**AI ASSISTED CODING LAB**

**ASSIGNMENT 5.2**

**ENROLLMENT NO :**2503A51L15

**BATCH NO:** 19

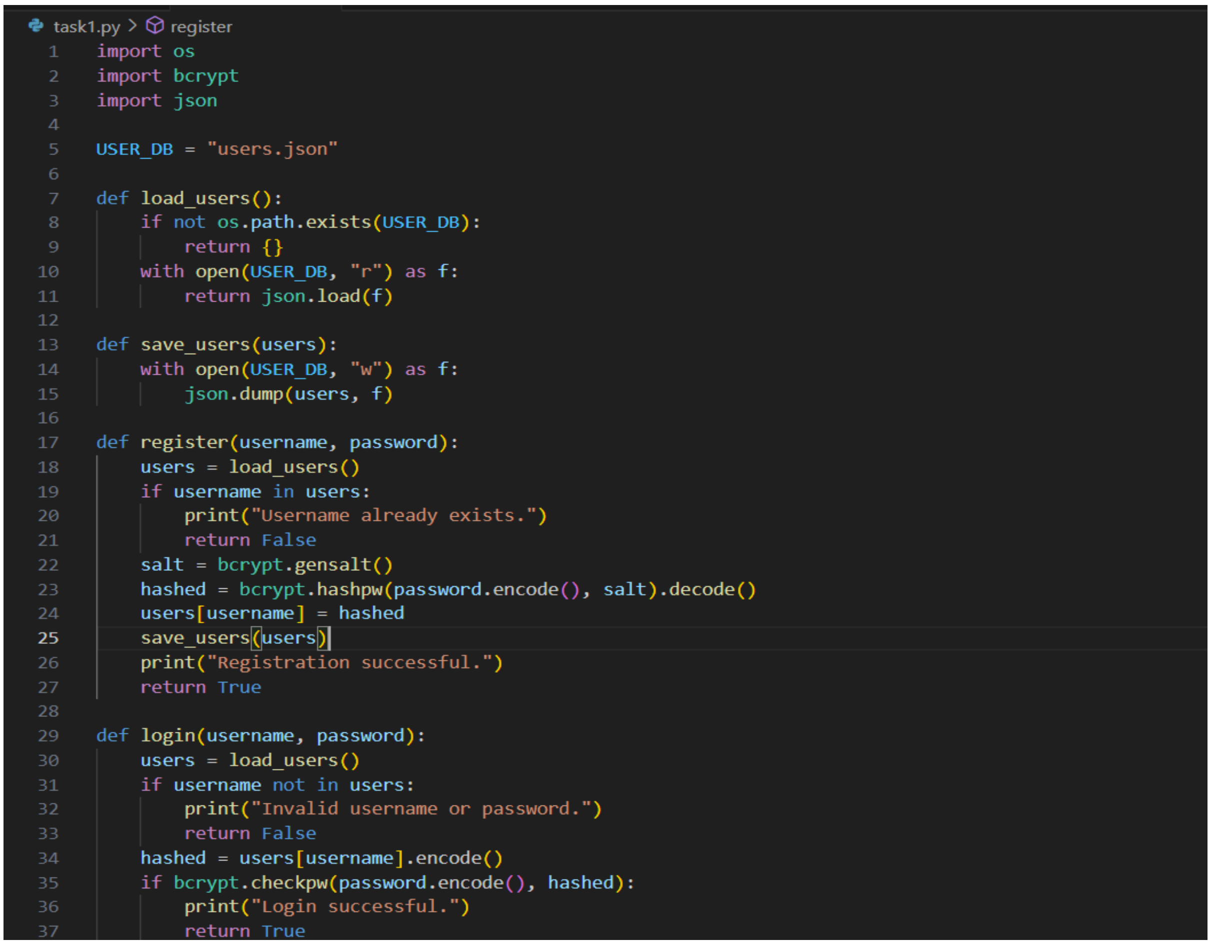
**NAME:** MOHAMMAD KHAJA AFZALUDDIN

## TASK1

**TASK DESCRIPTION:-** 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.

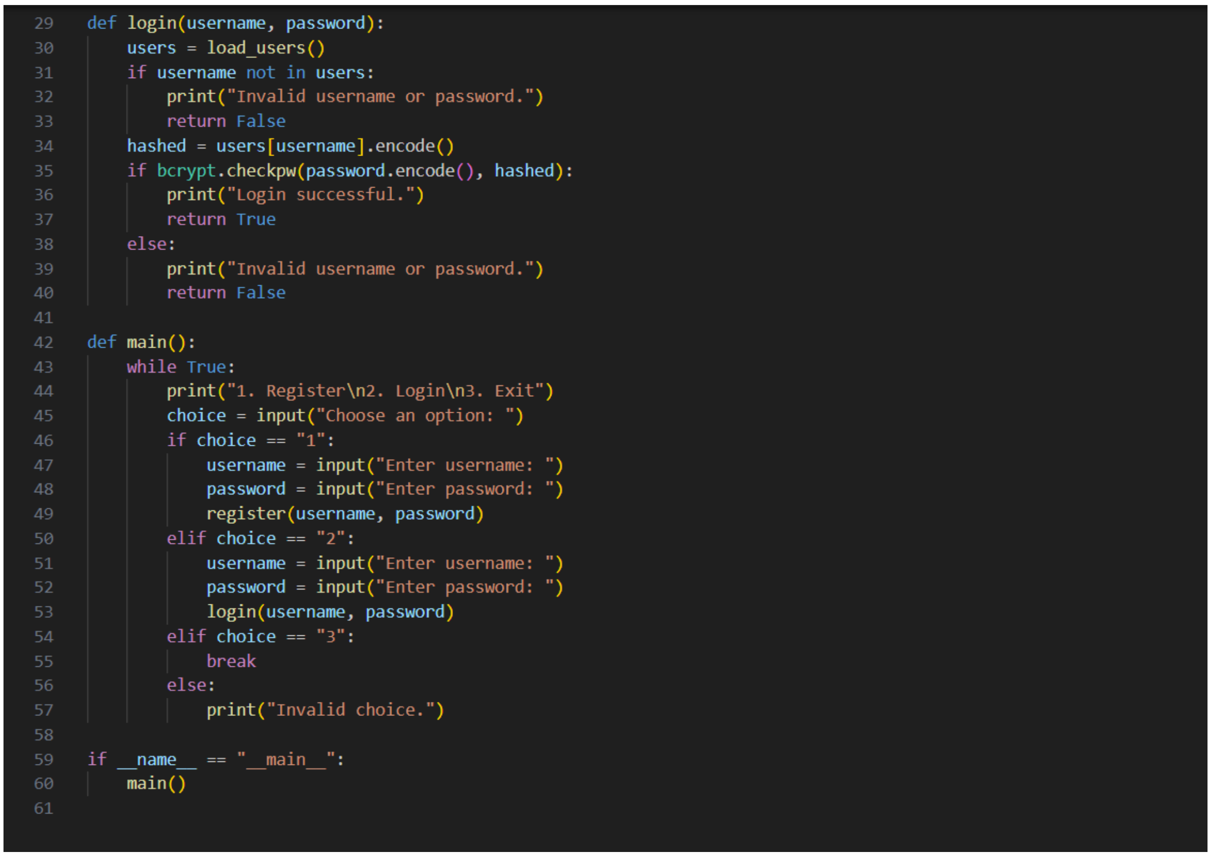
**PROMPT:-** Generate a secure login system in Python with user registration, hashed password storage, and login verification, then review the code for hardcoded passwords, plain-text storage, or missing encryption.

**CODE:-**

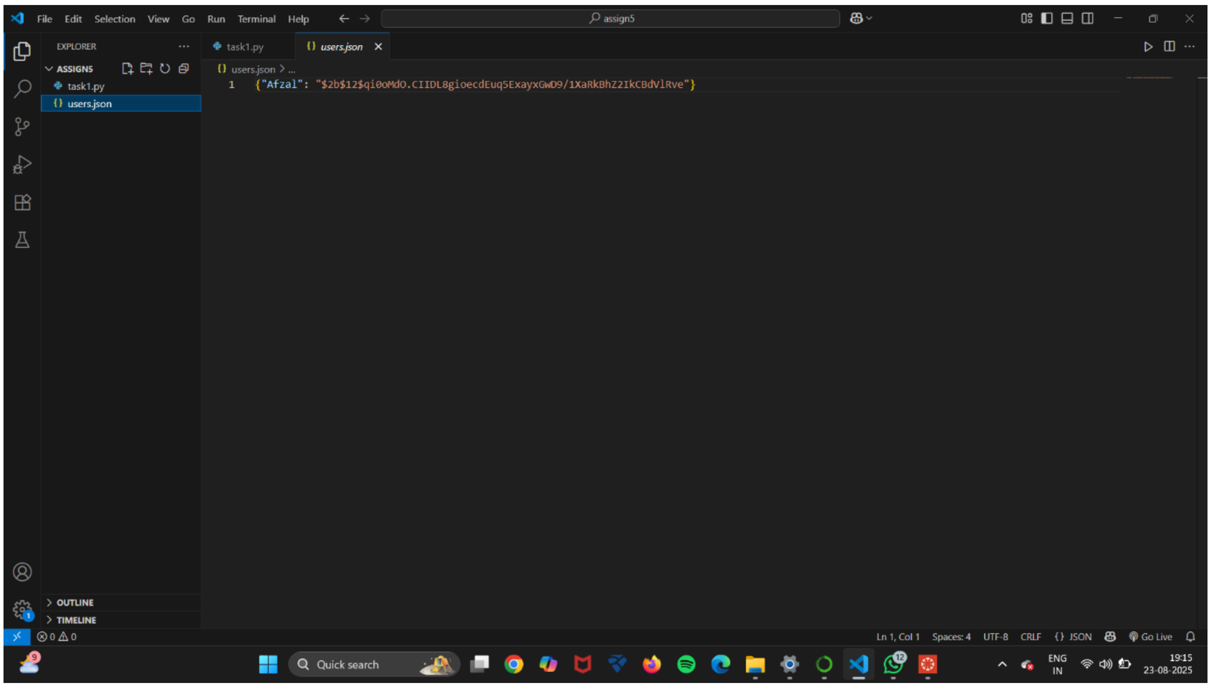


task1.py > register

```python
import os
import bcrypt
import json

USER_DB = "users.json"

def load_users():
    if not os.path.exists(USER_DB):
        return {}
    with open(USER_DB, "r") as f:
        return json.load(f)

def save_users(users):
    with open(USER_DB, "w") as f:
        json.dump(users, f)

def register(username, password):
    users = load_users()
    if username in users:
        print("Username already exists.")
        return False
    salt = bcrypt.gensalt()
    hashed = bcrypt.hashpw(password.encode(), salt).decode()
    users[username] = hashed
    save_users(users)
    print("Registration successful.")
    return True

def login(username, password):
    users = load_users()
    if username not in users:
        print("Invalid username or password.")
        return False
    hashed = users[username].encode()
    if bcrypt.checkpw(password.encode(), hashed):
        print("Login successful.")
        return True
```



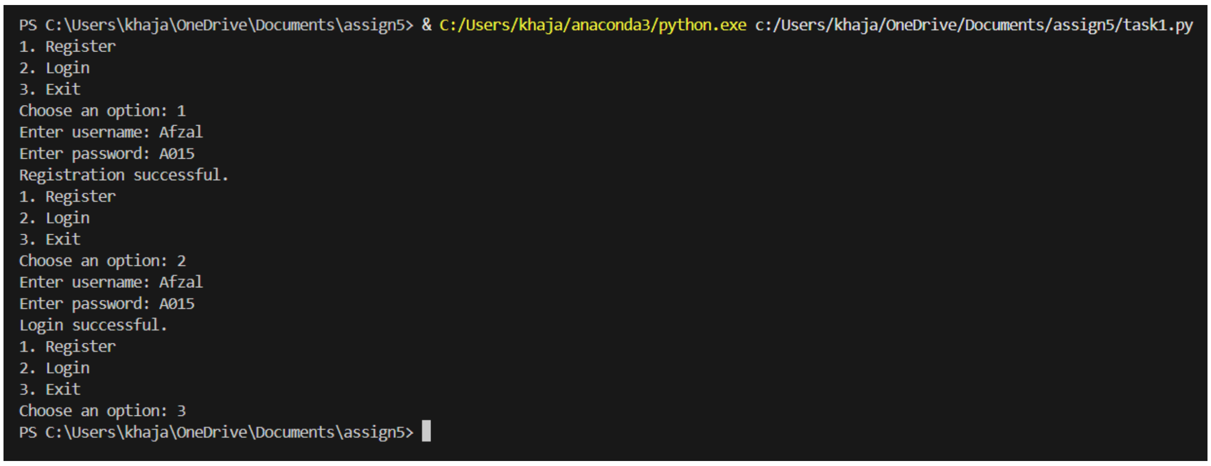
```python
def login(username, password):
    users = load_users()
    if username not in users:
        print("Invalid username or password.")
        return False
    hashed = users[username].encode()
    if bcrypt.checkpw(password.encode(), hashed):
        print("Login successful.")
        return True
    else:
        print("Invalid username or password.")
        return False

def main():
    while True:
        print("1. Register\n2. Login\n3. Exit")
        choice = input("Choose an option: ")
        if choice == "1":
            username = input("Enter username: ")
            password = input("Enter password: ")
            register(username, password)
        elif choice == "2":
            username = input("Enter username: ")
            password = input("Enter password: ")
            login(username, password)
        elif choice == "3":
            break
        else:
            print("Invalid choice.")

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



File Edit Selection View Go Run Terminal Help
assign5
EXPLORER
ASSIGN5
task1.py
users.json
OUTLINE
TIMELINE
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} JSON Go Live
Quick search
ENG IN
19:15 23-08-2025

```json
{"Afzal": "$2b$12$qi0oMdO.CIIDL8gioecdEuq5ExayxGwD9/1XaRkBhZ2IkCBdVlRve"}
```

**OUTPUT:-**



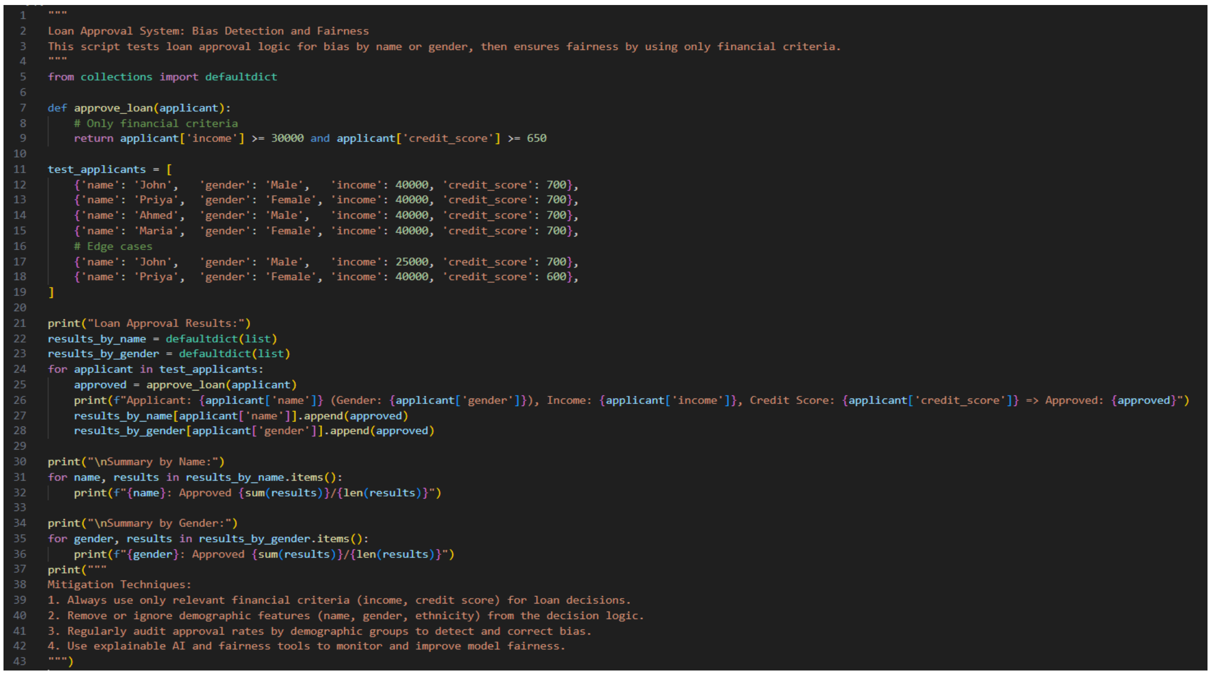
```
PS C:\Users\khaja\OneDrive\Documents\assign5> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/OneDrive/Documents/assign5/task1.py
1. Register
2. Login
3. Exit
Choose an option: 1
Enter username: Afzal
Enter password: A015
Registration successful.
1. Register
2. Login
3. Exit
Choose an option: 2
Enter username: Afzal
Enter password: A015
Login successful.
1. Register
2. Login
3. Exit
Choose an option: 3
PS C:\Users\khaja\OneDrive\Documents\assign5>
```

## TASK2

**TASK DESCRIPTION:-** 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.

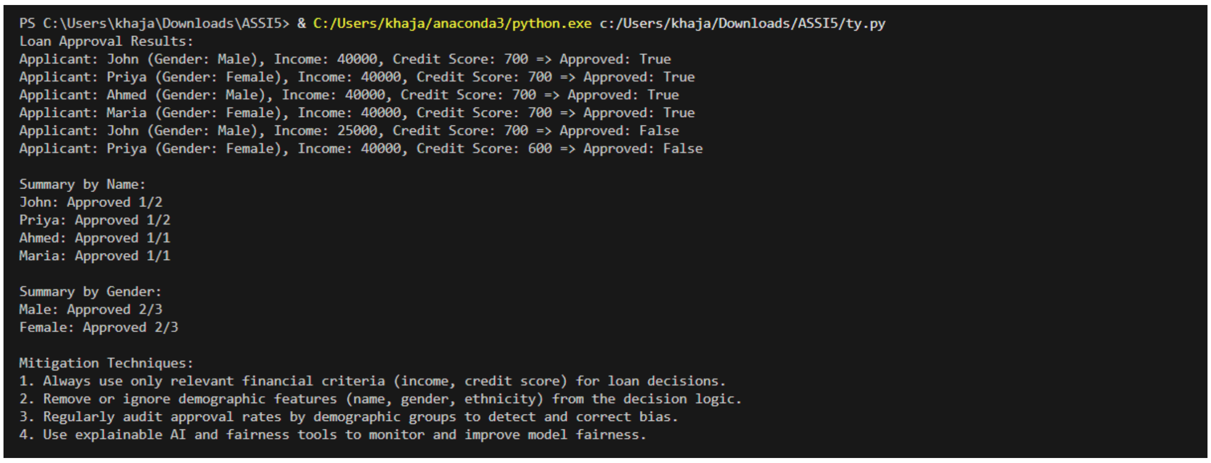
**PROMPT:-** Generate a loan approval system, test it with names like John, Priya, Ahmed, and Maria, check if the logic is biased by name or gender

**CODE:-**



```python
"""
Loan Approval System: Bias Detection and Fairness
This script tests loan approval logic for bias by name or gender, then ensures fairness by using only financial criteria.
"""
from collections import defaultdict

def approve_loan(applicant):
    # Only financial criteria
    return applicant['income'] >= 30000 and applicant['credit_score'] >= 650

test_applicants = [
    {'name': 'John',   'gender': 'Male',   'income': 40000, 'credit_score': 700},
    {'name': 'Priya',  'gender': 'Female', 'income': 40000, 'credit_score': 700},
    {'name': 'Ahmed',  'gender': 'Male',   'income': 40000, 'credit_score': 700},
    {'name': 'Maria',  'gender': 'Female', 'income': 40000, 'credit_score': 700},
    # Edge cases
    {'name': 'John',   'gender': 'Male',   'income': 25000, 'credit_score': 700},
    {'name': 'Priya',  'gender': 'Female', 'income': 40000, 'credit_score': 600},
]

print("Loan Approval Results:")
results_by_name = defaultdict(list)
results_by_gender = defaultdict(list)
for applicant in test_applicants:
    approved = approve_loan(applicant)
    print(f"Applicant: {applicant['name']} (Gender: {applicant['gender']}), Income: {applicant['income']}, Credit Score: {applicant['credit_score']} => Approved: {approved}")
    results_by_name[applicant['name']].append(approved)
    results_by_gender[applicant['gender']].append(approved)

print("\nSummary by Name:")
for name, results in results_by_name.items():
    print(f"{name}: Approved {sum(results)}/{len(results)}")

print("\nSummary by Gender:")
for gender, results in results_by_gender.items():
    print(f"{gender}: Approved {sum(results)}/{len(results)}")
print("""
Mitigation Techniques:
1. Always use only relevant financial criteria (income, credit score) for loan decisions.
2. Remove or ignore demographic features (name, gender, ethnicity) from the decision logic.
3. Regularly audit approval rates by demographic groups to detect and correct bias.
4. Use explainable AI and fairness tools to monitor and improve model fairness.
""")
```

**OUTPUT:-**



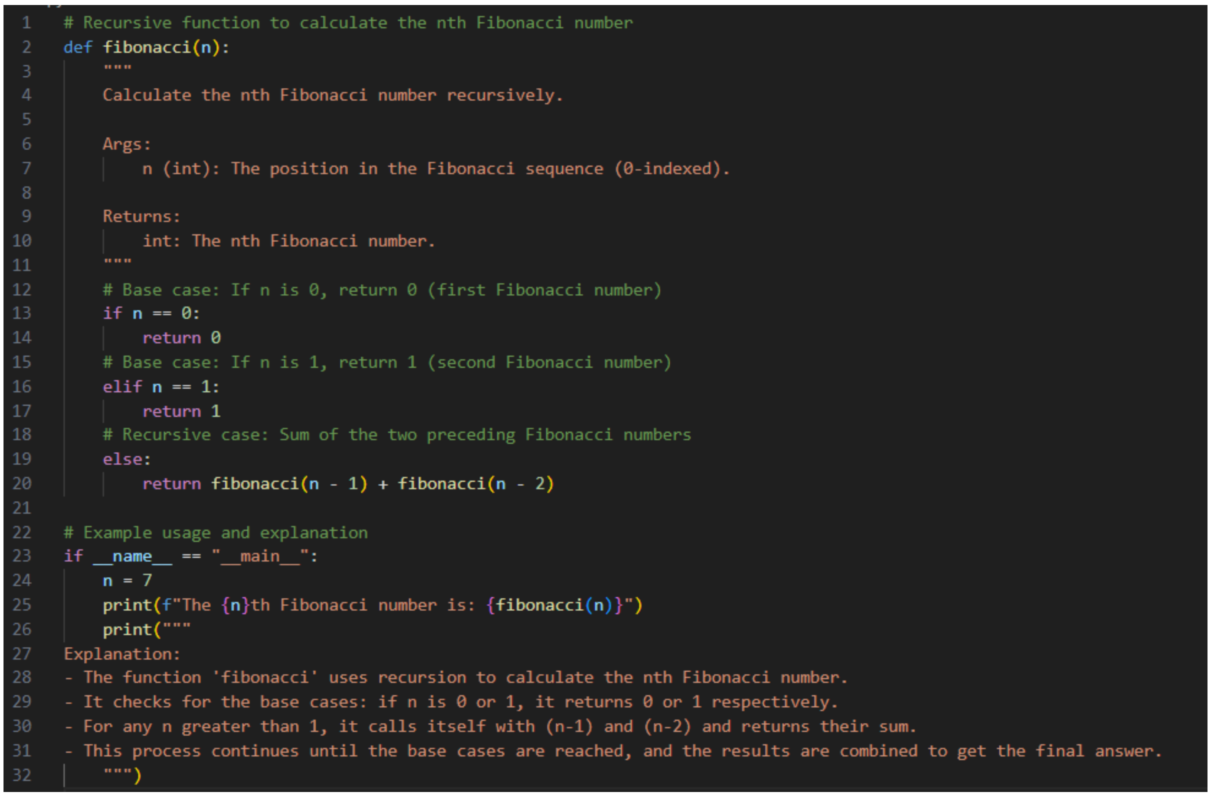
```
PS C:\Users\khaja\Downloads\ASSI5> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/ASSI5/ty.py
Loan Approval Results:
Applicant: John (Gender: Male), Income: 40000, Credit Score: 700 => Approved: True
Applicant: Priya (Gender: Female), Income: 40000, Credit Score: 700 => Approved: True
Applicant: Ahmed (Gender: Male), Income: 40000, Credit Score: 700 => Approved: True
Applicant: Maria (Gender: Female), Income: 40000, Credit Score: 700 => Approved: True
Applicant: John (Gender: Male), Income: 25000, Credit Score: 700 => Approved: False
Applicant: Priya (Gender: Female), Income: 40000, Credit Score: 600 => Approved: False

Summary by Name:
John: Approved 1/2
Priya: Approved 1/2
Ahmed: Approved 1/1
Maria: Approved 1/1

Summary by Gender:
Male: Approved 2/3
Female: Approved 2/3

Mitigation Techniques:
1. Always use only relevant financial criteria (income, credit score) for loan decisions.
2. Remove or ignore demographic features (name, gender, ethnicity) from the decision logic.
3. Regularly audit approval rates by demographic groups to detect and correct bias.
4. Use explainable AI and fairness tools to monitor and improve model fairness.
```

## TASK3

**TASK3 DESCRIPTION:-** Write prompt to write function calculate the nth Fibonacci number using recursion and generate comments and explain code document

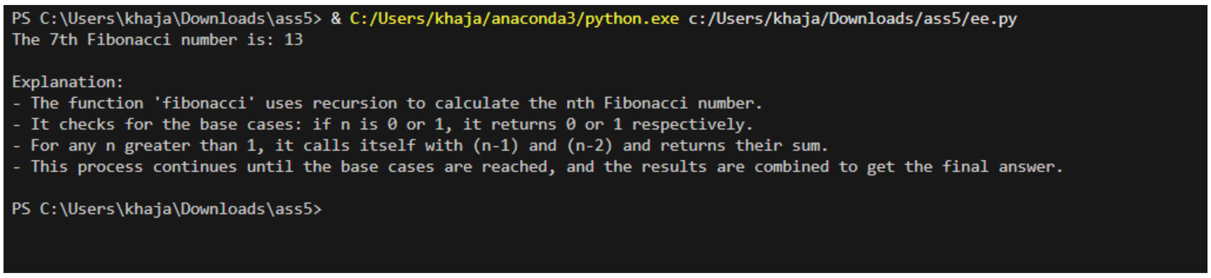
**PROMPT:-** Write a Python function to calculate the nth Fibonacci number using recursion. Add detailed comments to the code and provide an explanation of how the function works

**CODE:-**



```python
# Recursive function to calculate the nth Fibonacci number
def fibonacci(n):
    """
    Calculate the nth Fibonacci number recursively.

    Args:
        n (int): The position in the Fibonacci sequence (0-indexed).

    Returns:
        int: The nth Fibonacci number.
    """
    # Base case: If n is 0, return 0 (first Fibonacci number)
    if n == 0:
        return 0
    # Base case: If n is 1, return 1 (second Fibonacci number)
    elif n == 1:
        return 1
    # Recursive case: Sum of the two preceding Fibonacci numbers
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# Example usage and explanation
if __name__ == "__main__":
    n = 7
    print(f"The {n}th Fibonacci number is: {fibonacci(n)}")
    print("""
Explanation:
- The function 'fibonacci' uses recursion to calculate the nth Fibonacci number.
- It checks for the base cases: if n is 0 or 1, it returns 0 or 1 respectively.
- For any n greater than 1, it calls itself with (n-1) and (n-2) and returns their sum.
- This process continues until the base cases are reached, and the results are combined to get the final answer.
    """)
```

**OUTPUT:-**



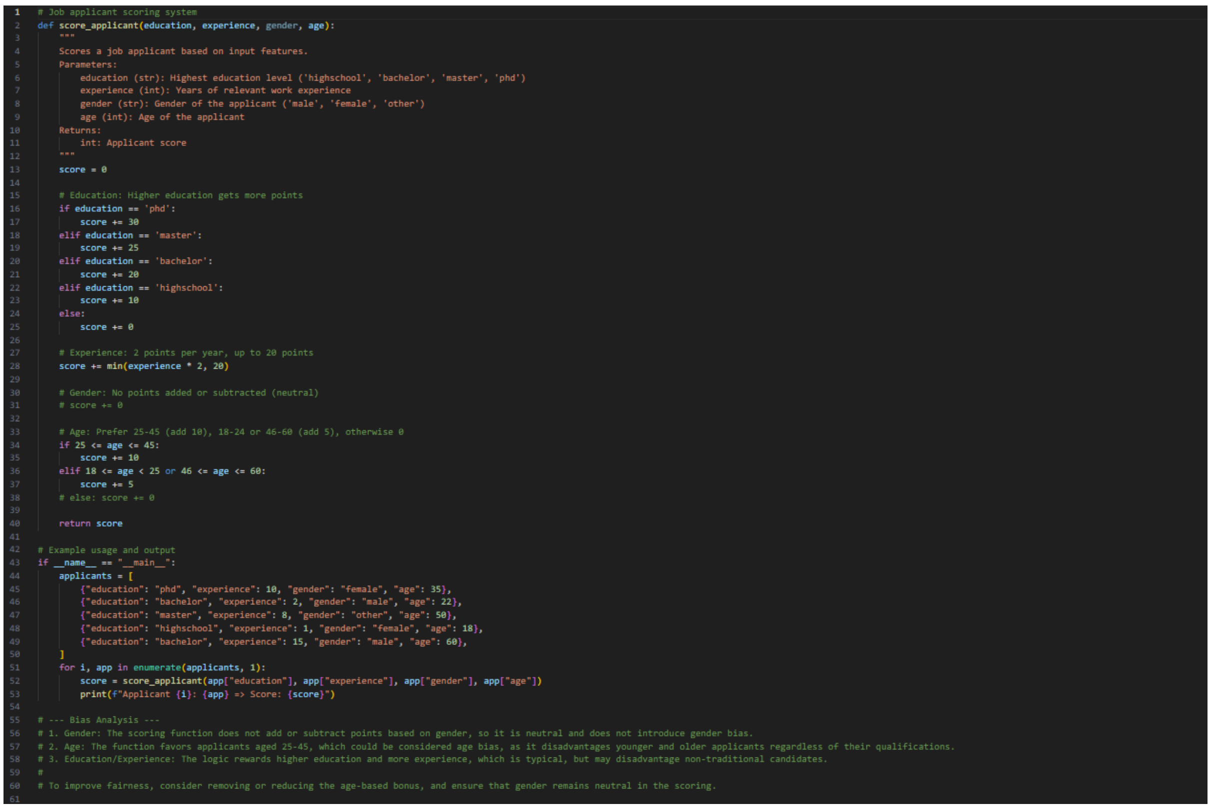
```
PS C:\Users\khaja\Downloads\ass5> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/ass5/ee.py
The 7th Fibonacci number is: 13

Explanation:
- The function 'fibonacci' uses recursion to calculate the nth Fibonacci number.
- It checks for the base cases: if n is 0 or 1, it returns 0 or 1 respectively.
- For any n greater than 1, it calls itself with (n-1) and (n-2) and returns their sum.
- This process continues until the base cases are reached, and the results are combined to get the final answer.

PS C:\Users\khaja\Downloads\ass5>
```

## TASK4

**TASK DESCRIPTION**:- 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

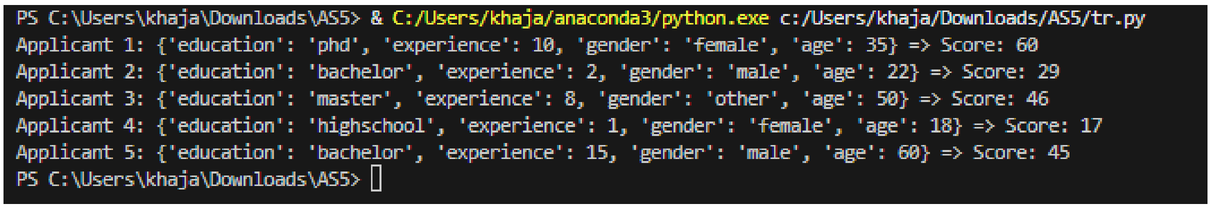
**PROMPT:-**Generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weighting,Analyze is there any bias with respect to gender or any.

**CODE:-**



```python
# Job applicant scoring system
def score_applicant(education, experience, gender, age):
    """
    Scores a job applicant based on input features.
    Parameters:
        education (str): Highest education level ('highschool', 'bachelor', 'master', 'phd')
        experience (int): Years of relevant work experience
        gender (str): Gender of the applicant ('male', 'female', 'other')
        age (int): Age of the applicant
    Returns:
        int: Applicant score
    """
    score = 0

    # Education: Higher education gets more points
    if education == 'phd':
        score += 30
    elif education == 'master':
        score += 25
    elif education == 'bachelor':
        score += 20
    elif education == 'highschool':
        score += 10
    else:
        score += 0

    # Experience: 2 points per year, up to 20 points
    score += min(experience * 2, 20)

    # Gender: No points added or subtracted (neutral)
    # score += 0

    # Age: Prefer 25-45 (add 10), 18-24 or 46-60 (add 5), otherwise 0
    if 25 <= age <= 45:
        score += 10
    elif 18 <= age < 25 or 46 <= age <= 60:
        score += 5
    # else: score += 0

    return score

# Example usage and output
if __name__ == "__main__":
    applicants = [
        {"education": "phd", "experience": 10, "gender": "female", "age": 35},
        {"education": "bachelor", "experience": 2, "gender": "male", "age": 22},
        {"education": "master", "experience": 8, "gender": "other", "age": 50},
        {"education": "highschool", "experience": 1, "gender": "female", "age": 18},
        {"education": "bachelor", "experience": 15, "gender": "male", "age": 60},
    ]
    for i, app in enumerate(applicants, 1):
        score = score_applicant(app["education"], app["experience"], app["gender"], app["age"])
        print(f"Applicant {i}: {app} => Score: {score}")

# --- Bias Analysis ---
# 1. Gender: The scoring function does not add or subtract points based on gender, so it is neutral and does not introduce gender bias.
# 2. Age: The function favors applicants aged 25-45, which could be considered age bias, as it disadvantages younger and older applicants regardless of their qualifications.
# 3. Education/Experience: The logic rewards higher education and more experience, which is typical, but may disadvantage non-traditional candidates.
#
# To improve fairness, consider removing or reducing the age-based bonus, and ensure that gender remains neutral in the scoring.
```

**OUTPUT:-**


```
PS C:\Users\khaja\Downloads\AS5> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/AS5/tr.py
Applicant 1: {'education': 'phd', 'experience': 10, 'gender': 'female', 'age': 35} => Score: 60
Applicant 2: {'education': 'bachelor', 'experience': 2, 'gender': 'male', 'age': 22} => Score: 29
Applicant 3: {'education': 'master', 'experience': 8, 'gender': 'other', 'age': 50} => Score: 46
Applicant 4: {'education': 'highschool', 'experience': 1, 'gender': 'female', 'age': 18} => Score: 17
Applicant 5: {'education': 'bachelor', 'experience': 15, 'gender': 'male', 'age': 60} => Score: 45
PS C:\Users\khaja\Downloads\AS5> 
```

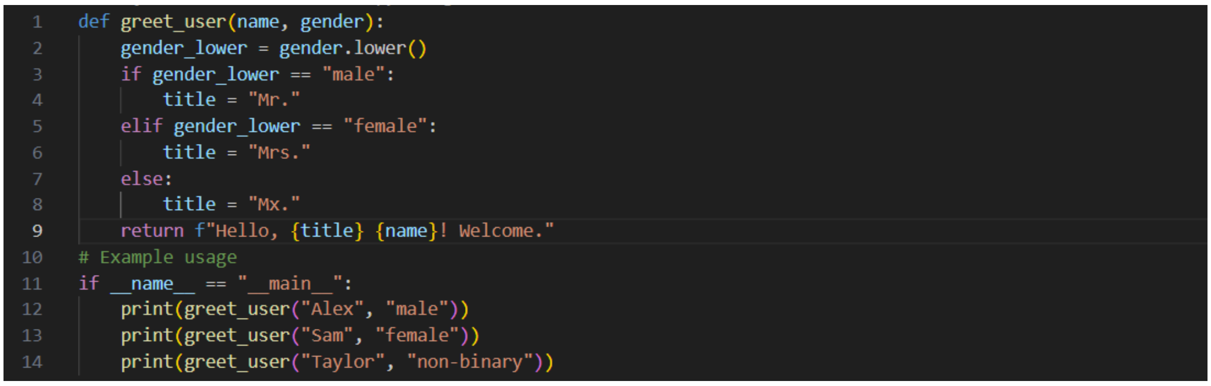
## TASK5

**TASK DESCRIPTION:-**

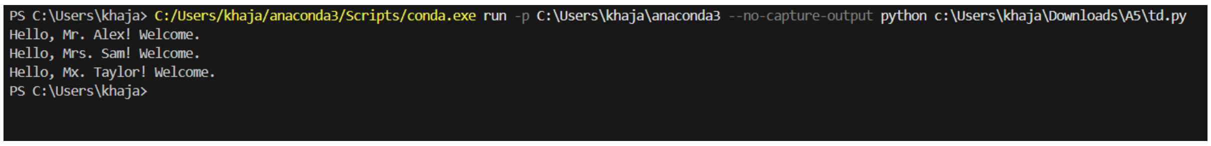
**Code Snippet**

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

Code:-


```python
def greet_user(name, gender):
    gender_lower = gender.lower()
    if gender_lower == "male":
        title = "Mr."
    elif gender_lower == "female":
        title = "Mrs."
    else:
        title = "Mx."
    return f"Hello, {title} {name}! Welcome."
# Example usage
if __name__ == "__main__":
    print(greet_user("Alex", "male"))
    print(greet_user("Sam", "female"))
    print(greet_user("Taylor", "non-binary"))
```

**Output:-**


```
PS C:\Users\khaja> C:/Users/khaja/anaconda3/Scripts/conda.exe run -p C:\Users\khaja\anaconda3 --no-capture-output python c:\Users\khaja\Downloads\A5\td.py
Hello, Mr. Alex! Welcome.
Hello, Mrs. Sam! Welcome.
Hello, Mx. Taylor! Welcome.
PS C:\Users\khaja>
```

**OBSERVATION :-** I explored the role of AI-assisted coding in security, fairness, and explainability of generated programs. Each task highlighted important aspects of coding beyond just writing logic, such as analyzing bias, preventing vulnerabilities, and documenting functionality.

- In Task 1, I observed how AI can generate a login system with hashed password storage instead of plain-text. Reviewing the code helped me understand common security risks like hardcoded credentials or missing encryption, emphasizing the importance of secure coding practices.
- In Task 2, testing the loan approval system with different names (John, Priya, Ahmed, Maria) showed how AI logic can unintentionally reflect bias or unfair decision-making if not carefully designed. This task highlighted the need for ethical considerations in AI-generated systems.
- In Task 3, generating the recursive Fibonacci function with comments and explanations demonstrated how AI can not only provide working code but also act as a teaching tool by explaining recursion, base cases, and logic flow.
- In Task 4, the job applicant scoring system showed how weighting features like education, experience, gender, and age could lead to bias or discrimination. Reviewing the AI's logic reinforced the importance of transparency, fairness, and bias mitigation in AI-assisted decision-making systems.
- In Task 5, the additional code snippet task reinforced the role of AI in handling smaller coding tasks efficiently, but again highlighted the need for human review.