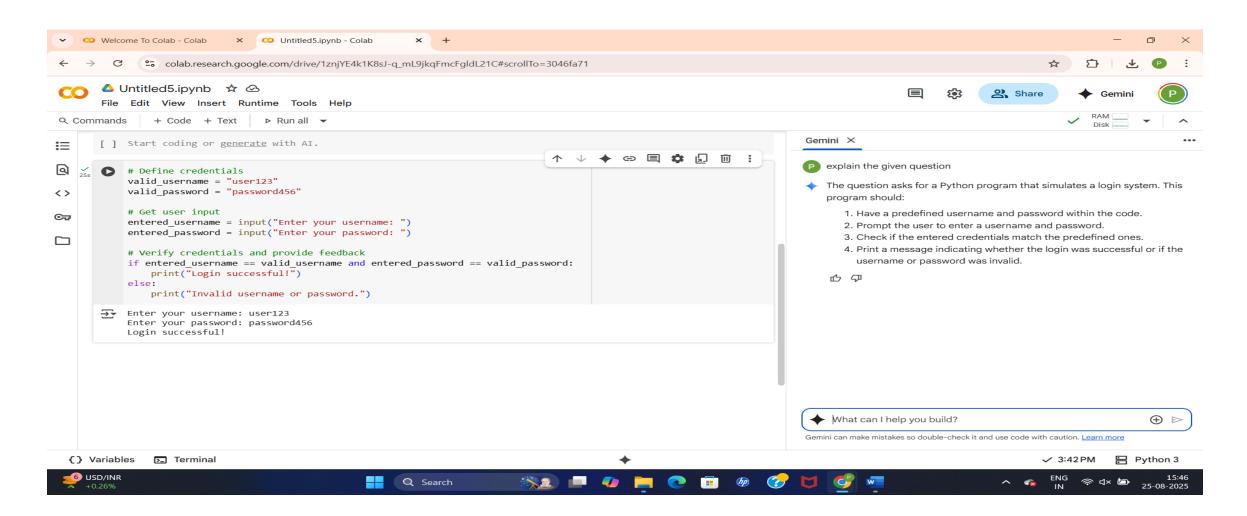
## AI ASSISTED CODING LAB 5

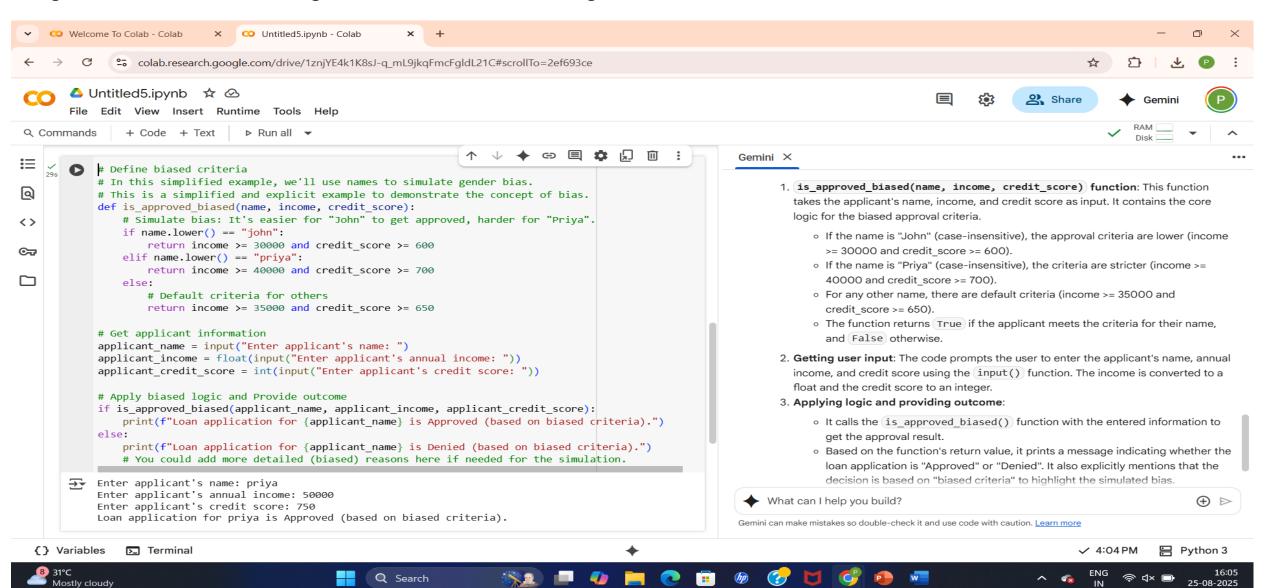
## Task Description#1 (Privacy and Data Security)

• 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.



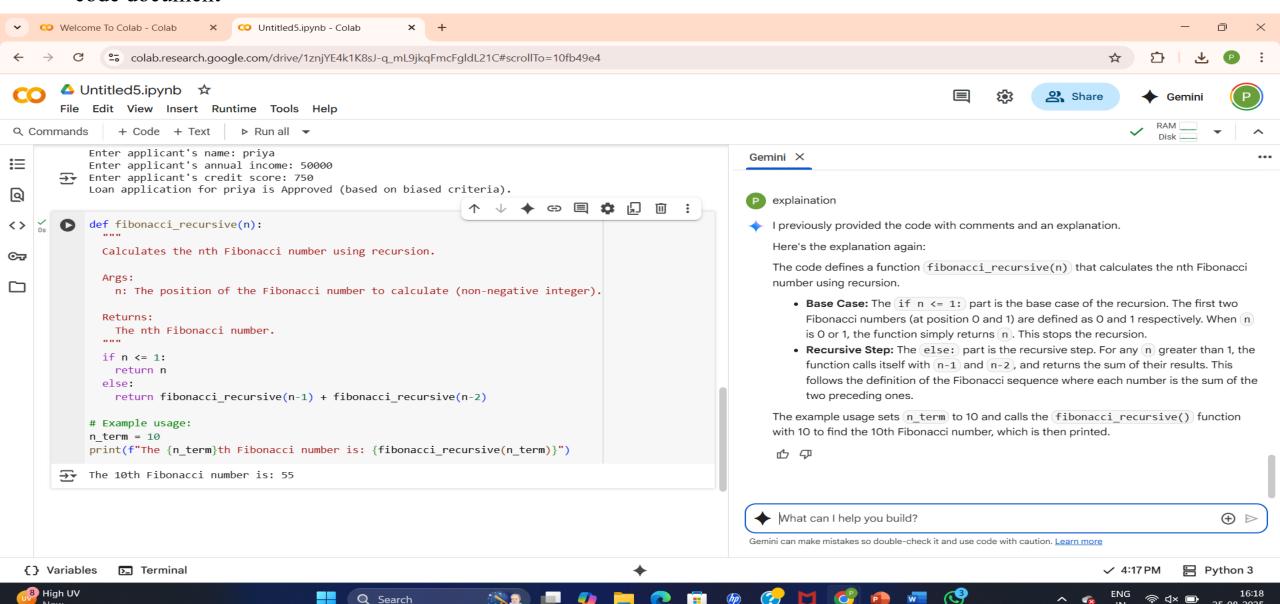
## Task Description#2 (Bias)

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



## Task Description#3 (Transparency)

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



4.Ask to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age).

```
# 1. Kepresent Applicant Data
class Applicant:
   def __init__(self, education, experience, gender, age):
       self.education = education # e.g., 'High School', 'Bachelor', 'Master', 'PhD'
       self.experience = experience # in years
       self.gender = gender # e.g., 'Male', 'Female', 'Other'
       self.age = age
                        # in years
# 2. Define Scoring Logic
# This is a simple example; you can adjust the weights and scoring based on your needs.
education scores = {
    'High School': 1,
   'Bachelor': 3,
    'Master': 5,
    'PhD': 7
# Scoring based on experience (linear scale)
experience weight = 2
# Scoring based on age (you might penalize very young or very old applicants)
# This is a simple example: points for a certain age range
def score age(age):
   if 25 <= age <= 45:
       return 5
   elif 20 <= age < 25 or 45 < age <= 55:
       return 3
    else:
        notunn 1
```

```
return 1
0
    # Gender could be neutral or have a slight bias depending on the job (be mindful of fairness and bias)
    # In this example, gender is neutral
    gender_scores = {
        'Male': 0,
        'Female': 0,
        'Other': 0
    # 3. Implement Scoring Function
    def calculate_score(applicant):
        score = 0
        # Add score for education
        score += education_scores.get(applicant.education, 0)
        # Add score for experience
        score += applicant.experience * experience_weight
        # Add score for age
        score += score_age(applicant.age)
        # Add score for gender (neutral in this case)
        score += gender_scores.get(applicant.gender, 0)
        return score
```

```
# 4. Apply Scoring to Applicants
# Create some sample applicants
applicant1 = Applicant("Bachelor", 5, "Female", 30)
applicant2 = Applicant("Master", 10, "Male", 40)
applicant3 = Applicant("High School", 2, "Other", 22)
applicant4 = Applicant("PhD", 15, "Female", 50)
applicants = [applicant1, applicant2, applicant3, applicant4]
# Calculate and store scores
applicant_scores = {}
for i, applicant in enumerate(applicants):
    score = calculate_score(applicant)
    applicant_scores[f"Applicant {i+1}"] = score
# 5. Display Results
print("Job Applicant Scores:")
for applicant, score in applicant_scores.items():
    print(f"{applicant}: {score}")
```

Job Applicant Scores:
Applicant 1: 18
Applicant 2: 30
Applicant 3: 8
Applicant 4: 40

5.write a program in python generate before the name for male is a MR for female is a MS for neutral is MX

