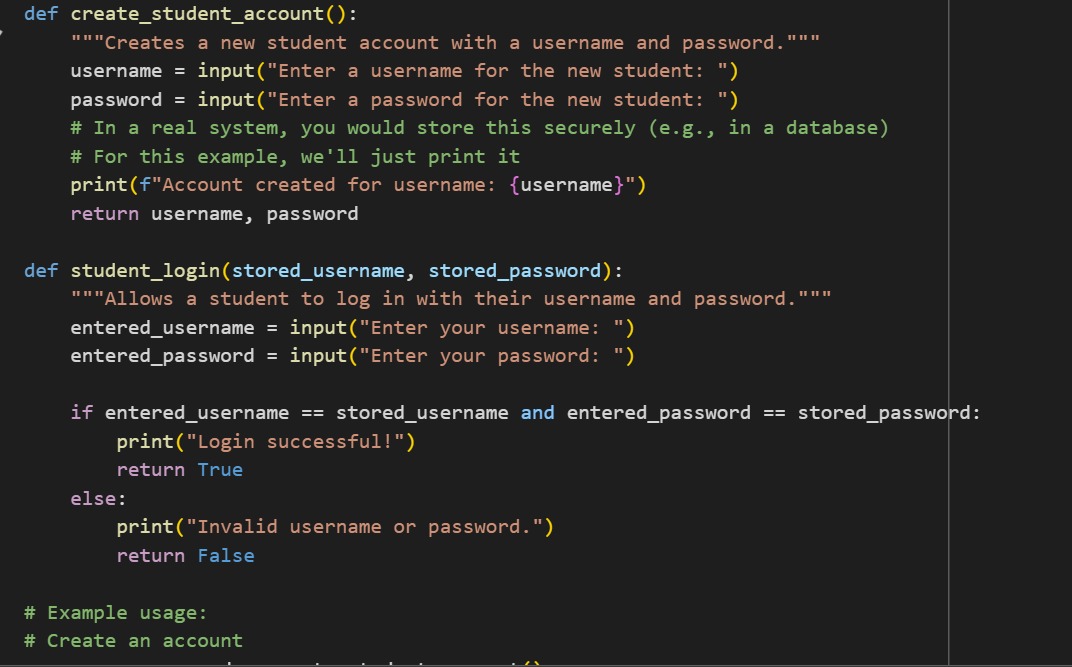
**ASSIGNMENT-5.2**

**TASK-1:**

**write a program in python to develop a student login system create username and password**

**CODE:**



A computer screen with text and images

AI-generated content may be incorrect.

**Output:**A screenshot of a computer program

AI-generated content may be incorrect.

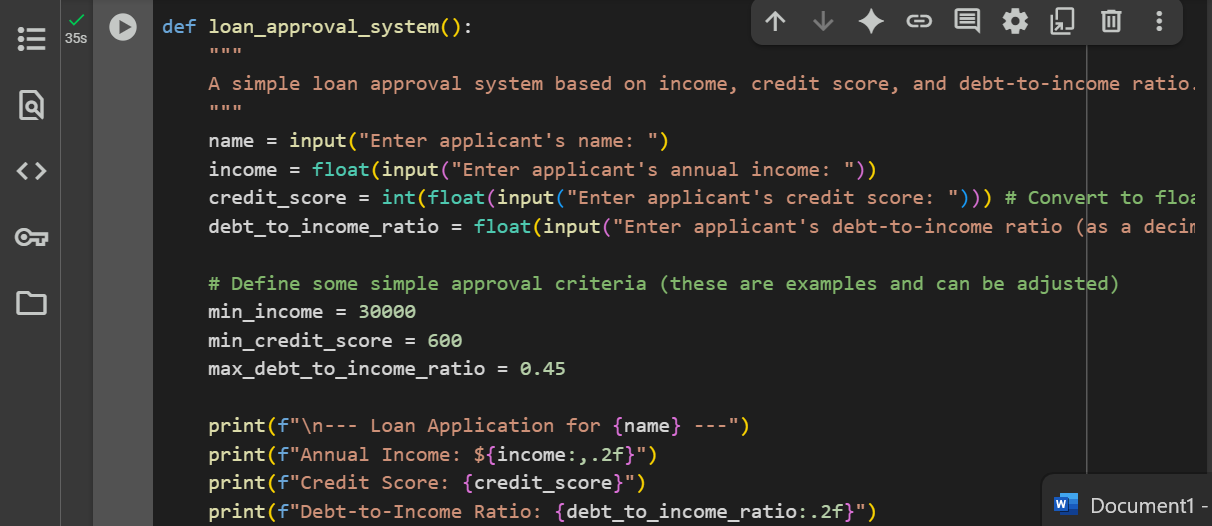
**Explanation:**

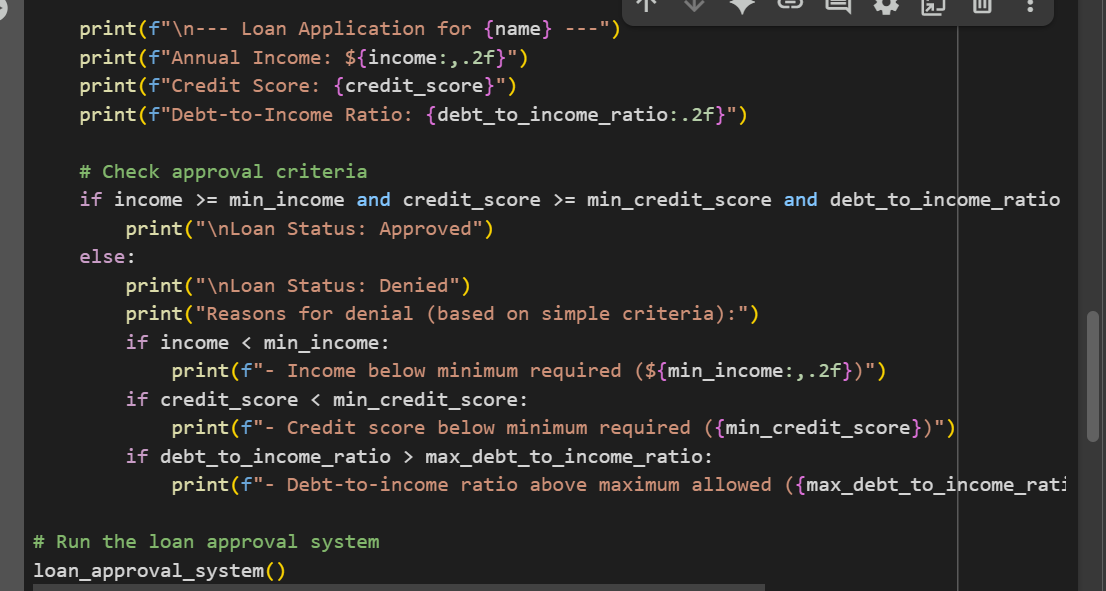
1. **, stored\_password) function:**
2. This function handles the **create\_student\_account() function:**
   * This function is responsible for creating a new student account.
   * It prompts the user to enter a username and password using the input() function.
   * In a real-world application, you would securely store this information (e.g., in a database), but in this example, it just prints a confirmation message.
   * It returns the entered username and password.
   * **student\_login(stored\_username**student login process.
   * It takes the stored username and password as arguments.
   * It prompts the user to enter their username and password for login.
   * It compares the entered credentials with the stored ones.
   * If they match, it prints "Login successful!" and returns True.
   * If they don't match, it prints "Invalid username or password." and returns False.
3. **Example usage:**
   * username, password = create\_student\_account(): This line calls the create\_student\_account() function to create a new account and stores the returned username and password in the username and password variables.
   * student\_login(username, password): This line calls the student\_login() function, passing the created username and password to attempt a login.

In essence, this code allows you to create one student account and then verify if the entered credentials match that single account.

**TASK-2:**

**write a program in python to develop a student login system create username and password**

****



**OUTPUT :**

**A screenshot of a computer

AI-generated content may be incorrect.**

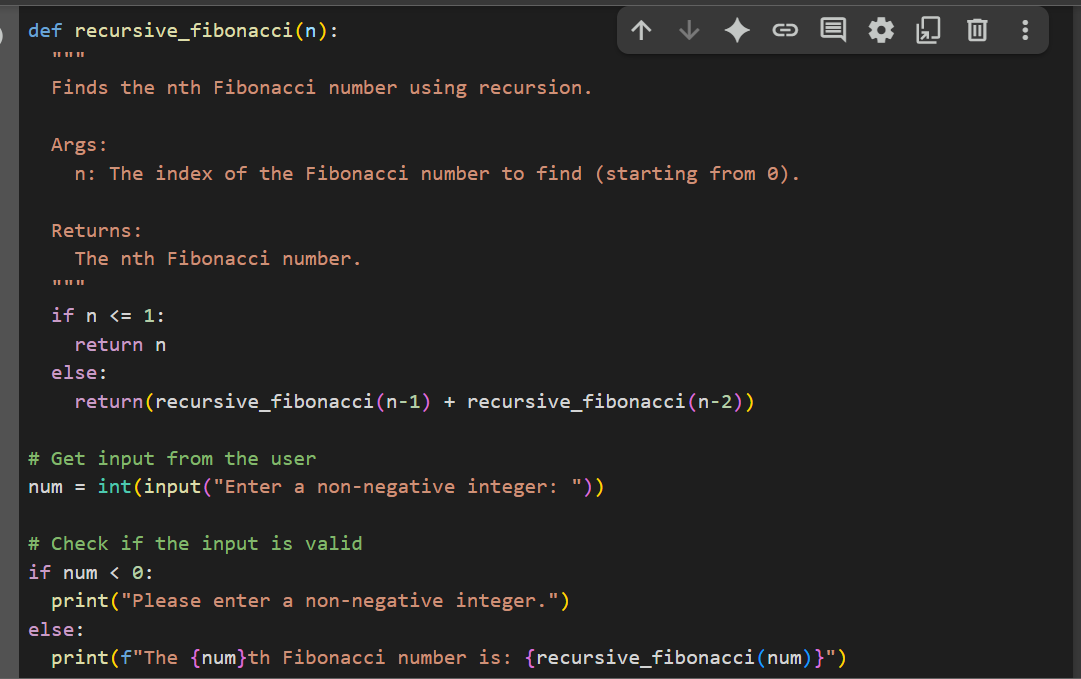
**EXPLANATION:**

1. **loan\_approval\_system() function:**
   * **This function encapsulates the entire loan approval logic.**
   * **It starts by prompting the user to enter the applicant's name, annual income, credit score, and debt-to-income ratio using the input() function.**
   * **It converts the income and debt-to-income ratio to floating-point numbers using float() to handle potential decimal values.**
   * **It converts the credit score to an integer using int(float()) to ensure it's treated as a whole number, while still allowing for decimal input initially.**
   * **Approval Criteria: It defines three variables (min\_income, min\_credit\_score, max\_debt\_to\_income\_ratio) with simple criteria for loan approval. These are example values and can be adjusted.**
   * **It then prints the applicant's information that was entered.**
   * **Approval Check: It uses an if statement to check if all three criteria are met simultaneously:**
     + **Income is greater than or equal to min\_income.**
     + **Credit score is greater than or equal to min\_credit\_score.**
     + **Debt-to-income ratio is less than or equal to max\_debt\_to\_income\_ratio.**
   * **If all criteria are met, it prints "Loan Status: Approved".**
   * **If any of the criteria are not met, it prints "Loan Status: Denied" and then provides specific reasons for the denial by checking which of the individual criteria were not met.**
2. **loan\_approval\_system():**
   * **This line calls the loan\_approval\_system() function to start the application process when the code is run.**

**In summary, this program takes in basic financial information from an applicant and uses a set of predefined rules to determine if the loan should be approved or denied, providing reasons for denial if applicable.**

**TASK-3:**

WRITE A PYTHON PROGRAM TO FIND THE NTH FIBANOCCI NUMBER USING RECURSSION



OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

EXPLANATION:

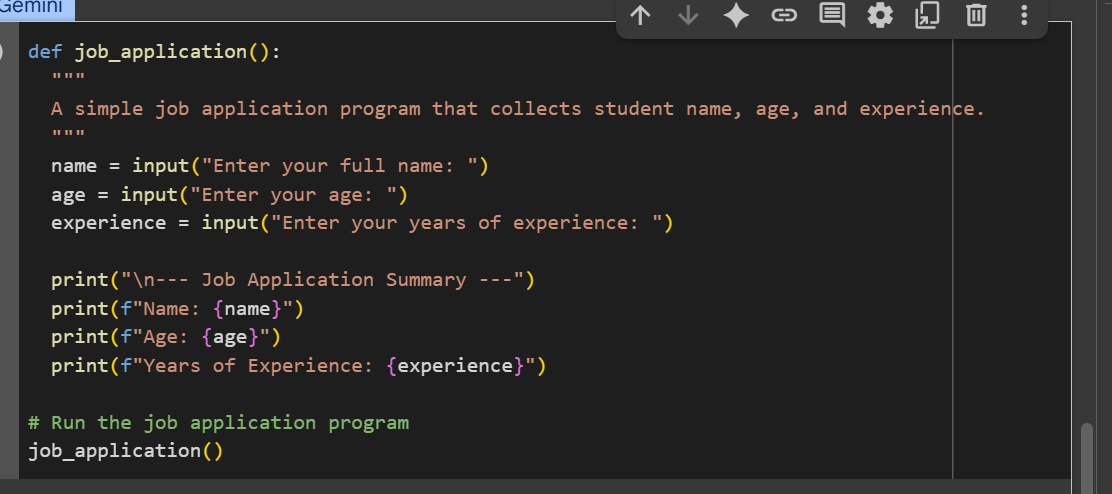
1. **recursive\_fibonacci(n) function:**
   * This function is defined to compute the Fibonacci number at a given index n.
   * **Base Case:** The if n <= 1: condition is the base case of the recursion. The Fibonacci sequence starts with 0 and 1. So, if n is 0 or 1, the function simply returns n. This is crucial to stop the recursion.
   * **Recursive Step:** The else block handles the recursive part. For any n greater than 1, the function calls itself twice:
     + recursive\_fibonacci(n-1): This finds the Fibonacci number at the previous index.
     + recursive\_fibonacci(n-2): This finds the Fibonacci number at the index before the previous one.
   * The function then returns the sum of these two recursive calls. This is the definition of the Fibonacci sequence: each number is the sum of the two preceding ones.
2. **Getting user input:**
   * num = int(input("Enter a non-negative integer: ")): This line prompts the user to enter a non-negative integer and stores it in the num variable after converting it to an integer.
3. **Input validation:**
   * if num < 0:: This checks if the entered number is negative.
   * If it is negative, it prints an error message asking the user to enter a non-negative integer.
   * else:: If the number is non-negative, it proceeds to calculate and print the Fibonacci number.
4. **Calculating and printing the result:**
   * print(f"The {num}th Fibonacci number is: {recursive\_fibonacci(num)}"): This line calls the recursive\_fibonacci() function with the user's input num and prints the result in a formatted string.

In essence, the recursive\_fibonacci function breaks down the problem of finding the Nth Fibonacci number into smaller subproblems (finding the (N-1)th and (N-2)th Fibonacci numbers) until it reaches the base cases (0 and 1), and then it builds up the solution from there.

**TASK-4:**

Write a python program to develop a job application with student name to generate age and experience by user input

**Code:**



**Output:**

**A screenshot of a computer

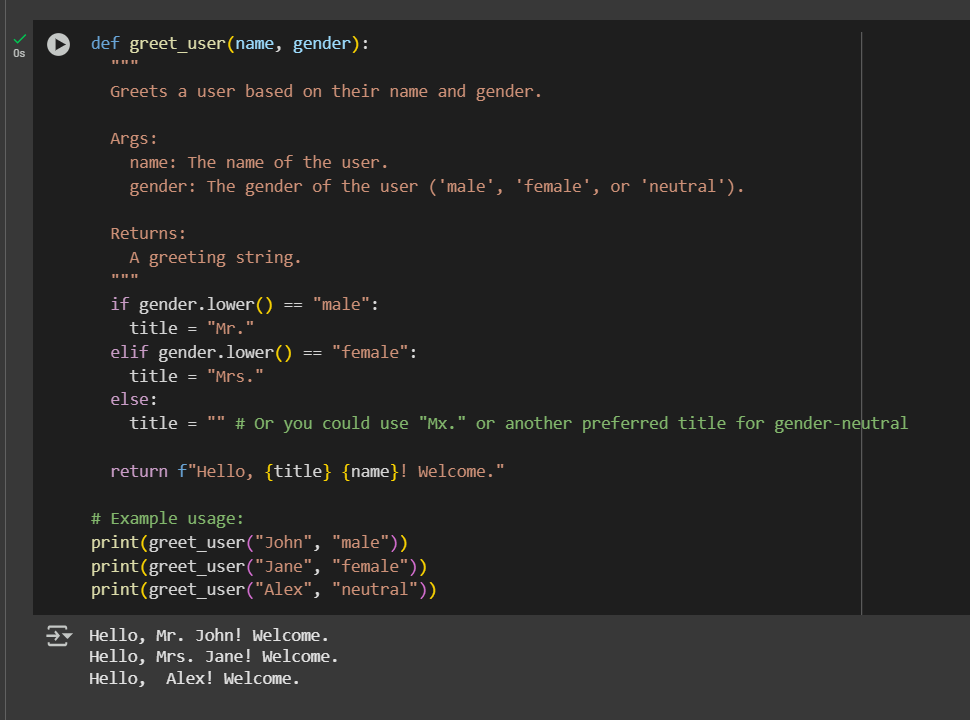
AI-generated content may be incorrect.**

**EXPLANATION:**

1. **job\_application() function:**
   * **This function contains the logic for the job application process.**
   * **It uses the input() function three times to prompt the user to enter their full name, age, and years of experience. The values entered by the user are stored in the name, age, and experience variables, respectively.**
   * **After collecting the input, it prints a header "--- Job Application Summary ---".**
   * **Finally, it prints the collected information (Name, Age, and Years of Experience) using f-strings for formatted output.**
2. **job\_application():**
   * **This line calls the job\_application() function to start the program when the cell is executed.**

**TASK-5:**

**Code:**

****

**Explanation:**

**Certainly! This Python code defines a function called greet\_user that takes a user's name and gender as input and returns a personalized greeting.**

**Here's a breakdown:**

**def greet\_user(name, gender):: This line defines the function greet\_user and specifies that it accepts two arguments: name and gender.**

**""" ... """: This is a docstring, which explains what the function does, its arguments, and what it returns.**

**if gender.lower() == "male":: This checks if the gender input, converted to lowercase, is equal to "male". If it is, it sets the title variable to "Mr.".**

**elif gender.lower() == "female":: If the gender is not "male", this checks if the lowercase gender is "female". If it is, it sets the title to "Mrs.".**

**else:: If the gender is neither "male" nor "female", this block is executed. It sets the title to an empty string "", allowing for a gender-neutral greeting. You could also change this to "Mx." or another preferred gender-neutral title.**

**return f"Hello, {title} {name}! Welcome.": This line constructs the greeting string using an f-string. It includes "Hello,", the determined title (if any), the user's name, and "! Welcome.". The return statement sends this string back as the output of the function.**

**print(greet\_user("John", "male")): This is an example of how to call the function with the name "John" and gender "male", and then prints the returned greeting.**

**print(greet\_user("Jane", "female")): This is another example, calling the function with "Jane" and "female".**

**print(greet\_user("Alex", "neutral")): This example calls the function with "Alex" and "neutral".**