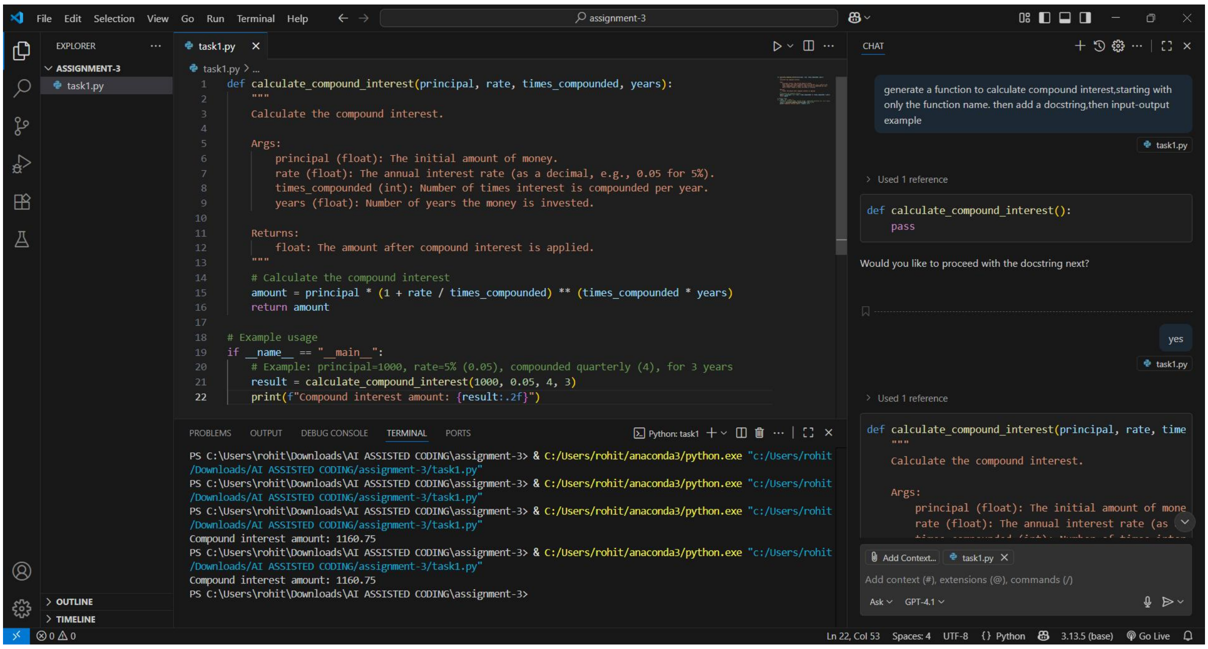
AI ASSISTED CODING LAB

ASSIGNMENT 3.2

NAME: Guangsinlung Phaomei
ENROLLMENT NO: 2503A51L20
BATCH NO: 19

TASK DESCRIPTION 1: Ask AI to write a function to calculate compound interest, starting with only the function name. Then add a docstring, then input-output example

PROMPT 1: Generate a Python function to calculate compound interest, starting with only the function name. Then add a docstring. Use the following parameters: Principal = 1000, Rate = 5% (0.05), Compounded quarterly (n = 4), Time = 3 years.



```python
def calculate_compound_interest(principal, rate, times_compounded, years):
    """
    Calculate the compound interest.

    Args:
        principal (float): The initial amount of money.
        rate (float): The annual interest rate (as a decimal, e.g., 0.05 for 5%).
        times_compounded (int): Number of times interest is compounded per year.
        years (float): Number of years the money is invested.

    Returns:
        float: The amount after compound interest is applied.
    """
    # Calculate the compound interest
    amount = principal * (1 + rate / times_compounded) ** (times_compounded * years)
    return amount

# Example usage
if __name__ == "__main__":
    # Example: principal=1000, rate=5% (0.05), compounded quarterly (4), for 3 years
    result = calculate_compound_interest(1000, 0.05, 4, 3)
    print(f"Compound interest amount: {result:.2f}")
```

```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit
/Downloads/AI ASSISTED CODING/assignment-3/task1.py"
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit
/Downloads/AI ASSISTED CODING/assignment-3/task1.py"
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit
/Downloads/AI ASSISTED CODING/assignment-3/task1.py"
Compound interest amount: 1160.75
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3> & C:/Users/rohit/anaconda3/python.exe "c:/Users/rohit
/Downloads/AI ASSISTED CODING/assignment-3/task1.py"
Compound interest amount: 1160.75
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3>
```

generate a function to calculate compound interest,starting with only the function name. then add a docstring,then input-output example

Used 1 reference

```python
def calculate_compound_interest():
    pass
```

Would you like to proceed with the docstring next?

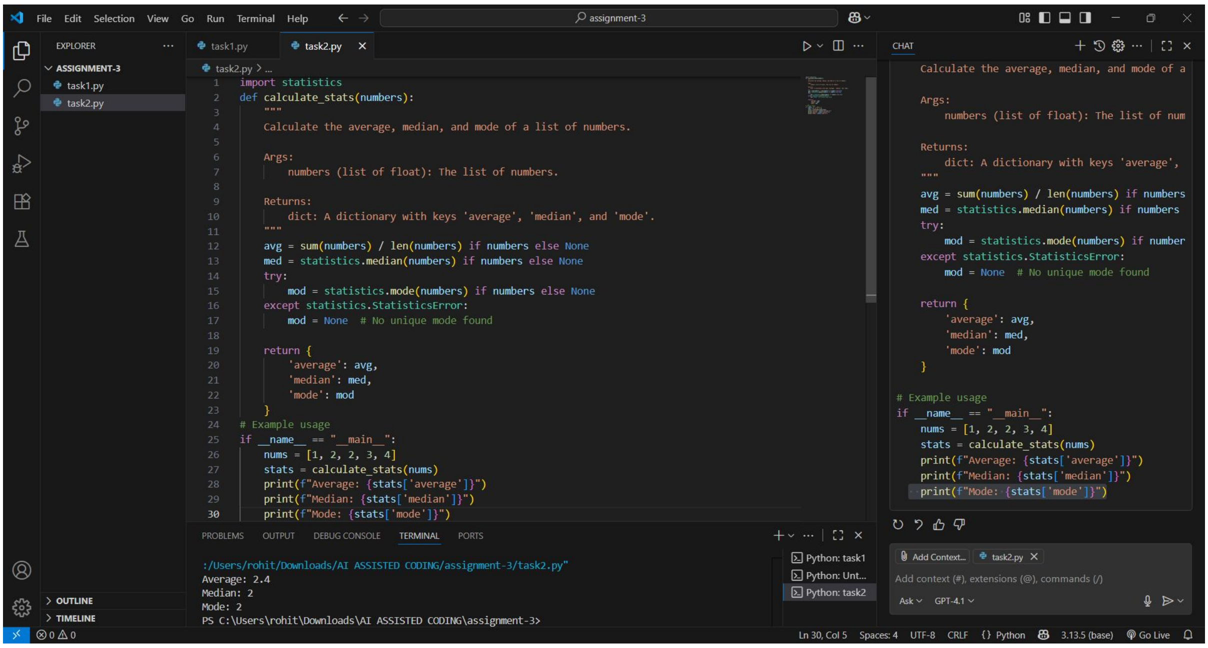
yes

Used 1 reference

```python
def calculate_compound_interest(principal, rate, time
    """
    Calculate the compound interest.

    Args:
        principal (float): The initial amount of mone
        rate (float): The annual interest rate (as
```

TASK DESCRIPTION 2: Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers.

PROMPT 1: Generate a Python function to calculate the **average, median, and mode** of a given list of numbers: [1, 2, 2, 3, 4].



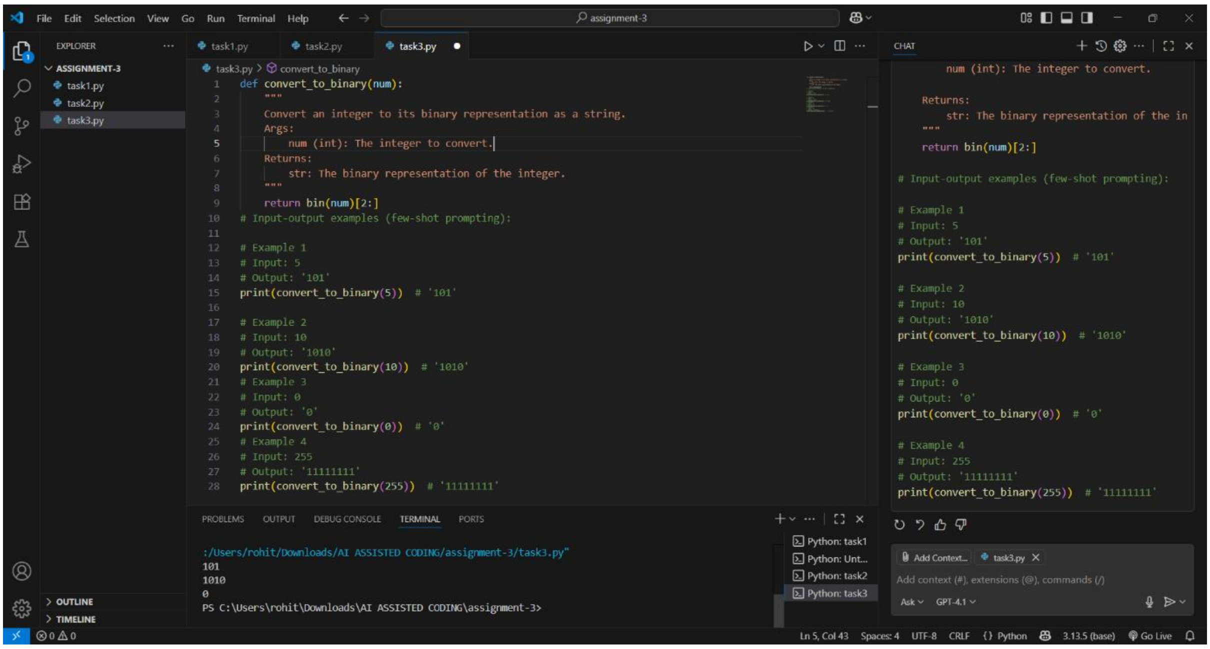
```python
import statistics
def calculate_stats(numbers):
    """
    Calculate the average, median, and mode of a list of numbers.

    Args:
        numbers (list of float): The list of numbers.

    Returns:
        dict: A dictionary with keys 'average', 'median', and 'mode'.
    """
    avg = sum(numbers) / len(numbers) if numbers else None
    med = statistics.median(numbers) if numbers else None
    try:
        mod = statistics.mode(numbers) if numbers else None
    except statistics.StatisticsError:
        mod = None  # No unique mode found

    return {
        'average': avg,
        'median': med,
        'mode': mod
    }
# Example usage
if __name__ == "__main__":
    nums = [1, 2, 2, 3, 4]
    stats = calculate_stats(nums)
    print(f"Average: {stats['average']}")
    print(f"Median: {stats['median']}")
    print(f"Mode: {stats['mode']}")
```

```
:/Users/rohit/Downloads/AI ASSISTED CODING/assignment-3/task2.py"
Average: 2.4
Median: 2
Mode: 2
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3>
```

```python
    Calculate the average, median, and mode of a

    Args:
        numbers (list of float): The list of num

    Returns:
        dict: A dictionary with keys 'average',
    """
    avg = sum(numbers) / len(numbers) if numbers
    med = statistics.median(numbers) if numbers
    try:
        mod = statistics.mode(numbers) if number
    except statistics.StatisticsError:
        mod = None  # No unique mode found

    return {
        'average': avg,
        'median': med,
        'mode': mod
    }

# Example usage
if __name__ == "__main__":
    nums = [1, 2, 2, 3, 4]
    stats = calculate_stats(nums)
    print(f"Average: {stats['average']}")
    print(f"Median: {stats['median']}")
    print(f"Mode: {stats['mode']}")
```

TASK DESCRIPTION 3: Provide multiple examples of input-output to the AI for convert_to_binary(num) function. Observe how AI uses few-shot prompting to generalize.

PROMPT 1: Generate Python code for a function convert_to_binary(num) that converts a number into its binary form. Provide multiple examples of input-output to guide the AI in generating generalized logic.


File Edit Selection View Go Run Terminal Help
assignment-3
EXPLORER
ASSIGNMENT-3
task1.py
task2.py
task3.py
task3.py > convert_to_binary
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python: task1
Python: Unt...
Python: task2
Python: task3
CHAT
Add Context... task3.py
Add context (#), extensions (@), commands (/)
Ask GPT-4.1
OUTLINE
TIMELINE
Ln 5, Col 43 Spaces: 4 UTF-8 CRLF {} Python 3.13.5 (base) Go Live

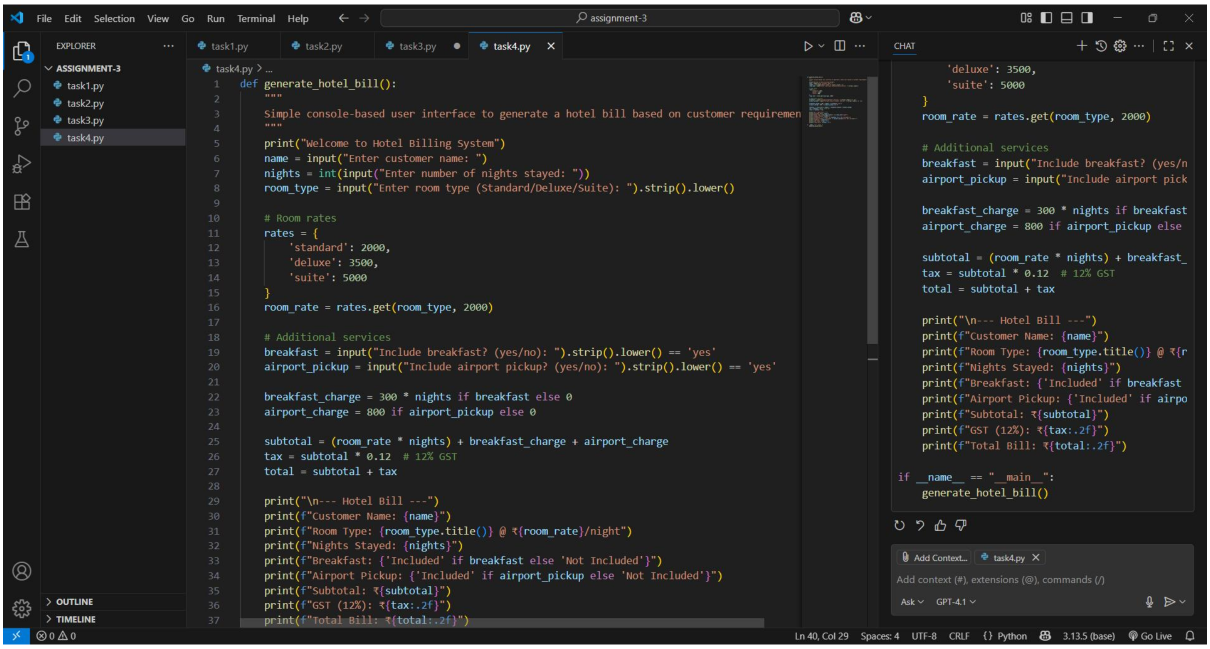
```python
def convert_to_binary(num):
    """
    Convert an integer to its binary representation as a string.
    Args:
        num (int): The integer to convert.
    Returns:
        str: The binary representation of the integer.
    """
    return bin(num)[2:]
# Input-output examples (few-shot prompting):

# Example 1
# Input: 5
# Output: '101'
print(convert_to_binary(5))  # '101'

# Example 2
# Input: 10
# Output: '1010'
print(convert_to_binary(10))  # '1010'
# Example 3
# Input: 0
# Output: '0'
print(convert_to_binary(0))  # '0'
# Example 4
# Input: 255
# Output: '11111111'
print(convert_to_binary(255))  # '11111111'
```

```
:/Users/rohit/Downloads/AI ASSISTED CODING/assignment-3/task3.py"
101
1010
0
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3>
```

```python
        num (int): The integer to convert.

    Returns:
        str: The binary representation of the in
    """
    return bin(num)[2:]

# Input-output examples (few-shot prompting):

# Example 1
# Input: 5
# Output: '101'
print(convert_to_binary(5))  # '101'

# Example 2
# Input: 10
# Output: '1010'
print(convert_to_binary(10))  # '1010'

# Example 3
# Input: 0
# Output: '0'
print(convert_to_binary(0))  # '0'

# Example 4
# Input: 255
# Output: '11111111'
print(convert_to_binary(255))  # '11111111'
```

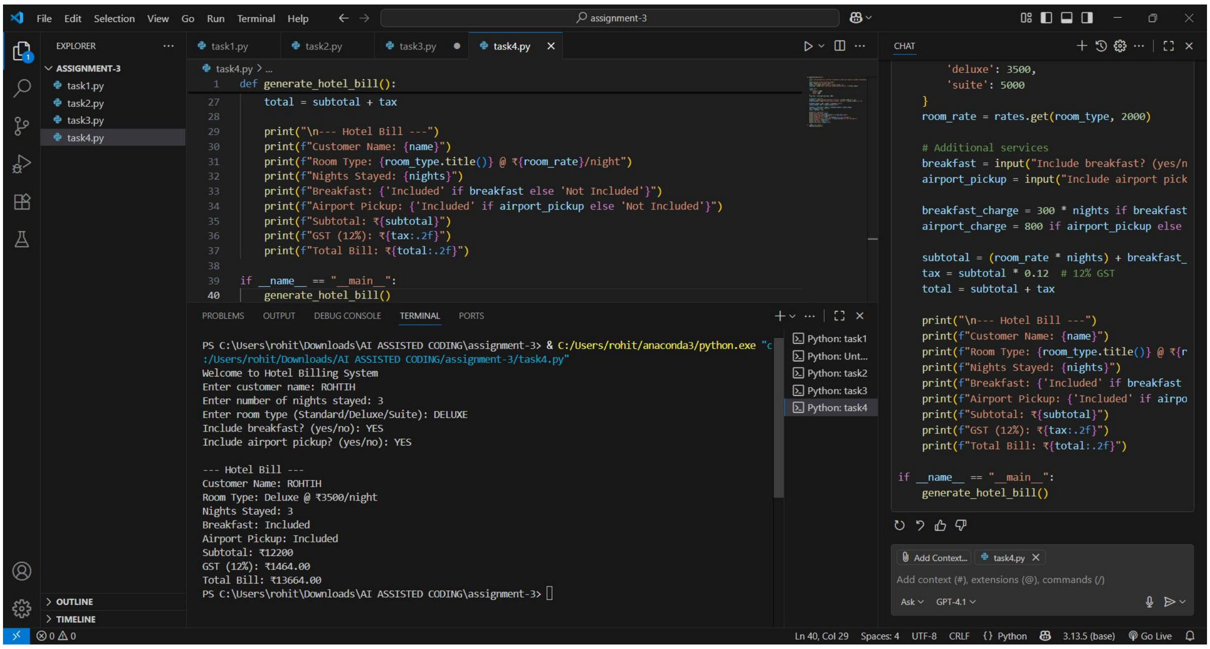
TASK DESCRIPTION 4: Create a user interface for a hotel to generate bill based on customer requirements

PROMPT 1: Generate Python code to create a user interface for a hotel billing system that calculates the total bill based on customer requirements (menu selection and quantity).


File Edit Selection View Go Run Terminal Help
assignment-3
EXPLORER
ASSIGNMENT-3
task1.py
task2.py
task3.py
task4.py
task4.py > ...
CHAT
Add Context... task4.py
Add context (#), extensions (@), commands (/)
Ask GPT-4.1
OUTLINE
TIMELINE
Ln 40, Col 29 Spaces: 4 UTF-8 CRLF {} Python 3.13.5 (base) Go Live

```python
def generate_hotel_bill():
    """
    Simple console-based user interface to generate a hotel bill based on customer requiremen
    """
    print("Welcome to Hotel Billing System")
    name = input("Enter customer name: ")
    nights = int(input("Enter number of nights stayed: "))
    room_type = input("Enter room type (Standard/Deluxe/Suite): ").strip().lower()

    # Room rates
    rates = {
        'standard': 2000,
        'deluxe': 3500,
        'suite': 5000
    }
    room_rate = rates.get(room_type, 2000)

    # Additional services
    breakfast = input("Include breakfast? (yes/no): ").strip().lower() == 'yes'
    airport_pickup = input("Include airport pickup? (yes/no): ").strip().lower() == 'yes'

    breakfast_charge = 300 * nights if breakfast else 0
    airport_charge = 800 if airport_pickup else 0

    subtotal = (room_rate * nights) + breakfast_charge + airport_charge
    tax = subtotal * 0.12  # 12% GST
    total = subtotal + tax

    print("\n--- Hotel Bill ---")
    print(f"Customer Name: {name}")
    print(f"Room Type: {room_type.title()} @ ₹{room_rate}/night")
    print(f"Nights Stayed: {nights}")
    print(f"Breakfast: {'Included' if breakfast else 'Not Included'}")
    print(f"Airport Pickup: {'Included' if airport_pickup else 'Not Included'}")
    print(f"Subtotal: ₹{subtotal}")
    print(f"GST (12%): ₹{tax:.2f}")
    print(f"Total Bill: ₹{total:.2f}")
```

```python
        'deluxe': 3500,
        'suite': 5000
    }
    room_rate = rates.get(room_type, 2000)

    # Additional services
    breakfast = input("Include breakfast? (yes/n
    airport_pickup = input("Include airport pick

    breakfast_charge = 300 * nights if breakfast
    airport_charge = 800 if airport_pickup else

    subtotal = (room_rate * nights) + breakfast_
    tax = subtotal * 0.12  # 12% GST
    total = subtotal + tax

    print("\n--- Hotel Bill ---")
    print(f"Customer Name: {name}")
    print(f"Room Type: {room_type.title()} @ ₹{r
    print(f"Nights Stayed: {nights}")
    print(f"Breakfast: {'Included' if breakfast
    print(f"Airport Pickup: {'Included' if airpo
    print(f"Subtotal: ₹{subtotal}")
    print(f"GST (12%): ₹{tax:.2f}")
    print(f"Total Bill: ₹{total:.2f}")

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



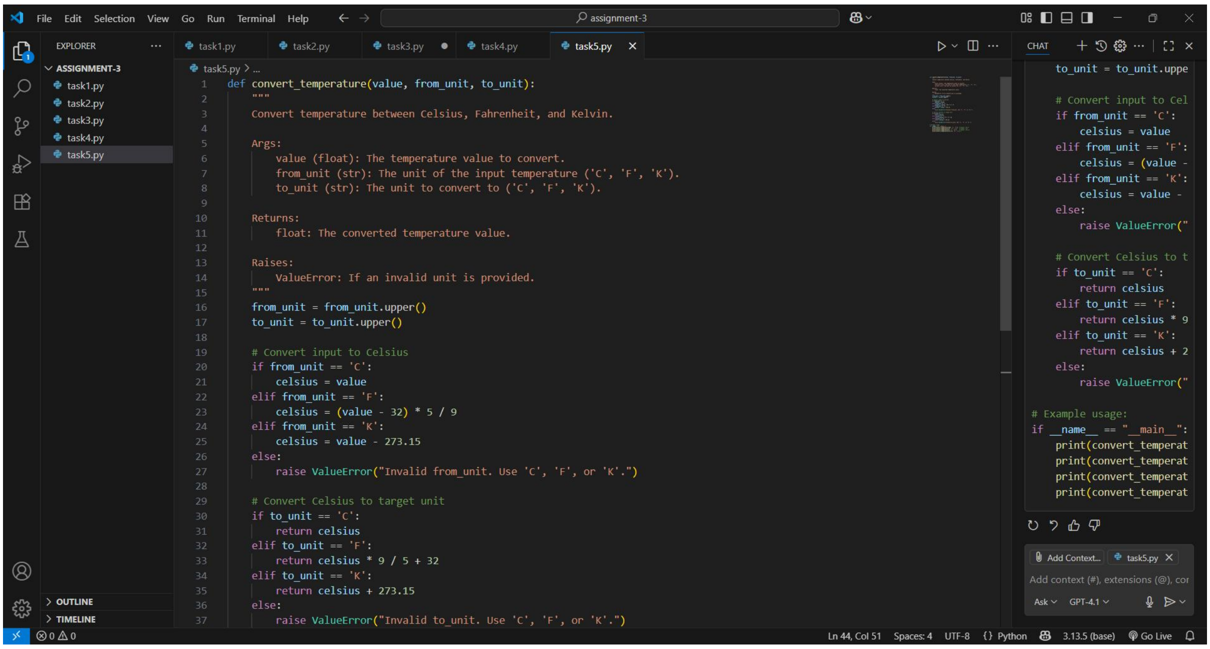
```python
def generate_hotel_bill():
    total = subtotal + tax

    print("\n--- Hotel Bill ---")
    print(f"Customer Name: {name}")
    print(f"Room Type: {room_type.title()} @ ₹{room_rate}/night")
    print(f"Nights Stayed: {nights}")
    print(f"Breakfast: {'Included' if breakfast else 'Not Included'}")
    print(f"Airport Pickup: {'Included' if airport_pickup else 'Not Included'}")
    print(f"Subtotal: ₹{subtotal}")
    print(f"GST (12%): ₹{tax:.2f}")
    print(f"Total Bill: ₹{total:.2f}")

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

```
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3> & C:/Users/rohit/anaconda3/python.exe "c
:/Users/rohit/Downloads/AI ASSISTED CODING/assignment-3/task4.py"
Welcome to Hotel Billing System
Enter customer name: ROHTIH
Enter number of nights stayed: 3
Enter room type (Standard/Deluxe/Suite): DELUXE
Include breakfast? (yes/no): YES
Include airport pickup? (yes/no): YES

--- Hotel Bill ---
Customer Name: ROHTIH
Room Type: Deluxe @ ₹3500/night
Nights Stayed: 3
Breakfast: Included
Airport Pickup: Included
Subtotal: ₹12200
GST (12%): ₹1464.00
Total Bill: ₹13664.00
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3> 
```

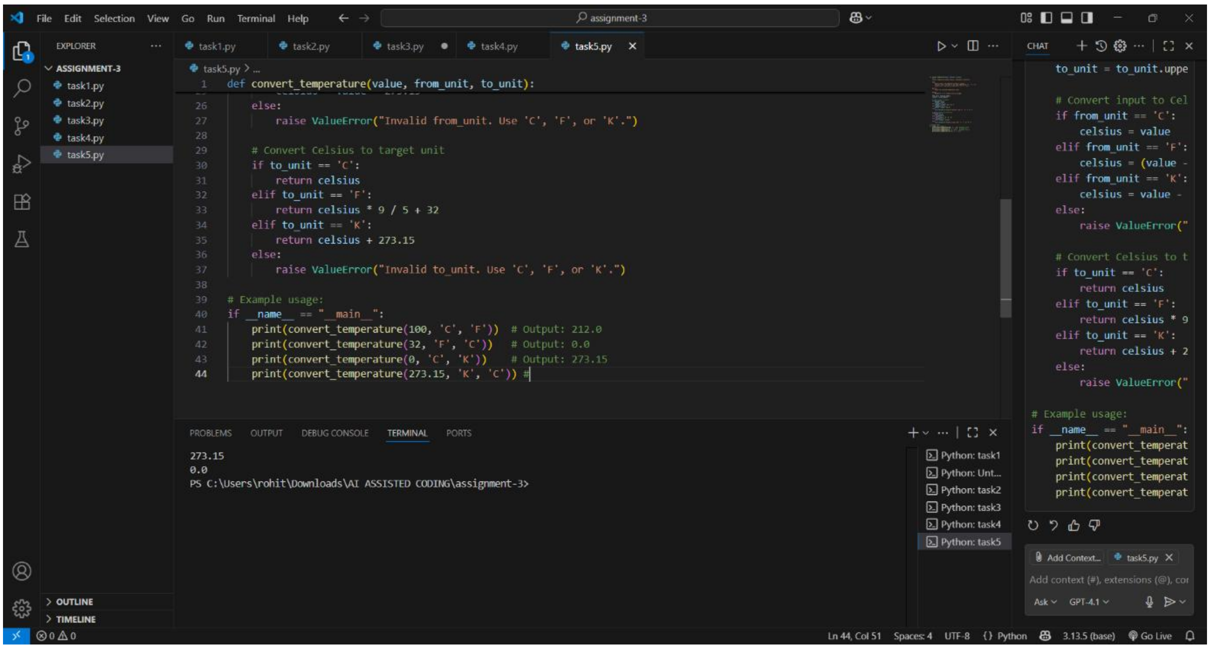
```python
        'deluxe': 3500,
        'suite': 5000
    }
    room_rate = rates.get(room_type, 2000)

    # Additional services
    breakfast = input("Include breakfast? (yes/n
    airport_pickup = input("Include airport pick

    breakfast_charge = 300 * nights if breakfast
    airport_charge = 800 if airport_pickup else

    subtotal = (room_rate * nights) + breakfast_
    tax = subtotal * 0.12  # 12% GST
    total = subtotal + tax

    print("\n--- Hotel Bill ---")
    print(f"Customer Name: {name}")
    print(f"Room Type: {room_type.title()} @ ₹{r
    print(f"Nights Stayed: {nights}")
    print(f"Breakfast: {'Included' if breakfast
    print(f"Airport Pickup: {'Included' if airpo
    print(f"Subtotal: ₹{subtotal}")
    print(f"GST (12%): ₹{tax:.2f}")
    print(f"Total Bill: ₹{total:.2f}")

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

TASK DESCRIPTION 5: Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions

PROMPT 1: Write a Python function that converts temperatures between Celsius and Fahrenheit. Accept two arguments: the temperature value and a string indicating the scale ('C' or 'F') Convert the temperature to the opposite scale (Celsius ↔ Fahrenheit)



```python
def convert_temperature(value, from_unit, to_unit):
    """
    Convert temperature between Celsius, Fahrenheit, and Kelvin.

    Args:
        value (float): The temperature value to convert.
        from_unit (str): The unit of the input temperature ('C', 'F', 'K').
        to_unit (str): The unit to convert to ('C', 'F', 'K').

    Returns:
        float: The converted temperature value.

    Raises:
        ValueError: If an invalid unit is provided.
    """
    from_unit = from_unit.upper()
    to_unit = to_unit.upper()

    # Convert input to Celsius
    if from_unit == 'C':
        celsius = value
    elif from_unit == 'F':
        celsius = (value - 32) * 5 / 9
    elif from_unit == 'K':
        celsius = value - 273.15
    else:
        raise ValueError("Invalid from_unit. Use 'C', 'F', or 'K'.")

    # Convert Celsius to target unit
    if to_unit == 'C':
        return celsius
    elif to_unit == 'F':
        return celsius * 9 / 5 + 32
    elif to_unit == 'K':
        return celsius + 273.15
    else:
        raise ValueError("Invalid to_unit. Use 'C', 'F', or 'K'.")
```



```python
def convert_temperature(value, from_unit, to_unit):
    else:
        raise ValueError("Invalid from_unit. Use 'C', 'F', or 'K'.")

    # Convert Celsius to target unit
    if to_unit == 'C':
        return celsius
    elif to_unit == 'F':
        return celsius * 9 / 5 + 32
    elif to_unit == 'K':
        return celsius + 273.15
    else:
        raise ValueError("Invalid to_unit. Use 'C', 'F', or 'K'.")

# Example usage:
if __name__ == "__main__":
    print(convert_temperature(100, 'C', 'F'))  # Output: 212.0
    print(convert_temperature(32, 'F', 'C'))   # Output: 0.0
    print(convert_temperature(0, 'C', 'K'))    # Output: 273.15
    print(convert_temperature(273.15, 'K', 'C')) #
```

```
273.15
0.0
PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-3>
```

OBSERVATION: This assignment demonstrates how **prompt engineering** can significantly influence the quality and accuracy of AI-generated code.

It begins with **simple prompts** (e.g., function name only) and gradually progresses to **more refined and detailed prompts**.

The process highlights how **adding clarity, examples, and specific requirements** improves AI output.

Few-shot prompting (Task 3) shows how AI can generalize from multiple examples.

The tasks illustrate a range of applications, from **mathematical functions** to **user interfaces**.

Overall, this exercise shows that **structured, specific, and context-rich prompts** yield more robust, readable, and accurate code.