

LAB ASSIGNMENT-3.3

AI Assisted coding

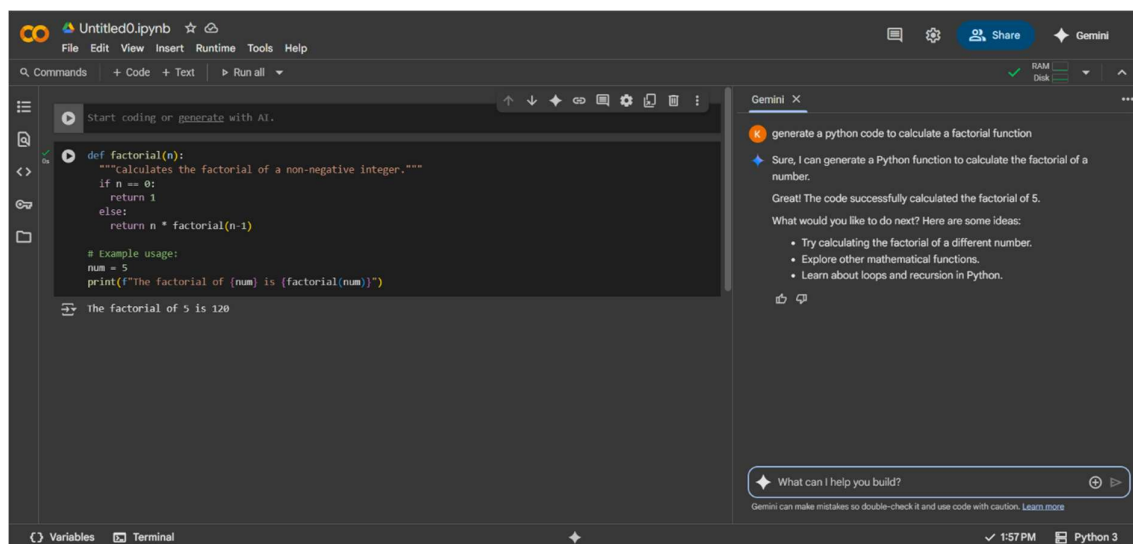
Name: V.Vamshi

Htno: 2403A52016

Task-1:

Description: Try 3 different prompts to generate a factorial function.

Prompt-1: Generate a python code to calculate a factorial function



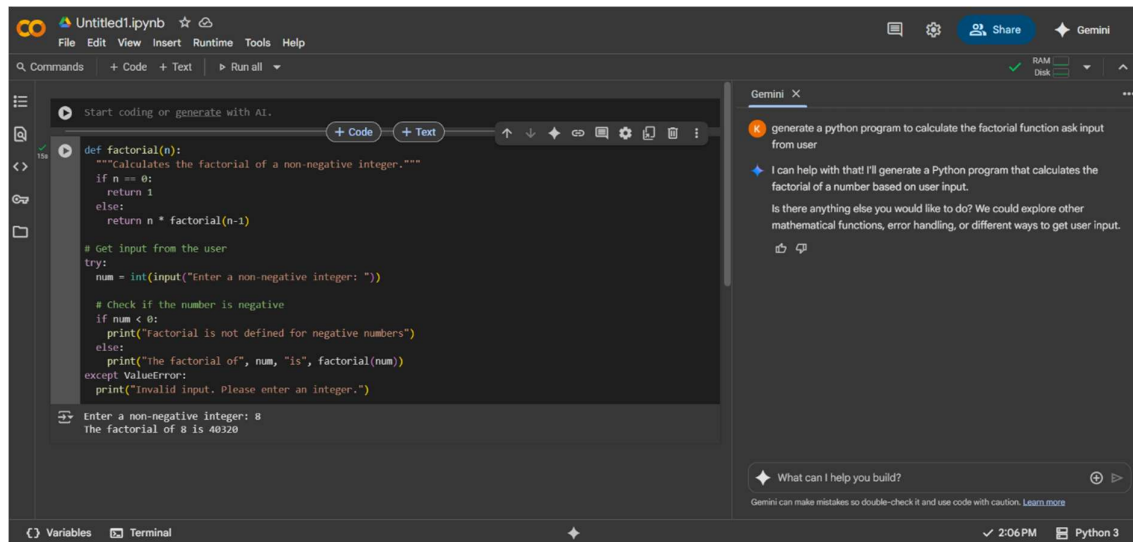
The screenshot shows a Jupyter Notebook environment. The main code cell contains a recursive Python function to calculate the factorial of a non-negative integer. The function is defined as follows:

```
def factorial(n):  
    """calculates the factorial of a non-negative integer."""  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n-1)  
  
# Example usage:  
num = 5  
print(f"The factorial of {num} is {factorial(num)}")
```

Below the code, the output shows: "The factorial of 5 is 120".

On the right side, there is a Gemini chat window. The chat history shows a prompt: "generate a python code to calculate a factorial function". The Gemini response is: "Sure, I can generate a Python function to calculate the factorial of a number. Great! The code successfully calculated the factorial of 5. What would you like to do next? Here are some ideas: • Try calculating the factorial of a different number. • Explore other mathematical functions. • Learn about loops and recursion in Python." At the bottom of the chat window, there is a prompt input field with the text: "What can I help you build?".

Prompt-2: Write a python script that aska the user for a number and prints its factorial using loop



Prompt-3: Create a python function called `factorial(n)` that returns the factorial of `n`.

```
[3] def factorial(n):  
    """Calculates the factorial of a non-negative integer."""  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n-1)
```

```
Enter a non-negative integer: 8  
The factorial of 8 is 40320
```

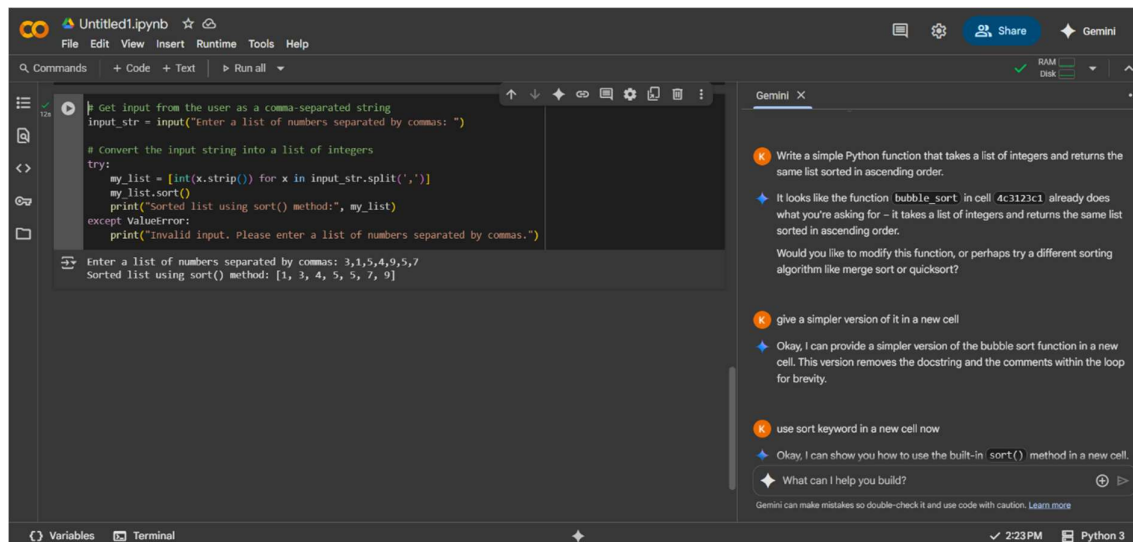
Observation: In this task I have used 3 different prompts to generate factorial function. when I use different prompts the AI also gives me different code based on the prompt.

Task-2:

Description: Provide a clear example input-output prompt to generate a sorting function.

Prompt: Write a simple python function that takes a list of integers and returns the same list sorted in ascending order.

Code:



The screenshot shows a Jupyter Notebook titled 'Untitled1.ipynb' with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar. The code cell contains the following Python code:

```
# Get input from the user as a comma-separated string
input_str = input("Enter a list of numbers separated by commas: ")

# Convert the input string into a list of integers
try:
    my_list = [int(x.strip()) for x in input_str.split(',')]
    my_list.sort()
    print("Sorted list using sort() method:", my_list)
except ValueError:
    print("Invalid input. Please enter a list of numbers separated by commas.")
```

The input field shows 'Enter a list of numbers separated by commas: 3,1,5,4,9,5,7' and the output field shows 'Sorted list using sort() method: [1, 3, 4, 5, 5, 7, 9]'. On the right, the Gemini chat window is open, showing a conversation:

- K** Write a simple Python function that takes a list of integers and returns the same list sorted in ascending order.
- I** It looks like the function `bubble_sort` in cell `4c3123c1` already does what you're asking for – it takes a list of integers and returns the same list sorted in ascending order. Would you like to modify this function, or perhaps try a different sorting algorithm like merge sort or quicksort?
- K** give a simpler version of it in a new cell
- I** Okay, I can provide a simpler version of the bubble sort function in a new cell. This version removes the docstring and the comments within the loop for brevity.
- K** use sort keyword in a new cell now
- I** Okay, I can show you how to use the built-in `sort()` method in a new cell.
- I** What can I help you build?

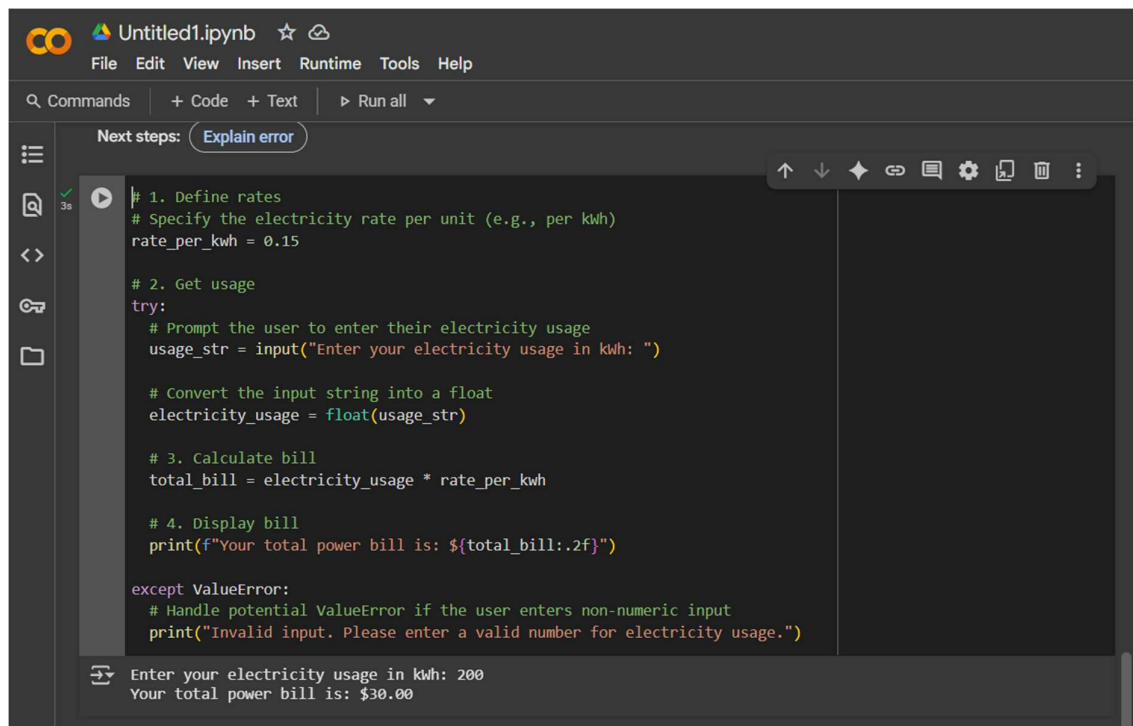
At the bottom of the chat window, it says: 'Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)'.

Observation: Gemini can understand the task very clearly. Gemini used try except made to sorted the given list and used sort() function.

Task-3:

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

Prompt-1: Generate a python code to calculate power bill.



```
Untitled1.ipynb
File Edit View Insert Runtime Tools Help

Next steps: Explain error

# 1. Define rates
# Specify the electricity rate per unit (e.g., per kwh)
rate_per_kwh = 0.15

# 2. Get usage
try:
    # Prompt the user to enter their electricity usage
    usage_str = input("Enter your electricity usage in kwh: ")

    # Convert the input string into a float
    electricity_usage = float(usage_str)

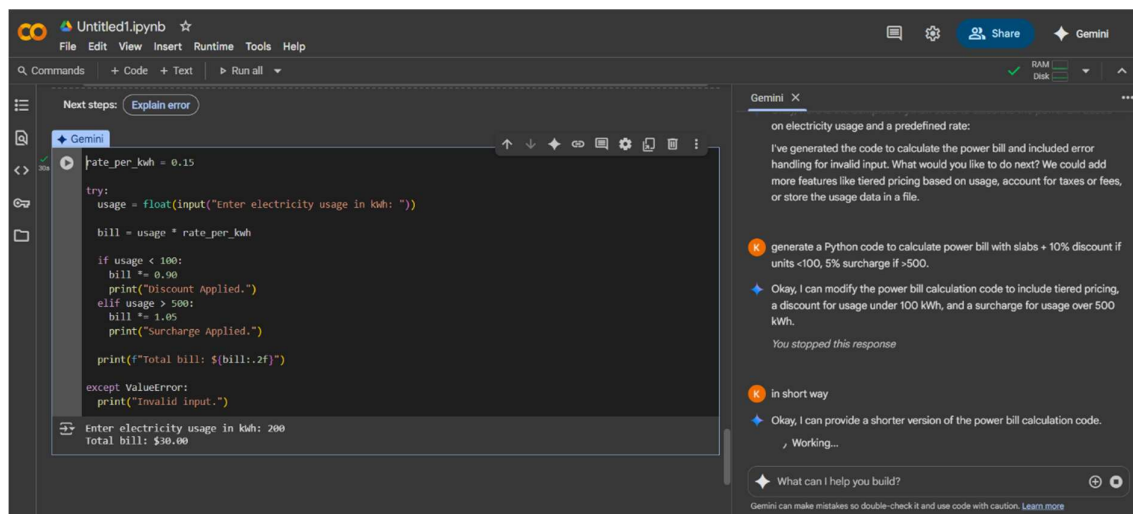
# 3. Calculate bill
total_bill = electricity_usage * rate_per_kwh

# 4. Display bill
print(f"Your total power bill is: ${total_bill:.2f}")

except ValueError:
    # Handle potential ValueError if the user enters non-numeric input
    print("Invalid input. Please enter a valid number for electricity usage.")

Enter your electricity usage in kwh: 200
Your total power bill is: $30.00
```

Prompt-2: Generate a python code to calculate power bill wit slabs +10% discount if units <100, 5% surcharge if >500.



```
Untitled1.ipynb
File Edit View Insert Runtime Tools Help

Next steps: Explain error

rate_per_kwh = 0.15

try:
    usage = float(input("Enter electricity usage in kwh: "))

    bill = usage * rate_per_kwh

    if usage < 100:
        bill *= 0.90
        print("Discount Applied.")
    elif usage > 500:
        bill *= 1.05
        print("Surcharge Applied.")

    print(f"Total bill: ${bill:.2f}")

except ValueError:
    print("Invalid input.")

Enter electricity usage in kwh: 200
Total bill: $30.00
```

on electricity usage and a predefined rate:

I've generated the code to calculate the power bill and included error handling for invalid input. What would you like to do next? We could add more features like tiered pricing based on usage, account for taxes or fees, or store the usage data in a file.

generate a Python code to calculate power bill with slabs + 10% discount if units <100, 5% surcharge if >500.

Okay, I can modify the power bill calculation code to include tiered pricing, a discount for usage under 100 kWh, and a surcharge for usage over 500 kWh.

You stopped this response

in short way

Okay, I can provide a shorter version of the power bill calculation code.

Working...

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Prompt-3: Write a python that asks the user for the number of electricity units consumed. Use these slab rates : Frist 100 units for %5 per unit next 100 units for \$7 per unit above 200 units for \$10 per unit finally, print total bill.

The screenshot shows a Jupyter Notebook environment with a file named 'Untitled1.ipynb'. The code in the notebook is a Python program that calculates electricity bills based on different slab rates. The program prompts the user to enter the number of electricity units consumed and then calculates the total bill based on the following rates: 100 units or less at ₹5 per unit, 100 to 200 units at ₹7 per unit, and above 200 units at ₹10 per unit. The program also includes an exception handler for invalid input. The output of the program shows that for 1000 units, the total bill is ₹9200.00. To the right of the notebook, there is a Gemini AI chat window. The chat history shows a prompt asking for a Python program to calculate electricity bills based on specific slab rates, and the AI's response, which includes the code and a confirmation that it has generated the code. The chat window also shows a follow-up question about adding more complex slab structures, taxes, or fixed charges, and the AI's response suggesting that it can help with that as well.

```
try:
    units = int(input("Enter the number of electricity units consumed: "))
    total_bill = 0
    if units <= 100:
        total_bill = units * 5
    elif units <= 200:
        total_bill = (100 * 5) + ((units - 100) * 7)
    else:
        total_bill = (100 * 5) + (100 * 7) + ((units - 200) * 10)
    print(f"Your total power bill is: ₹{total_bill:.2f}")
except ValueError:
    print("Invalid input. Please enter a valid integer for the number of units.")
```

Enter the number of electricity units consumed: 1000
Your total power bill is: ₹9200.00

Gemini X

Write a Python program that asks the user for the number of electricity units consumed. Use these slab rates: First 100 units → ₹5 per unit Next 100 units → ₹7 per unit Above 200 units → ₹10 per unit Finally, print the total bill."

Okay, I can write a Python program that calculates the power bill based on the electricity units consumed and the specified slab rates.

I have generated the code to calculate the power bill based on the slab rates you provided. What would you like to do next? We could add more complex slab structures, include taxes or fixed charges, or explore ways to store the billing information.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Observation: I have given different types of prompts by changing the electricity units conditions but the Gemini AI can easily given the code.

Task-4:

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

Prompt: Create a python program with two functions `login_user()` and `register_user()`. Use a dictionary { username, password } to store users .Add menu-driven loop where users can register like username, password, phnumber ,age ,clgname, login and quit.

```
[ ] 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_name}
    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.")
```

Gemini ✕

create a Python program with two functions: register_user() and login_user(). Use a dictionary (username: password) to store users. Add a menu-driven loop where users can register like username ,password,ph number,age,cig name. login, or quit

Absolutely! Here is the Python program with the register_user() and login_user() functions and a menu-driven loop.

The program is ready. You can now run the cell to interact with the registration and login system. Would you like to add more features, such as saving user data to a file, or perhaps adding more user details to the profile?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

```
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()
```

Gemini ✕

create a Python program with two functions: register_user() and login_user(). Use a dictionary (username: password) to store users. Add a menu-driven loop where users can register like username ,password,ph number,age,cig name. login, or quit

Absolutely! Here is the Python program with the register_user() and login_user() functions and a menu-driven loop.

The program is ready. You can now run the cell to interact with the registration and login system. Would you like to add more features, such as saving user data to a file, or perhaps adding more user details to the profile?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Output:

```
Menu:
1. Register
2. Login
3. Quit
Enter your choice: 1
Enter username: pardhu
Enter password: pardhu98
Enter phone number: 8121454932
Enter age: 19
Enter college name: sru
Registration successful!

Menu:
1. Register
2. Login
3. Quit
Enter your choice: 2
Enter username: pardhu
Enter password: pardhu98
Login successful!
User details:
password: pardhu98
phone_number: 8121454932
age: 19
college_name: sru

Menu:
1. Register
2. Login
3. Quit
Enter your choice: 3
Exiting program.
```

Gemini ✕

create a Python program with two functions: register_user() and login_user(). Use a dictionary (username: password) to store users. Add a menu-driven loop where users can register like username ,password,ph number,age,cig name. login, or quit

Absolutely! Here is the Python program with the register_user() and login_user() functions and a menu-driven loop.

The program is ready. You can now run the cell to interact with the registration and login system. Would you like to add more features, such as saving user data to a file, or perhaps adding more user details to the profile?

What can I help you build?

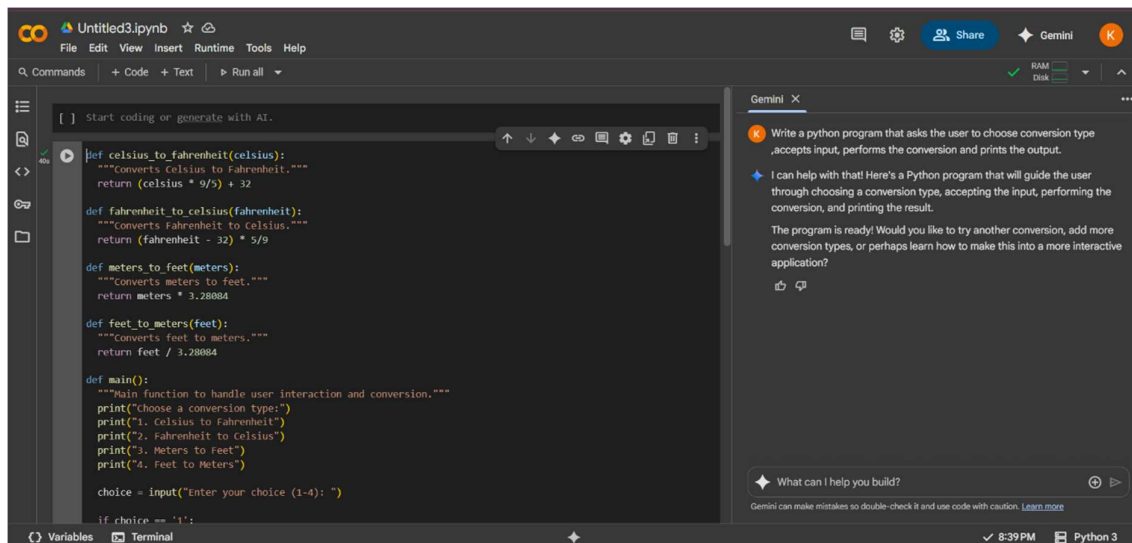
Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Observation: Gemini AI used two linked functions and Used dictionary to store username and password. It can understand my prompt very easily. It gives correct output.

Task-5:

Description: Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions.

Prompt: Write a python program that asks the user to choose conversion type ,accepts input, performs the conversion and prints the output.



The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb' with a Python program for temperature conversion. The program includes functions for Celsius to Fahrenheit, Fahrenheit to Celsius, meters to feet, and feet to meters, along with a main function for user interaction. A Gemini AI chat window is open on the right, displaying a prompt and a response.

```
[ ] Start coding or generate with AI.
```

```
def celsius_to_fahrenheit(celsius):  
    """Converts Celsius to Fahrenheit."""  
    return (celsius * 9/5) + 32  
  
def fahrenheit_to_celsius(fahrenheit):  
    """Converts Fahrenheit to Celsius."""  
    return (fahrenheit - 32) * 5/9  
  
def meters_to_feet(meters):  
    """Converts meters to feet."""  
    return meters * 3.28084  
  
def feet_to_meters(feet):  
    """Converts feet to meters."""  
    return feet / 3.28084  
  
def main():  
    """Main function to handle user interaction and conversion."""  
    print("Choose a conversion type:")  
    print("1. Celsius to Fahrenheit")  
    print("2. Fahrenheit to Celsius")  
    print("3. Meters to feet")  
    print("4. Feet to Meters")  
    choice = input("Enter your choice (1-4): ")  
    if choice == '1':
```

Gemini X

Write a python program that asks the user to choose conversion type ,accepts input, performs the conversion and prints the output.

I can help with that! Here's a Python program that will guide the user through choosing a conversion type, accepting the input, performing the conversion, and printing the result.

The program is ready! Would you like to try another conversion, add more conversion types, or perhaps learn how to make this into a more interactive application?

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

8:39 PM Python 3

The screenshot shows a Jupyter Notebook titled 'Untitled3.ipynb'. The code cell contains a Python program that asks the user to choose a conversion type (1-4) and then performs the conversion. The program uses functions like `celsius_to_fahrenheit` and `feet_to_meters`. A Gemini AI chat window is open on the right, showing a conversation about writing a Python program for unit conversion. The chat window includes a prompt, a response from Gemini, and a follow-up question.

```
print("choose a conversion type:")
print("1. Celsius to Fahrenheit")
print("2. Fahrenheit to Celsius")
print("3. Meters to Feet")
print("4. Feet to Meters")

choice = input("Enter your choice (1-4): ")

if choice == '1':
    celsius = float(input("Enter temperature in Celsius: "))
    fahrenheit = celsius_to_fahrenheit(celsius)
    print(f"{celsius}°C is equal to {fahrenheit}°F")
elif choice == '2':
    fahrenheit = float(input("Enter temperature in Fahrenheit: "))
    celsius = fahrenheit_to_celsius(fahrenheit)
    print(f"{fahrenheit}°F is equal to {celsius}°C")
elif choice == '3':
    meters = float(input("Enter length in meters: "))
    feet = meters_to_feet(meters)
    print(f"{meters} meters is equal to {feet} feet")
elif choice == '4':
    feet = float(input("Enter length in feet: "))
    meters = feet_to_meters(feet)
    print(f"{feet} feet is equal to {meters} meters")
else:
    print("Invalid choice.")

if __name__ == "__main__":
    main()
```

Choose a conversion type:

Output:

The screenshot shows the same Jupyter Notebook after execution. The code cell is now empty, and the output cell displays the program's output. The output shows the user choosing option 1, entering a temperature of 230, and receiving the converted temperature in Fahrenheit (446.0°F).

```
feet = float(input("Enter length in feet: "))
meters = feet_to_meters(feet)
print(f"{feet} feet is equal to {meters} meters")
else:
    print("Invalid choice.")

if __name__ == "__main__":
    main()
```

Choose a conversion type:
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
3. Meters to Feet
4. Feet to Meters
Enter your choice (1-4): 1
Enter temperature in Celsius: 230
230.0°C is equal to 446.0°F

Observation: Gemini AI used functions and loops method .The code is understanding , clean, readable and clear . the output given by temperature symbols

