Lab Assignment-3.3

Name : Saini.Deepthi

Hall ticket No. : 2403A52014

Batch No. : 24BTCAIAIB02

Course Title : Ai Assisted Coding

TASK -1

Description:

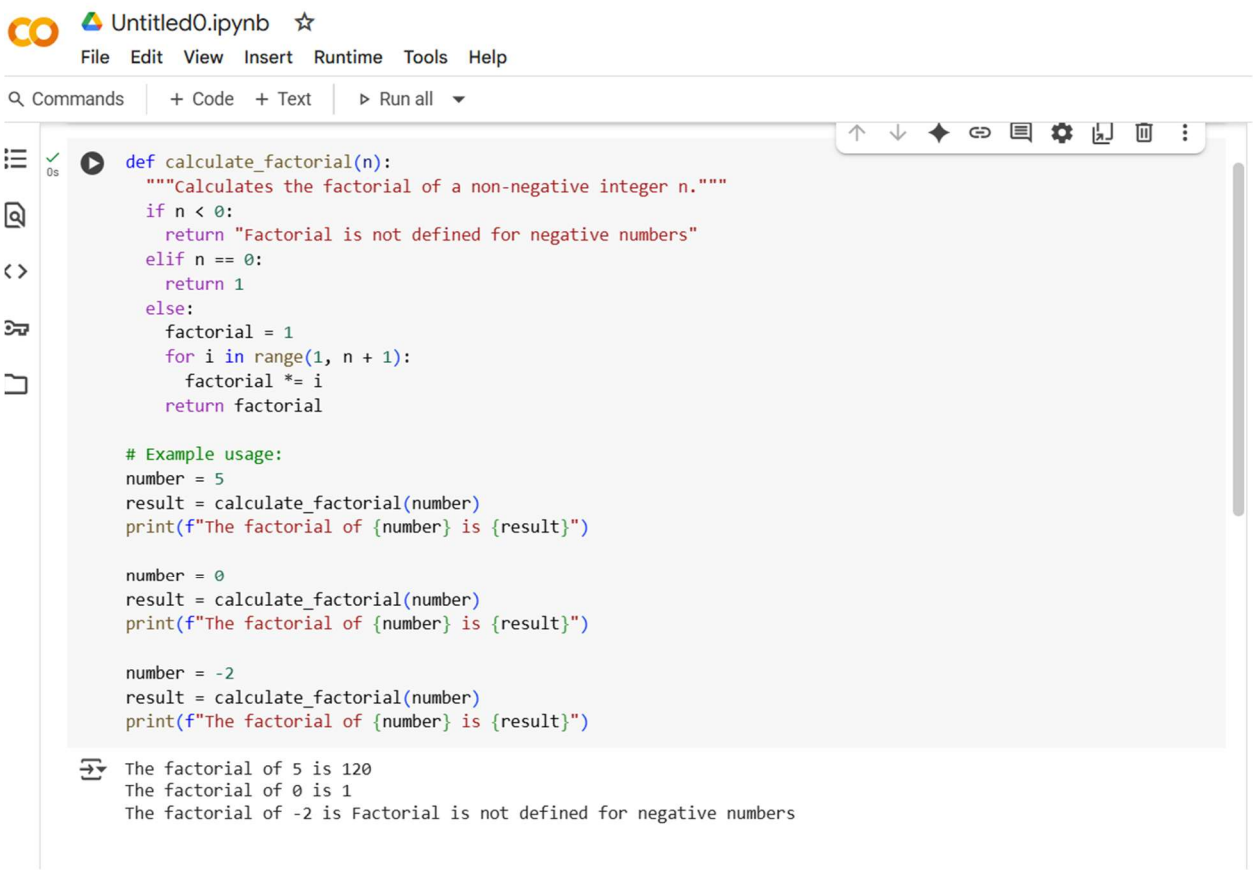
>>Try 3 different prompts to generate a factorial function.

Expected Output:

>>Comparison of AI-generated code styles.

Prompt1:

>>Generate Python code for a factorial function using a for loop.



Prompt2:

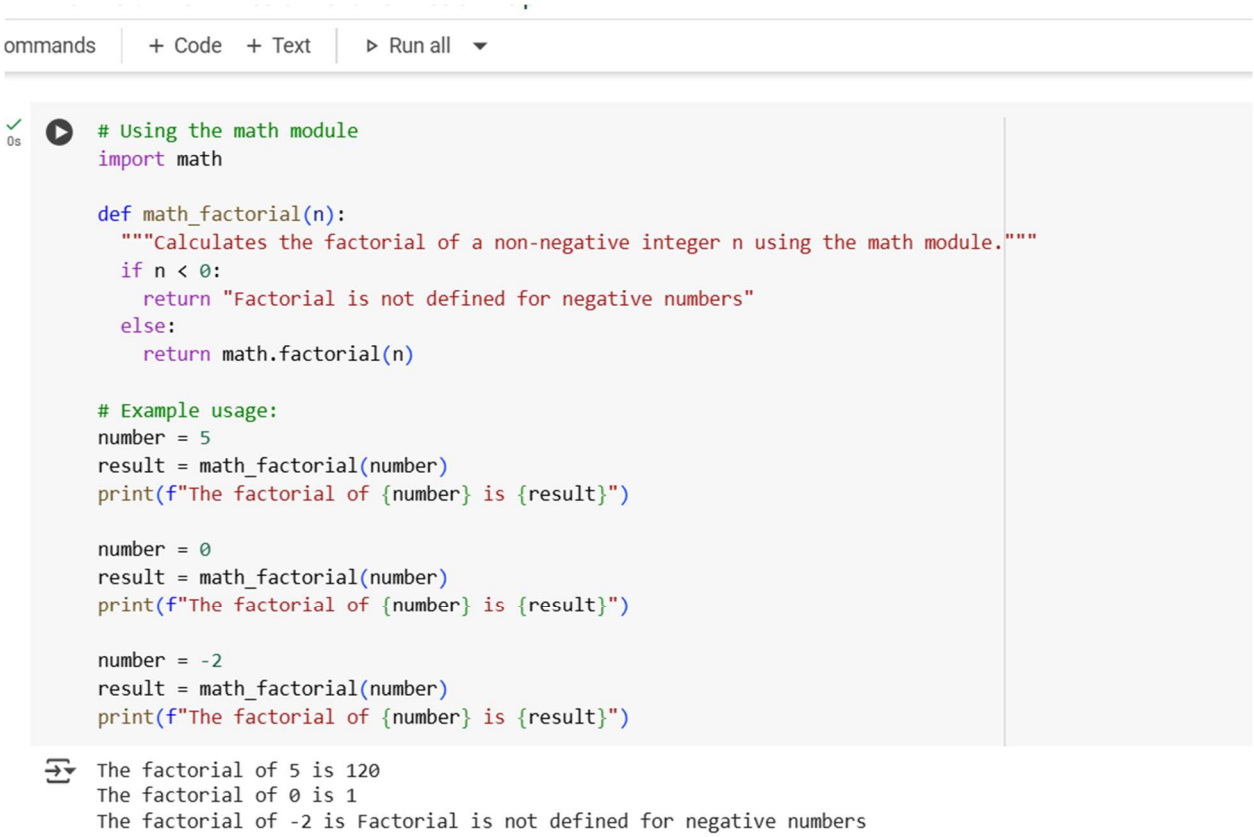
>> Write Python code that calculates factorial using the math module.

```
[2] # Using recursion
    def recursive_factorial(n):
      """Calculates the factorial of a non-negative integer n using recursion."""
      if n < 0:
        return "Factorial is not defined for negative numbers"
      elif n == 0:
        return 1
      else:
        return n * recursive_factorial(n - 1)

    # Example usage:
    number = 5
    result = recursive_factorial(number)
    print(f"The factorial of {number} is {result}")

    number = 0
    result = recursive_factorial(number)
    print(f"The factorial of {number} is {result}")

    number = -2
    result = recursive_factorial(number)
    print(f"The factorial of {number} is {result}")
```

```
The factorial of 5 is 120
The factorial of 0 is 1
The factorial of -2 is Factorial is not defined for negative numbers
```

Prompt3:

>> Write Python code that calculates factorial using the math module.



Observation:

>>All three methods (using a for loop, recursion, and the math module) correctly calculate the factorial for non-negative integers and handle negative inputs by returning an appropriate message.

Explanation:

>>Using a for loop: This is like counting on your fingers. You start with 1 and multiply it by each number up to the given number.

Using recursion: This is like a set of Russian nesting dolls. Each doll is the same, but smaller. Here, the function calls itself with a smaller number until it reaches 0, and then it builds the answer back up.

Using the math module: This is like using a calculator. Python has a built-in tool (math.factorial) that already knows how to calculate factorials for you. You just give it the number, and it gives you the answer.

TASK -2

Description:

>>Provide a clear example input-output prompt to generate a sorting function

Expected output:

>> Functional sorting code from AI.

Prompt:

>> Generate a sorting function and provide a clear example input-output prompt.

```
[5]  # Example input:
     my_list = [64, 34, 25, 12, 22, 11, 90]
     print(f"Original list: {my_list}")

     # Call the sorting function
     sorted_list = bubble_sort(my_list)

     # Example output:
     print(f"Sorted list: {sorted_list}")
```

```
Original list: [64, 34, 25, 12, 22, 11, 90]
Sorted list: [11, 12, 22, 25, 34, 64, 90]
```

Observation:

>> This code takes a list of numbers that are in a jumbled order and rearranges them so they are in order from smallest to largest. The output shows the list before and after the sorting happens.

Explanation :

>> Think of Bubble Sort like organizing bubbles in a glass of soda. The biggest bubbles naturally rise to the top. In Bubble Sort, we go through the list of numbers again and again, comparing two numbers next to each other. If the number on the left is bigger than the number on the right, we swap them. We keep doing this until no more swaps are needed, meaning all the numbers are in the correct order, like the bubbles settling.

TASK -3

Description:

>> Start with the vague prompt “Generate python code to calculate power bill” and improve it step-by-step

Expected output:

>>Enhanced AI output with clearer prompts.

Prompt:

>> Generate python code to calculate power bill” and improve it step-by-step.

```
def calculate_power_bill(units_consumed, rate_per_unit):
  if units_consumed < 0 or rate_per_unit < 0:
    return "Units consumed and rate per unit cannot be negative."
  else:
    total_bill = units_consumed * rate_per_unit
    return total_bill
```

```
# Example Usage:
units = 250  # units in kWh
rate = 0.15  # rate in currency per kWh

bill_amount = calculate_power_bill(units, rate)

if isinstance(bill_amount, str):
  print(bill_amount)
else:
  print(f"For {units} units consumed at a rate of {rate} per unit, the total power bill is: {bill_amount}")

# Example with negative input
units_negative = -10
rate_positive = 0.10
bill_amount_negative = calculate_power_bill(units_negative, rate_positive)
print(bill_amount_negative)
```

```
For 250 units consumed at a rate of 0.15 per unit, the total power bill is: 37.5
Units consumed and rate per unit cannot be negative.
```

Observation:

>>The provided Python code effectively calculates a power bill by multiplying the units of electricity consumed by a fixed rate per unit. As demonstrated by the examples, consuming 250 units at a rate of 0.15 results in a bill of 37.5. The code also includes basic error handling to prevent calculations with negative units, returning a specific message in such cases, as seen with the input of -10 units.

Explanation :

>> This code calculates your electricity bill by multiplying how much power you used by the cost per unit. It also checks for negative usage.

TASK -4

Description:

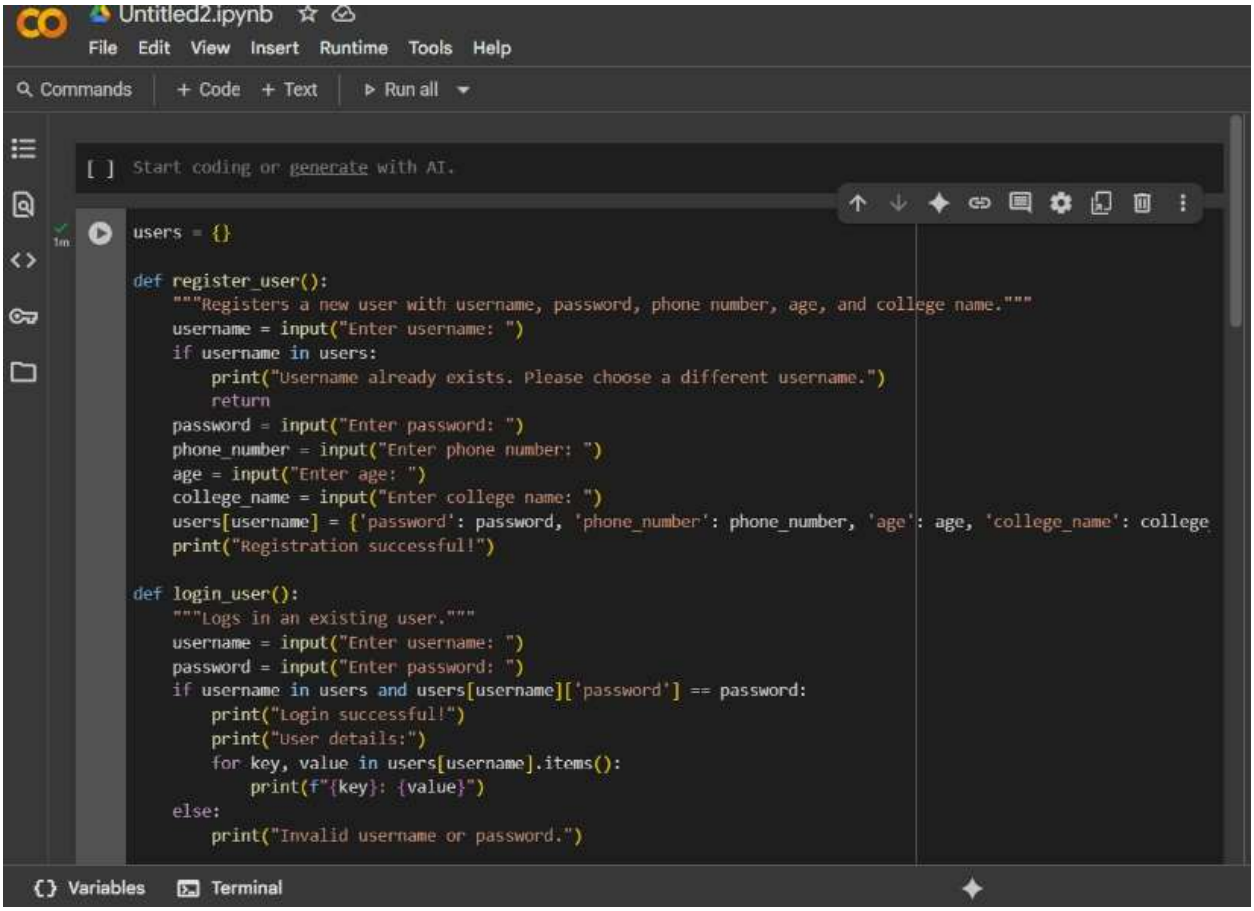
>> Write structured comments to help AI generate two linked functions (e.g., login_user() and register_user()).

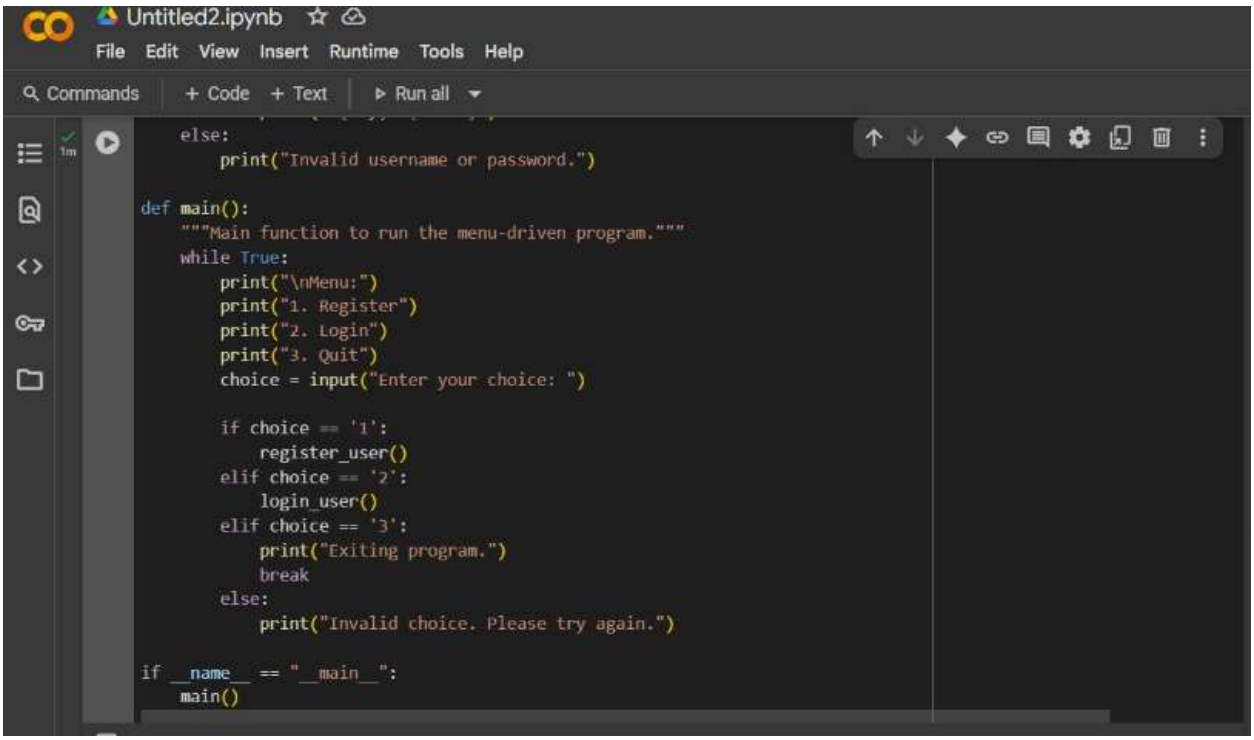
Expected output:

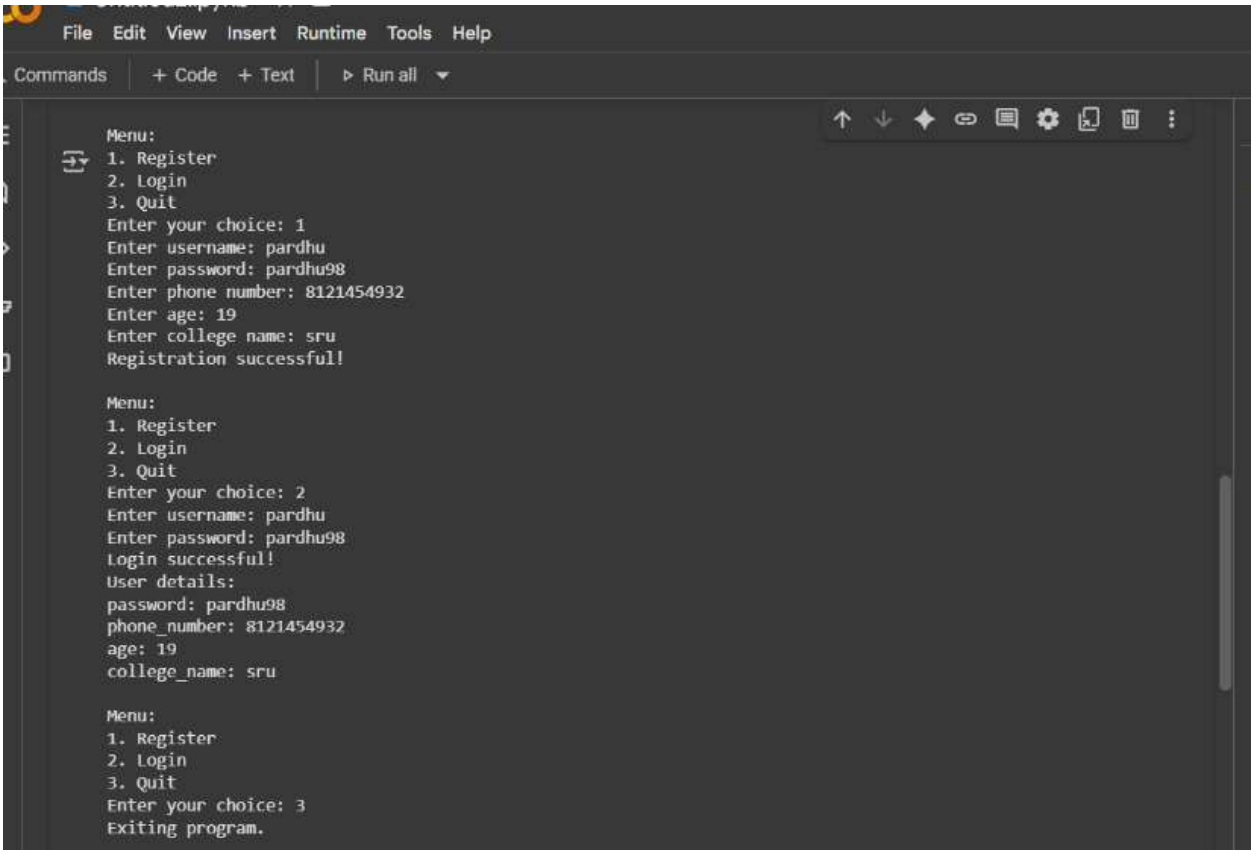
>>Consistent functions with shared logic

Prompt:

>>Generate a code for structured comments to help AI generate two linked functions (e.g., login_user() and register_user()).


Untitled2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[ ] Start coding or generate with AI.
users = {}
def register_user():
"""Registers a new user with username, password, phone number, age, and college name."""
username = input("Enter username: ")
if username in users:
print("Username already exists. Please choose a different username.")
return
password = input("Enter password: ")
phone_number = input("Enter phone number: ")
age = input("Enter age: ")
college_name = input("Enter college name: ")
users[username] = {'password': password, 'phone_number': phone_number, 'age': age, 'college_name': college
print("Registration successful!")
def login_user():
"""Logs in an existing user."""
username = input("Enter username: ")
password = input("Enter password: ")
if username in users and users[username]['password'] == password:
print("Login successful!")
print("User details:")
for key, value in users[username].items():
print(f"{key}: {value}")
else:
print("Invalid username or password.")
Variables Terminal


Untitled2.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
else:
print("Invalid username or password.")
def main():
"""Main function to run the menu-driven program."""
while True:
print("\nMenu:")
print("1. Register")
print("2. Login")
print("3. Quit")
choice = input("Enter your choice: ")
if choice == '1':
register_user()
elif choice == '2':
login_user()
elif choice == '3':
print("Exiting program.")
break
else:
print("Invalid choice. Please try again.")
if __name__ == "__main__":
main()



Observation:

>> register_user() creates a new account by checking if the username is unique and storing the password securely. login_user() verifies the entered credentials against stored data to allow access. They're linked because login depends on registration. Both must handle data safely and follow a logical flow: first register, then login.

Explanation:

>> The register_user() function helps a new user create an account. It checks if the username is already taken, and if not, it saves the username and password safely. The login_user() function is used when someone wants to sign in. It checks if the username exists and if the password matches the one saved earlier. These two functions work together—first you register, then you can log in. They're like the front door and the key: registration gives you the key, and login uses it to open the door.

TASK -5

Description:

>>Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions

Expected output:

>>Code quality difference analysis for various prompts.

Prompt:

>> Improve a Python function that converts temperatures between Celsius and Fahrenheit. Focus on making the code clean, readable, and user-friendly

Gemini

```
def celsius_to_fahrenheit(celsius):
  """Converts Celsius to Fahrenheit."""
  fahrenheit = (celsius * 9/5) + 32
  return fahrenheit

# Example usage:
celsius_temp = 25
fahrenheit_temp = celsius_to_fahrenheit(celsius_temp)
print(f"{celsius_temp}°C is equal to {fahrenheit_temp}°F")
```

```
25°C is equal to 77.0°F
```

Observation:

>> A temperature conversion function should be easy to read and use. It needs to check if the input is in Celsius or Fahrenheit, use the right formula to convert, and give a clear result. It should also handle wrong inputs nicely and explain what went wrong. Adding comments and a small example helps others understand how it works. Clean code with clear names makes everything easier to follow.

Explanation:

>>A temperature conversion function changes values between Celsius and Fahrenheit. It checks which scale is given, uses the correct formula, and returns the result rounded. It should also handle wrong inputs clearly and be easy to read and use.