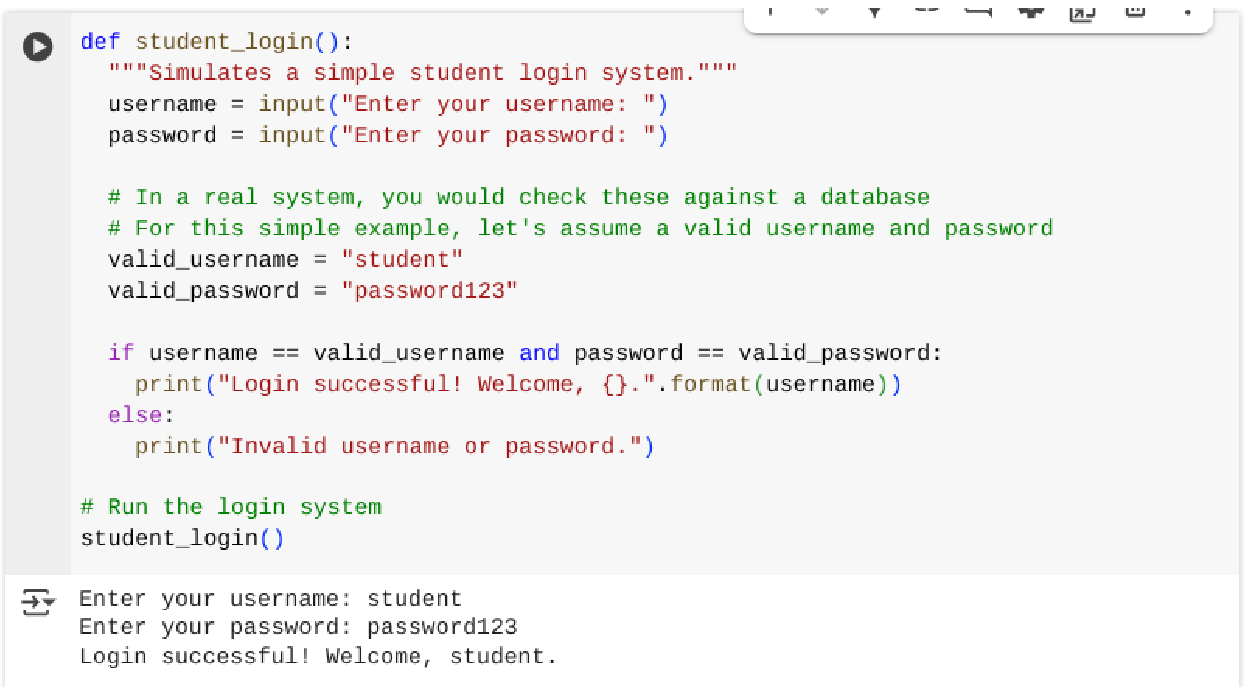
ASSIGNMENT-5.2

Task-1:

Write a python program that develops a student login system creating username and password.

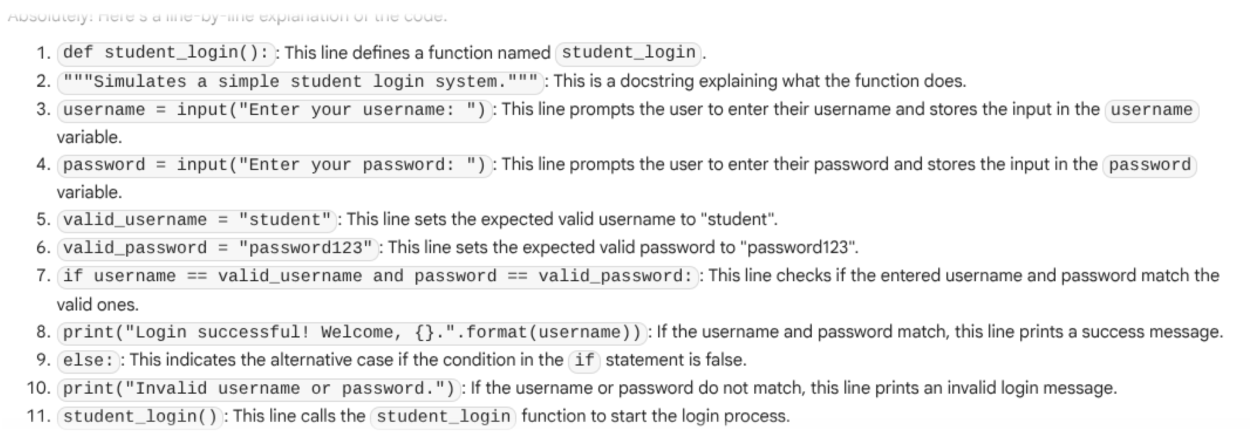
Code and Output:



```python
def student_login():
  """Simulates a simple student login system."""
  username = input("Enter your username: ")
  password = input("Enter your password: ")

  # In a real system, you would check these against a database
  # For this simple example, let's assume a valid username and password
  valid_username = "student"
  valid_password = "password123"

  if username == valid_username and password == valid_password:
    print("Login successful! Welcome, {}.".format(username))
  else:
    print("Invalid username or password.")

# Run the login system
student_login()
```

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

Explanation:



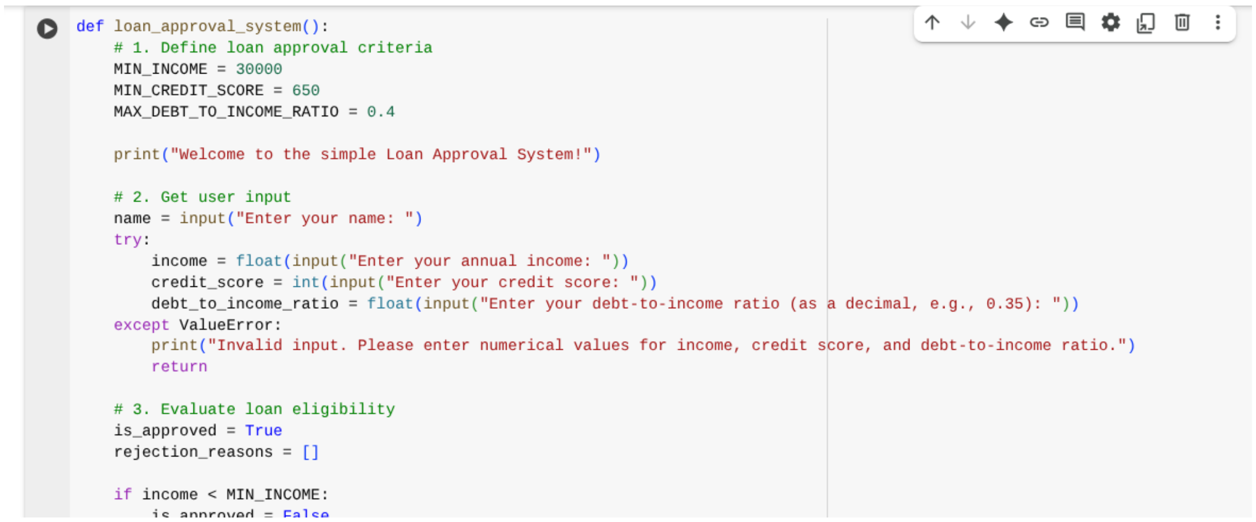
Absolutely! Here's a line-by-line explanation of the code:

1. `def student_login():` : This line defines a function named `student_login`.
2. `"""Simulates a simple student login system."""` : This is a docstring explaining what the function does.
3. `username = input("Enter your username: ")` : This line prompts the user to enter their username and stores the input in the `username` variable.
4. `password = input("Enter your password: ")` : This line prompts the user to enter their password and stores the input in the `password` variable.
5. `valid_username = "student"` : This line sets the expected valid username to "student".
6. `valid_password = "password123"` : This line sets the expected valid password to "password123".
7. `if username == valid_username and password == valid_password:` : This line checks if the entered username and password match the valid ones.
8. `print("Login successful! Welcome, {}.".format(username))` : If the username and password match, this line prints a success message.
9. `else:` : This indicates the alternative case if the condition in the `if` statement is false.
10. `print("Invalid username or password.")` : If the username or password do not match, this line prints an invalid login message.
11. `student_login()` : This line calls the `student_login` function to start the login process.

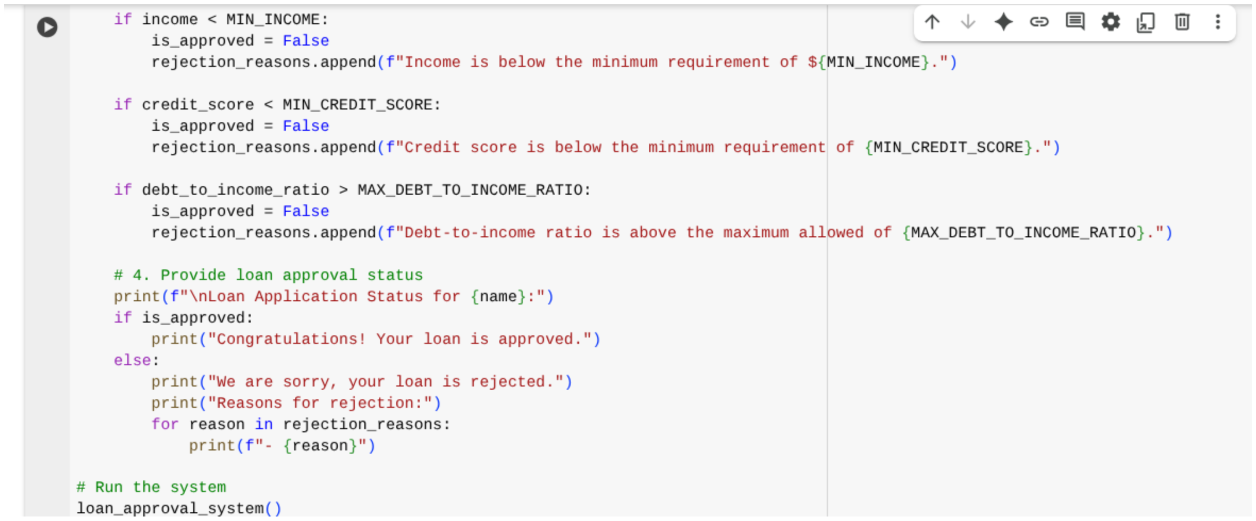
Task-2:

Write a program in python to develop a simple loan apporaval system with name,income, credit works and debt to income ratio in input.

Code:

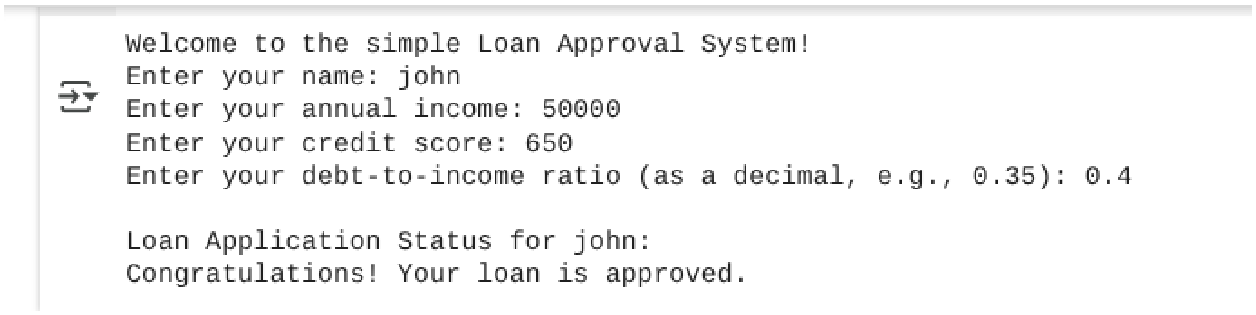


```python
def loan_approval_system():
    # 1. Define loan approval criteria
    MIN_INCOME = 30000
    MIN_CREDIT_SCORE = 650
    MAX_DEBT_TO_INCOME_RATIO = 0.4

    print("Welcome to the simple Loan Approval System!")

    # 2. Get user input
    name = input("Enter your name: ")
    try:
        income = float(input("Enter your annual income: "))
        credit_score = int(input("Enter your credit score: "))
        debt_to_income_ratio = float(input("Enter your debt-to-income ratio (as a decimal, e.g., 0.35): "))
    except ValueError:
        print("Invalid input. Please enter numerical values for income, credit score, and debt-to-income ratio.")
        return

    # 3. Evaluate loan eligibility
    is_approved = True
    rejection_reasons = []

    if income < MIN_INCOME:
        is_approved = False
```



```python
    if income < MIN_INCOME:
        is_approved = False
        rejection_reasons.append(f"Income is below the minimum requirement of ${MIN_INCOME}.")

    if credit_score < MIN_CREDIT_SCORE:
        is_approved = False
        rejection_reasons.append(f"Credit score is below the minimum requirement of {MIN_CREDIT_SCORE}.")

    if debt_to_income_ratio > MAX_DEBT_TO_INCOME_RATIO:
        is_approved = False
        rejection_reasons.append(f"Debt-to-income ratio is above the maximum allowed of {MAX_DEBT_TO_INCOME_RATIO}.")

    # 4. Provide loan approval status
    print(f"\nLoan Application Status for {name}:")
    if is_approved:
        print("Congratulations! Your loan is approved.")
    else:
        print("We are sorry, your loan is rejected.")
        print("Reasons for rejection:")
        for reason in rejection_reasons:
            print(f"- {reason}")

# Run the system
loan_approval_system()
```

Output:


```
Welcome to the simple Loan Approval System!
Enter your name: john
Enter your annual income: 50000
Enter your credit score: 650
Enter your debt-to-income ratio (as a decimal, e.g., 0.35): 0.4

Loan Application Status for john:
Congratulations! Your loan is approved.
```

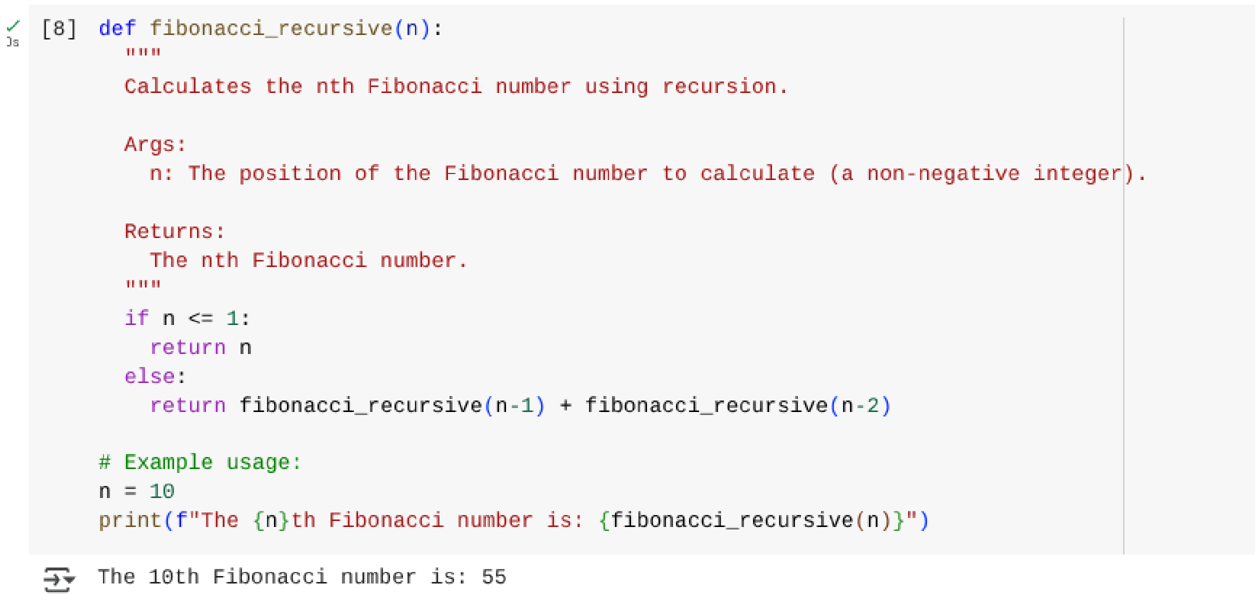
Explanation:

1. The code defines a function `loan_approval_system`.
2. Inside, it sets minimum income, credit score, and maximum debt-to-income ratio for loan approval.
3. It welcomes the user and prompts for their name, income, credit score, and debt-to-income ratio.
4. It includes error handling for non-numeric input.
5. It initializes `is_approved` to `True` and an empty list for `rejection_reasons`.
6. It checks if the income is below the minimum; if so, it sets `is_approved` to `False` and adds a reason.
7. It checks if the credit score is below the minimum; if so, it sets `is_approved` to `False` and adds a reason.
8. It checks if the debt-to-income ratio is above the maximum; if so, it sets `is_approved` to `False` and adds a reason.
9. Finally, it prints the loan application status, indicating approval or rejection with reasons if applicable.
10. The `loan_approval_system()` function is called to run the program.

Task-3:

Write a python function to calculate the nth fibonacci number using recursion.

Code and Output:



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

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

      Returns:
        The nth Fibonacci number.
      """
      if n <= 1:
        return n
      else:
        return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

    # Example usage:
    n = 10
    print(f"The {n}th Fibonacci number is: {fibonacci_recursive(n)}")
```

```
The 10th Fibonacci number is: 55
```

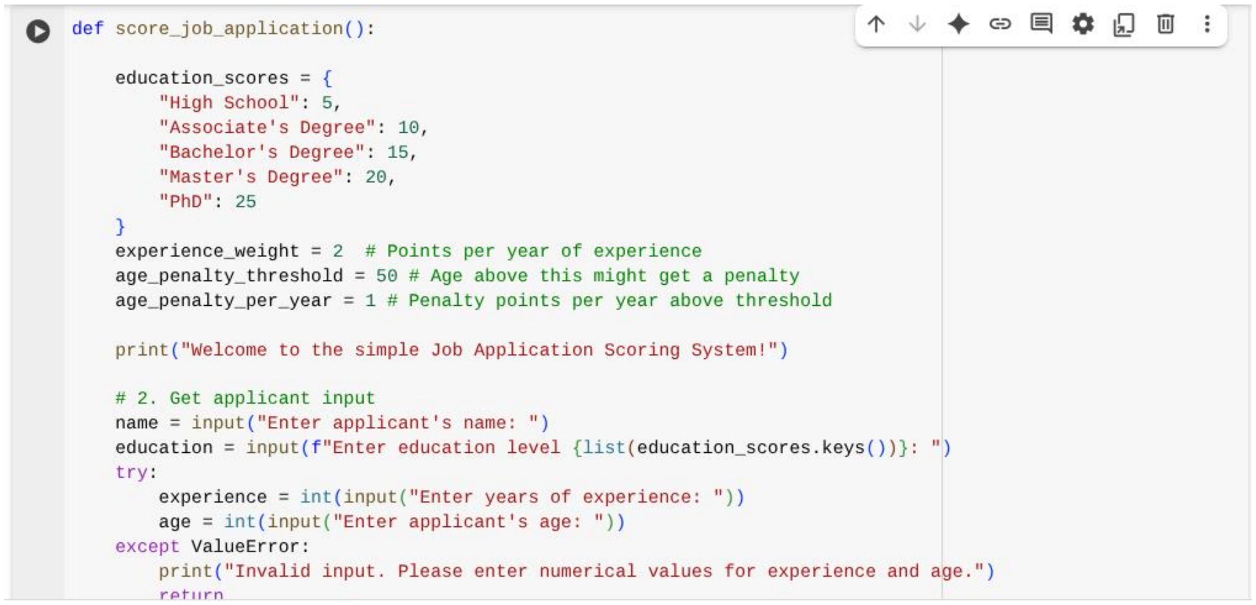
Explanation:

1. The code defines a function `fibonacci_recursive(n)` that calculates the nth Fibonacci number.
2. It uses recursion, where the function calls itself.
3. The base case for the recursion is when `n` is less than or equal to 1.
4. In the base case, the function returns `n` directly (0 for n=0, 1 for n=1).
5. For `n` greater than 1, the function enters the recursive step.
6. It calculates the Fibonacci number by summing the (n-1)th and (n-2)th Fibonacci numbers.
7. This is done by calling `fibonacci_recursive(n-1)` and `fibonacci_recursive(n-2)`.
8. The function continues to call itself with smaller values of `n` until it reaches the base case.
9. The results from the base cases are then combined upwards to calculate the final result.
10. An example demonstrates calling the function with `n = 10` and printing the result.

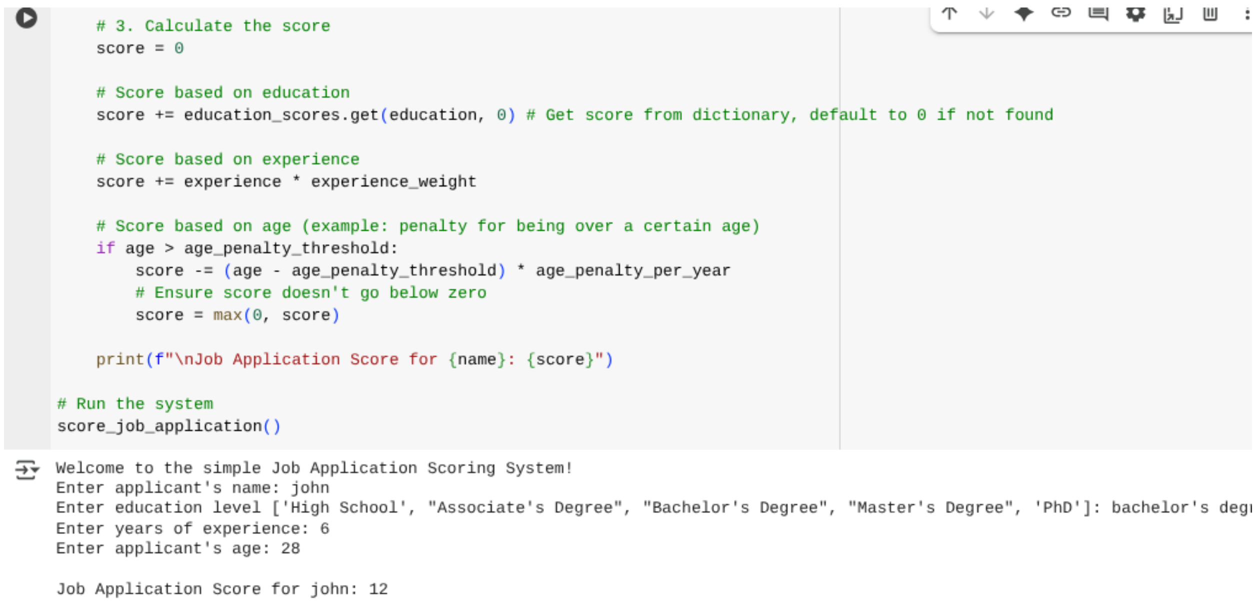
Task-4:

Write a python function to score job application based on input based on input features (e.g., education, experience, gender, age)

Code and Output:



```python
def score_job_application():

    education_scores = {
        "High School": 5,
        "Associate's Degree": 10,
        "Bachelor's Degree": 15,
        "Master's Degree": 20,
        "PhD": 25
    }
    experience_weight = 2  # Points per year of experience
    age_penalty_threshold = 50 # Age above this might get a penalty
    age_penalty_per_year = 1 # Penalty points per year above threshold

    print("Welcome to the simple Job Application Scoring System!")

    # 2. Get applicant input
    name = input("Enter applicant's name: ")
    education = input(f"Enter education level {list(education_scores.keys())}: ")
    try:
        experience = int(input("Enter years of experience: "))
        age = int(input("Enter applicant's age: "))
    except ValueError:
        print("Invalid input. Please enter numerical values for experience and age.")
        return
```



```
    # 3. Calculate the score
    score = 0

    # Score based on education
    score += education_scores.get(education, 0) # Get score from dictionary, default to 0 if not found

    # Score based on experience
    score += experience * experience_weight

    # Score based on age (example: penalty for being over a certain age)
    if age > age_penalty_threshold:
        score -= (age - age_penalty_threshold) * age_penalty_per_year
        # Ensure score doesn't go below zero
        score = max(0, score)

    print(f"\nJob Application Score for {name}: {score}")

# Run the system
score_job_application()
```

```
Welcome to the simple Job Application Scoring System!
Enter applicant's name: john
Enter education level ['High School', "Associate's Degree", "Bachelor's Degree", "Master's Degree", 'PhD']: bachelor's degi
Enter years of experience: 6
Enter applicant's age: 28

Job Application Score for john: 12
```

Explanation:

1. The code defines a function `score_job_application()` to calculate an application score.
2. It sets up scoring criteria: points for education levels and points per year of experience.
3. It also defines a penalty for age above a certain threshold.
4. The program welcomes the user and asks for the applicant's name and education level.
5. It prompts for years of experience and age, with error handling for non-numeric input.
6. A `score` variable is initialized to zero.
7. Points are added to the score based on the education level provided.
8. Points are added based on the years of experience, multiplied by a weight.
9. A penalty is applied if the applicant's age is above the defined threshold, ensuring the score doesn't go below zero.
10. Finally, the program prints the calculated job application score for the applicant's name.

Task-5:

Prompt:

def greet_user(name, gender):

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

title = "Mr."

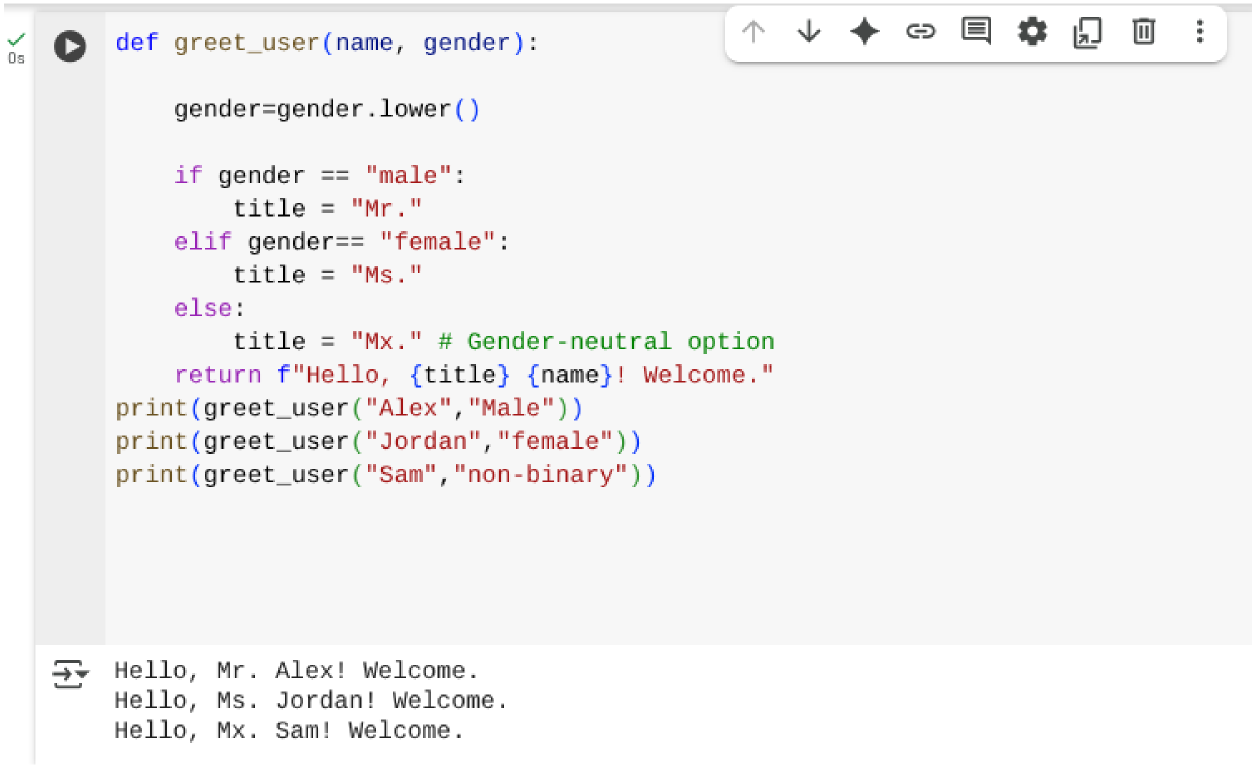
else:

title = "Mrs."

return f"Hello, {title} {name}! Welcome."

Regenerate code that includes gender-neutral also

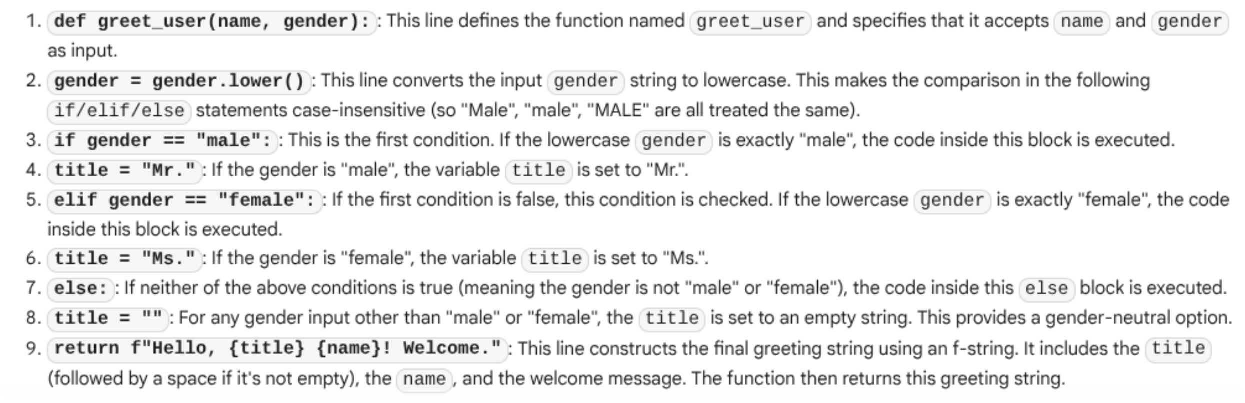
Code and Output:



```
def greet_user(name, gender):

    gender=gender.lower()

    if gender == "male":
        title = "Mr."
    elif gender== "female":
        title = "Ms."
    else:
        title = "Mx." # Gender-neutral option
    return f"Hello, {title} {name}! Welcome."
print(greet_user("Alex","Male"))
print(greet_user("Jordan","female"))
print(greet_user("Sam","non-binary"))
```

```
Hello, Mr. Alex! Welcome.
Hello, Ms. Jordan! Welcome.
Hello, Mx. Sam! Welcome.
```

Explanation:



1. **`def greet_user(name, gender):`**: This line defines the function named `greet_user` and specifies that it accepts `name` and `gender` as input.
2. **`gender = gender.lower()`**: This line converts the input `gender` string to lowercase. This makes the comparison in the following `if/elif/else` statements case-insensitive (so "Male", "male", "MALE" are all treated the same).
3. **`if gender == "male":`**: This is the first condition. If the lowercase `gender` is exactly "male", the code inside this block is executed.
4. **`title = "Mr."`**: If the gender is "male", the variable `title` is set to "Mr.".
5. **`elif gender == "female":`**: If the first condition is false, this condition is checked. If the lowercase `gender` is exactly "female", the code inside this block is executed.
6. **`title = "Ms."`**: If the gender is "female", the variable `title` is set to "Ms.".
7. **`else:`**: If neither of the above conditions is true (meaning the gender is not "male" or "female"), the code inside this `else` block is executed.
8. **`title = ""`**: For any gender input other than "male" or "female", the `title` is set to an empty string. This provides a gender-neutral option.
9. **`return f"Hello, {title} {name}! Welcome."`**: This line constructs the final greeting string using an f-string. It includes the `title` (followed by a space if it's not empty), the `name`, and the welcome message. The function then returns this greeting string.