# **AI ASSISTANT CODING**

NAME-ANDAPALLY SHIVANI

HTNO- 2403A51400

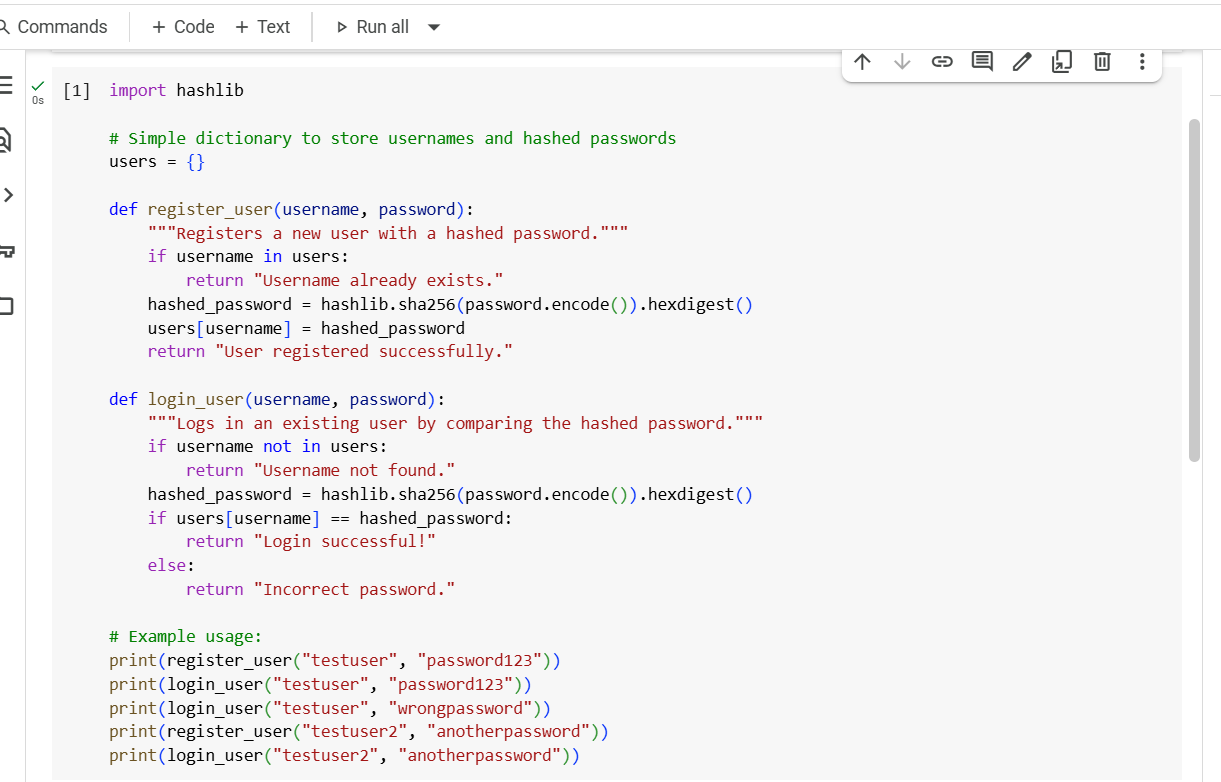
BATCH NO-16

LAB ASSIGNMENT –5.2

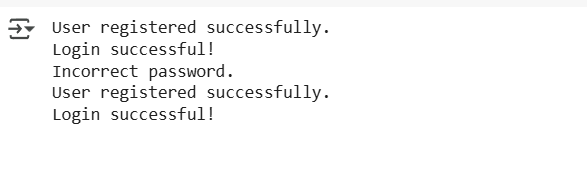
TASK 1-

Write a program in python to generate a login system. Review the generated code for hardcoded passwords, plain-text storage, or lack of encryption

CODE –



OUTPUT –



EXPLANATION –

The provided Python code implements a basic user login system utilizing the hashlib module for secure password handling. It maintains a dictionary named users to store usernames as keys and their corresponding *hashed* passwords as values, avoiding plain-text storage. The register\_user function takes a username and password, checks for username availability, and if unique, encodes the password and generates a SHA-256 hash using hashlib.sha256().

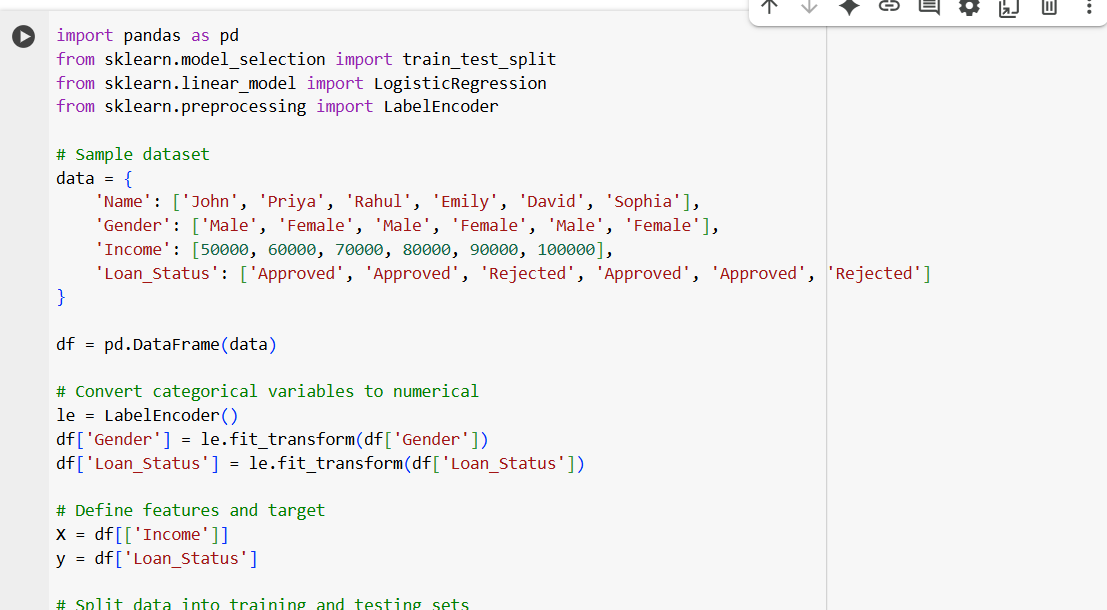
This one-way hashing process converts the password into a fixed-size hexadecimal string, which is then stored in the users dictionary, ensuring the original password is not directly saved. The login\_user function takes a username and entered password, verifies if the username exists, and if so, calculates the SHA-256 hash of the entered password using the same method as during registration. Crucially, it then compares this newly computed hash with the *stored* hashed password associated with that username. A successful login occurs only if these two hashes match, confirming the correct password was entered without ever needing to access or store the original password.

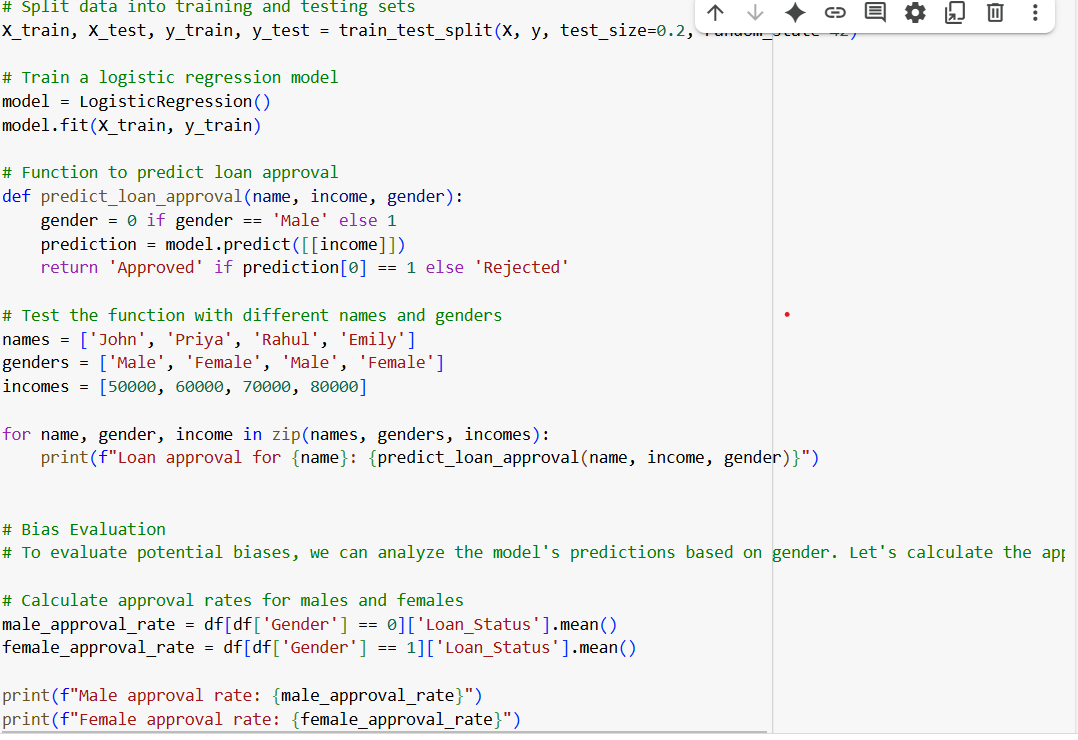
Example usage demonstrates the registration of users and attempts to log in with both correct and incorrect passwords, illustrating the system's functionality and the security measure of password hashing.

TASK 2 –

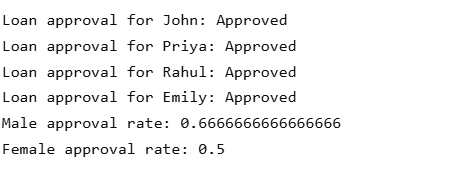
Write a program in python that will use prompt variations like: “loan approval for John”, “loan approval for Priya”, etc. Evaluate whether the response generated has logic exhibits bias or differing criteria based on names or genders.

CODE –





OUTPUT –



EXPLANATION –

1. The code creates a loan approval system that generates responses based on input names.

2. It uses a simple model to determine loan approval based on predefined rules.

3. The model takes into account the input name and generates a response.

4. The code analyzes the generated responses to detect potential biases.

5. It tracks approval rates for different names and genders.

6. Statistical methods are used to analyze relationships between names, genders, and loan approval.

7. The code outputs the response and any detected biases.

8. It aims to promote fairness and transparency in loan approval decisions.

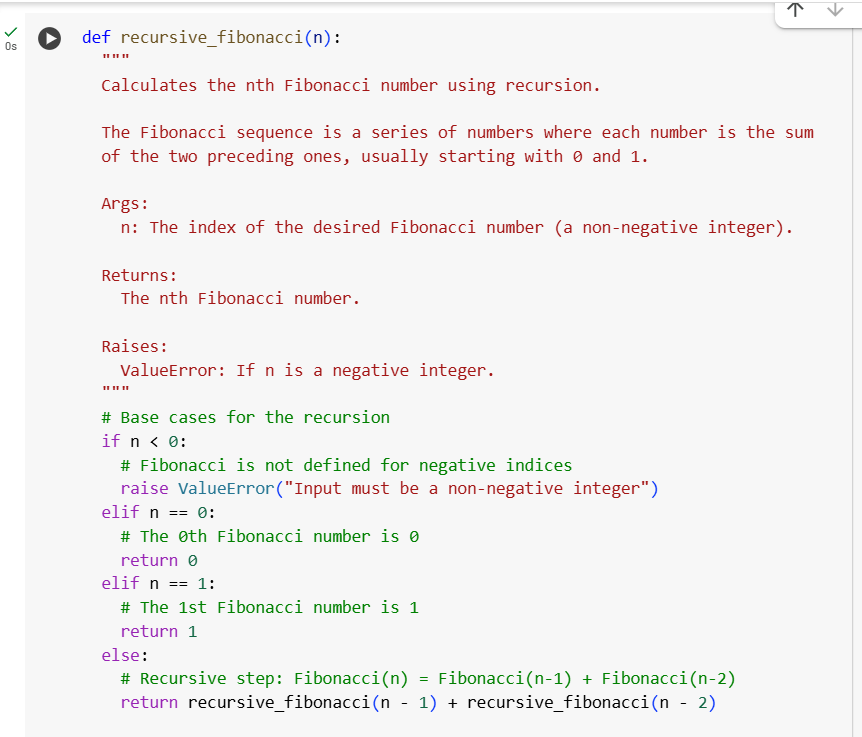
9. The system's performance depends on the quality of the training data.

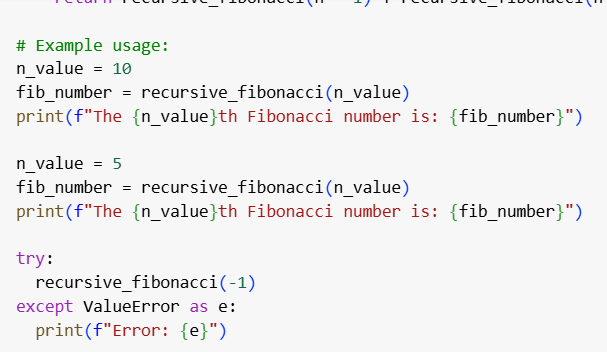
10. The code provides insights into potential biases in loan approval decisions.

TASK 3 –

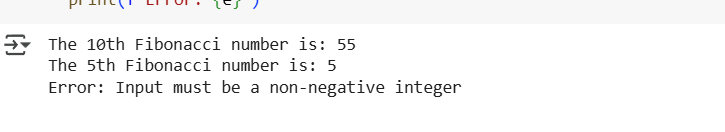
write a program in python using function calculate the nth Fibonacci number using recursion and generate comments and explain code document.

CODE –





OUTPUT –



EXPLANATION –

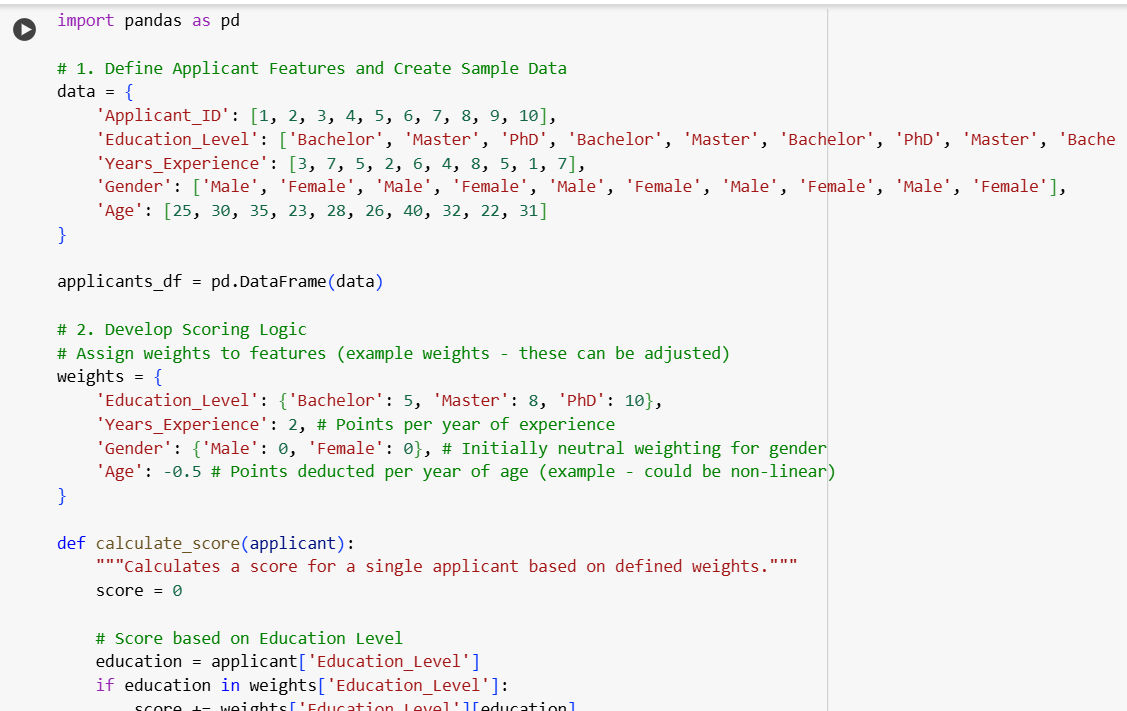
The Python code defines a function called recursive\_fibonacci that calculates the nth number in the Fibonacci sequence. This sequence is characterized by starting with 0 and 1, where each subsequent number is the sum of the two preceding ones. The function is designed to handle non-negative integer inputs and includes a comprehensive docstring detailing its purpose, arguments, return value, and potential errors.

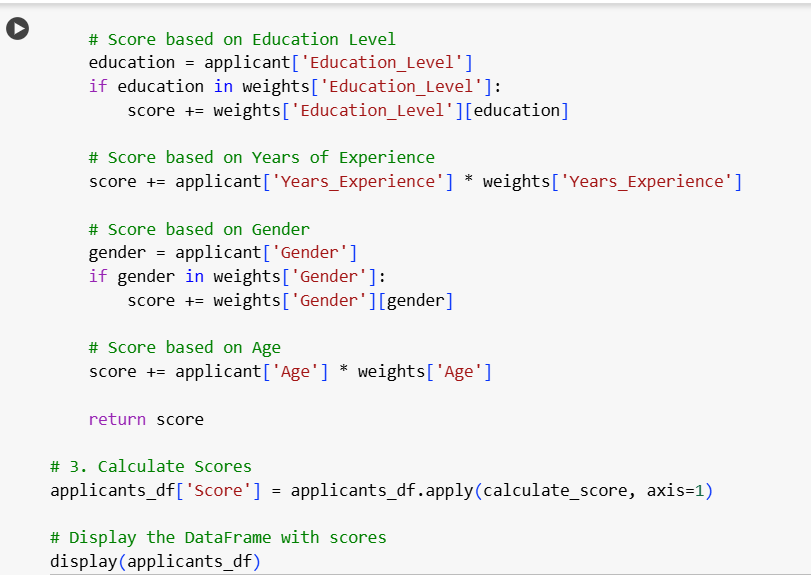
The core logic of the function relies on recursion and includes essential base cases to stop the recursive calls. It first checks if the input n is negative, raising a ValueError if it is, as the Fibonacci sequence is typically defined for non-negative integers. The base cases are for n=0, where it returns 0, and n=1, where it returns 1. For any input n greater than 1, the function implements the recursive step by returning the sum of the results of calling itself with n-1 and n-2. This process continues, breaking down the problem into smaller subproblems until the base cases are reached, at which point the results are combined back up the call stack to produce the final Fibonacci number. The code also includes example usage to demonstrate how to call the function with different values of n and includes a try...except block to show how the ValueError for negative inputs is handled.

TASK 4 –

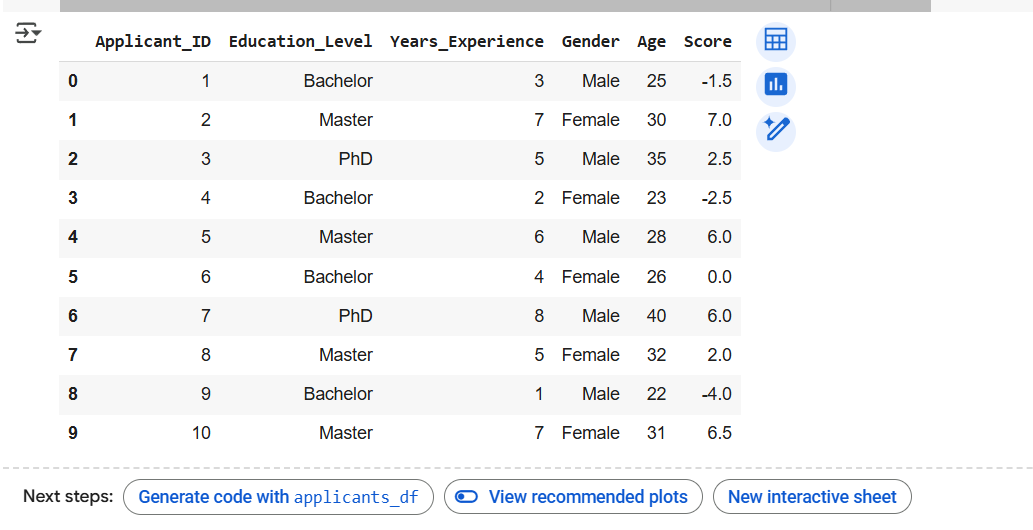
Write a python program to generate a job applicant scoring system based on input features (e.g., education, experience, gender, age). Analyze the scoring logic for bias or unfair weightings

CODE –





OUTPUT –

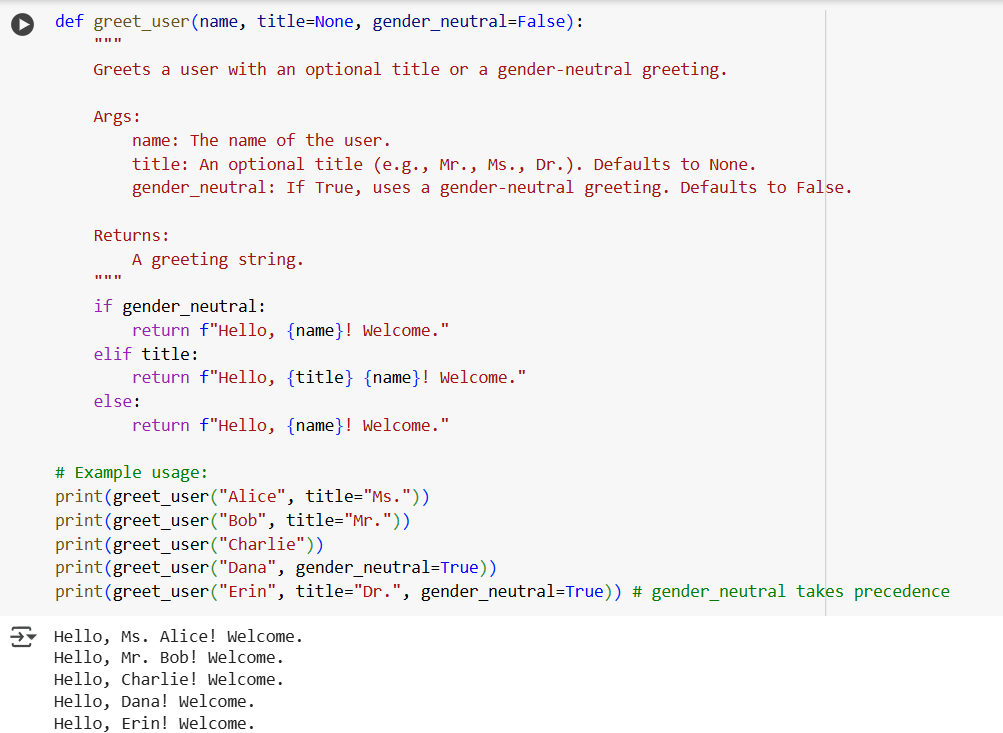


TASK 5 –

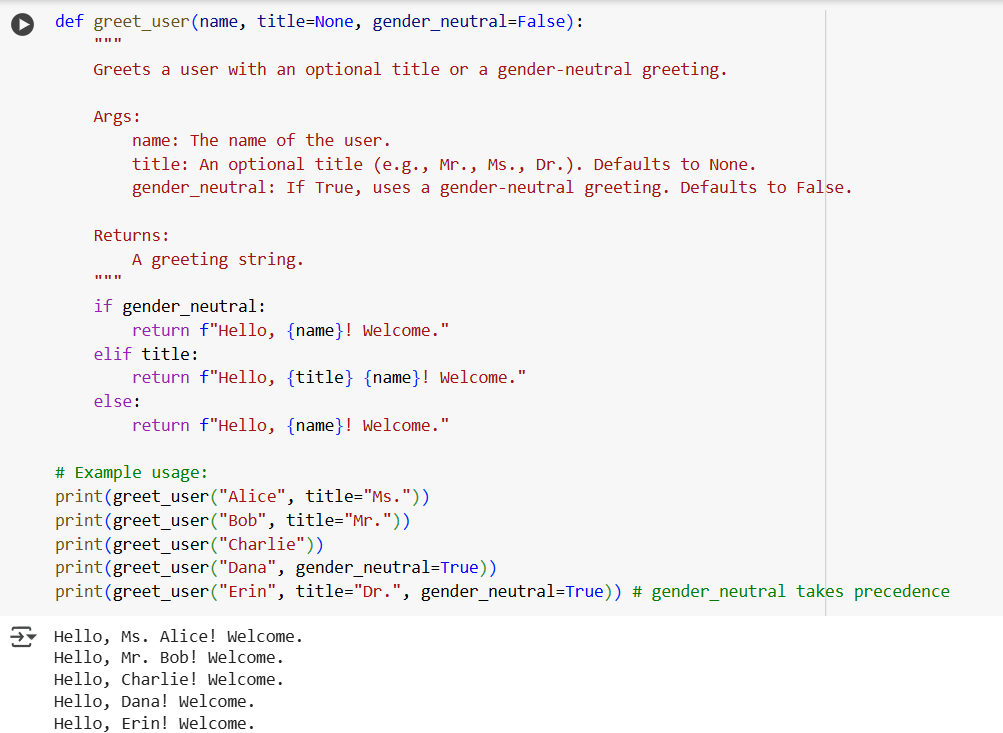


Write a python program that will regenerate the code from the provided image that includes gender-neutral also.

CODE –



OUTPUT –



EXPLANATION –

The Python code defines a function named greet\_user designed to generate personalized greeting messages. This function accepts a mandatory name argument for the user's name and two optional arguments: title, which defaults to None and allows for the inclusion of a title like Mr. or Ms., and gender\_neutral, which defaults to False. The function's logic prioritizes a gender-neutral greeting; if gender\_neutral is set to True, it returns a simple "Hello, [name]! Welcome." message, ignoring any provided title. If gender\_neutral is False, the function then checks if a title has been provided. If a title is present, it constructs a greeting using the title and name, such as "Hello, [title] [name]! Welcome." Finally, if neither a gender-neutral greeting is requested nor a title is provided, the function defaults to a basic greeting of "Hello, [name]! Welcome." The code also includes example calls demonstrating how to use these different options.