LAB ASSIGNMENT : 5

TASK-01

PROMPT:

Write a python program to generate a code for login system and password .

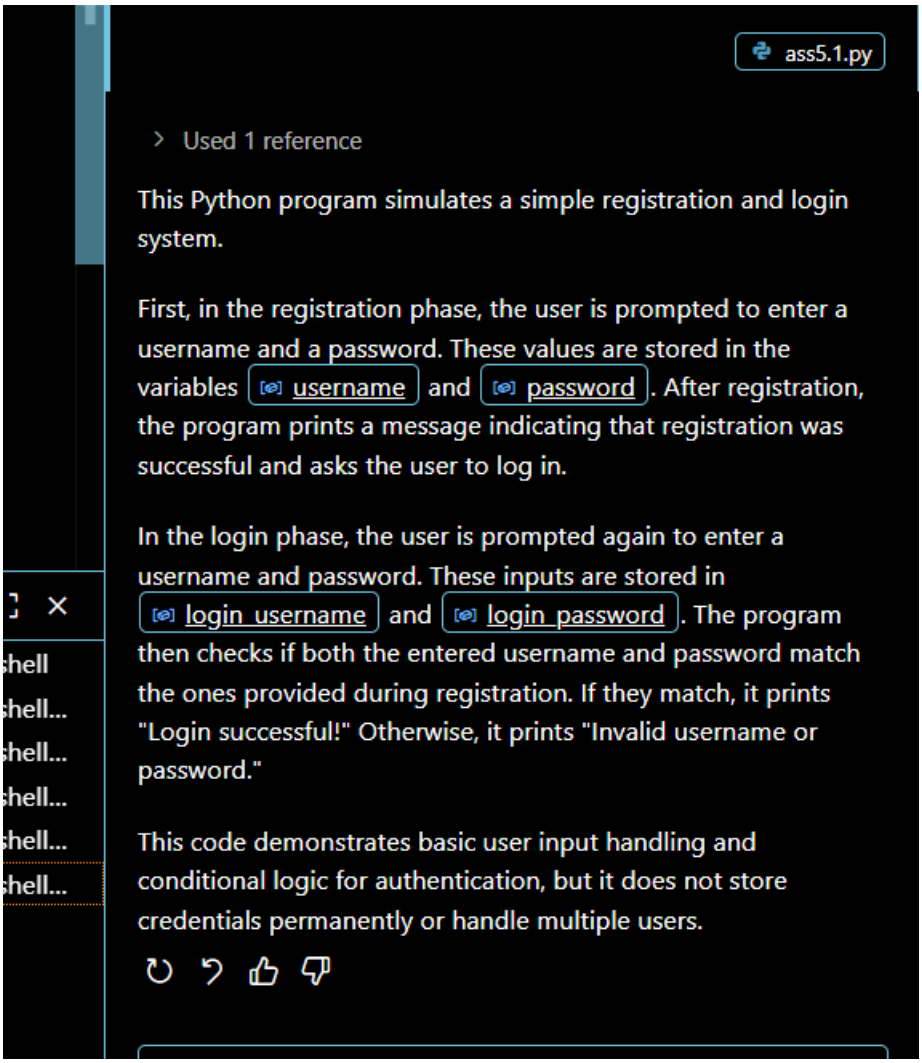
CODE:

```python
#write a program in python to generate a code for log in system and password
# Registration phase
username = input("Register a username: ")
password = input("Register a password: ")
# Store the registered username and password for later verification
print("Registration successful! Please log in.\n")

# Login phase
login_username = input("Enter username: ")
login_password = input("Enter password: ")

if login_username == username and login_password == password:
    print("Login successful!")
else:
    print("Invalid username or password.")
```

OUTPUT:

```
Enter username: sahithi
Enter password: sahithi@4070
Invalid username or password.
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code> & C:\Users\sahit\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/ass5.1.py"
Register a username: sahithi
Register a password: sahithi@4070
Registration successful! Please log in.

Enter username: sahithi
Enter password: sahithi@4070
Login successful!
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

EXPLANATION:



TASK-02

PROMPT:

Write a python program to generate a code for the loan approval

CODE:

```python
#write a python program to generate a code for loan approval
#write a python program to generate a code for loan approval

def loan_approval(income, credit_score, loan_amount):
    # Simple criteria for loan approval
    if income >= 30000 and credit_score >= 650 and loan_amount <= income * 10:
        return "Loan Approved"
    else:
        return "Loan Denied"


# Take user input
income = float(input("Enter your annual income: "))
credit_score = int(input("Enter your credit score: "))
loan_amount = float(input("Enter desired loan amount: "))

result = loan_approval(income, credit_score, loan_amount)
print(result)
```

OUTPUT:

```
Enter desired loan amount: 500000
Loan Denied
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code> & C:\Users\sahit\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/ass5.2.py"
Enter your annual income: 40000
Enter your credit score: 700
Enter desired loan amount: 200000
Loan Approved
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

EXPLANATION:

This Python program is designed to determine whether a loan should be approved based on three user-provided inputs: annual income, credit score, and desired loan amount. It begins by defining a function called `loan_approval` that takes these three values as arguments. Inside the function, a simple conditional check is performed: if the income is at least 30,000, the credit score is 650 or higher, and the requested loan amount does not exceed ten times the income, then the function returns "Loan Approved". Otherwise, it returns "Loan Denied". After defining the function, the program prompts the user to enter their annual income, credit score, and desired loan amount using the `input()` function. These inputs are converted to the appropriate data types—`float` for income and loan amount, and `int` for credit score—to ensure accurate comparison. The program then calls the `loan_approval` function with the user's inputs and prints the result, which tells the user whether their loan has been approved or denied based on the predefined criteria.

TASK- 03

PROMPT:

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

CODE:

```python
# Define a recursive function to calculate the nth Fibonacci number
def fibonacci(n):
    # Handle invalid input
    if n <= 0:
        return "Invalid input. Please enter a positive integer."
    # Base case: the 1st Fibonacci number is 0
    elif n == 1:
        return 0
    # Base case: the 2nd Fibonacci number is 1
    elif n == 2:
        return 1
    # Recursive case: sum of the two preceding Fibonacci numbers
    else:
        return fibonacci(n - 1) + fibonacci(n - 2)

# Prompt the user to enter the position of the Fibonacci number
n = int(input("Enter the position (n) of the Fibonacci number you want: "))

# Call the recursive function and store the result
result = fibonacci(n)

# Display the result to the user
print(f"The {n}th Fibonacci number is: {result}")

```

OUTPUT:

```
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code> ^C
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code> & C:\Users\sahit\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/sahit/New folder (2)/2403a54070/ass5.3.py"
Enter the position (n) of the Fibonacci number you want: 8
The 8th Fibonacci number is: 13
PS C:\Users\sahit\AppData\Local\Programs\Microsoft VS Code>
```

EXPLANATION:

This Python program calculates the nth Fibonacci number using recursion. It begins by defining a function called `fibonacci` that takes an integer `n` as input. The function first checks if `n` is less than or equal to zero, in which case it returns an error message indicating invalid input. If `n` equals 1, it returns 0, and if `n` equals 2, it returns 1—these are the base cases of the Fibonacci sequence. For any value of `n` greater than 2, the function calls itself twice: once with `n - 1` and once with `n - 2`, and returns the sum of those two calls. This recursive structure mirrors the mathematical definition of the Fibonacci sequence, where each term is the sum of the two preceding ones. After defining the function, the program prompts the user to enter the desired position in the Fibonacci sequence. It converts the input to an integer and passes it to the `fibonacci` function. The result is stored in the variable `result`, which is then printed to the screen, showing the user the nth Fibonacci number.

TASK-04

PROMPT:

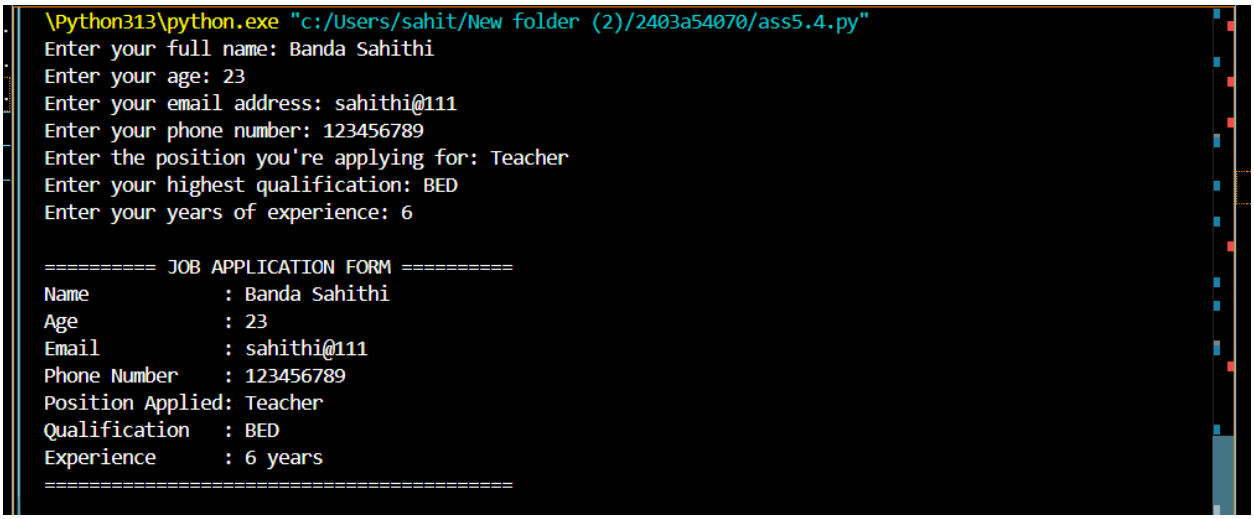
Write a python program to generate the job application with some basic information

CODE:

```python
# Function to generate a job application form
def generate_application_form(name, age, email, phone, position, qualification, experience):
    form = f"""
========== JOB APPLICATION FORM ==========
Name           : {name}
Age            : {age}
Email          : {email}
Phone Number   : {phone}
Position Applied: {position}
Qualification  : {qualification}
Experience     : {experience} years
==========================================
"""
    return form

# Collect user input
name = input("Enter your full name: ")
age = input("Enter your age: ")
email = input("Enter your email address: ")
phone = input("Enter your phone number: ")
position = input("Enter the position you're applying for: ")
qualification = input("Enter your highest qualification: ")
experience = input("Enter your years of experience: ")

# Generate and display the application form
application_form = generate_application_form(name, age, email, phone, position, qualification, e
print(application_form)
```

```python
# Prompt the user to enter their phone number
phone = input("Enter your phone number: ")

# Prompt the user to enter the job position they are applying for
position = input("Enter the position you're applying for: ")

# Prompt the user to enter their highest qualification
qualification = input("Enter your highest qualification: ")

# Prompt the user to enter their years of experience
experience = input("Enter your years of experience: ")

# Call the function to generate the application form using the collected data
application_form = generate_application_form(name, age, email, phone, position, qualification, e

# Print the generated application form to the console
print(application_form)
```

OUTPUT:



EXPLANATION:

This Python program is designed to generate a structured job application form using user input. It begins by defining a function called `generate_application_form` that takes several parameters: name, age, email, phone number, position applied for, qualification, and years of experience. Inside the function, a formatted multi-line string is created using f-string syntax, which neatly organizes the provided information into a readable form that resembles a printed application. After defining the function, the program prompts the user to enter each piece of required information using the `input()` function. These inputs are stored in variables and then passed to the `generate_application_form` function. The returned formatted string is stored in the variable `application_form`, which is then printed to the console. The result is a clean, console-based job application that displays all the entered details in a structured layout. This approach makes it easy to collect and present user data without needing a graphical interface or external file handling.