AI-ASSISTED Coding Lab
Assignment 3
Enrollment No: 2503A51L44
Batch: 20
Student Name: Meer Burhan Ali Hashmi

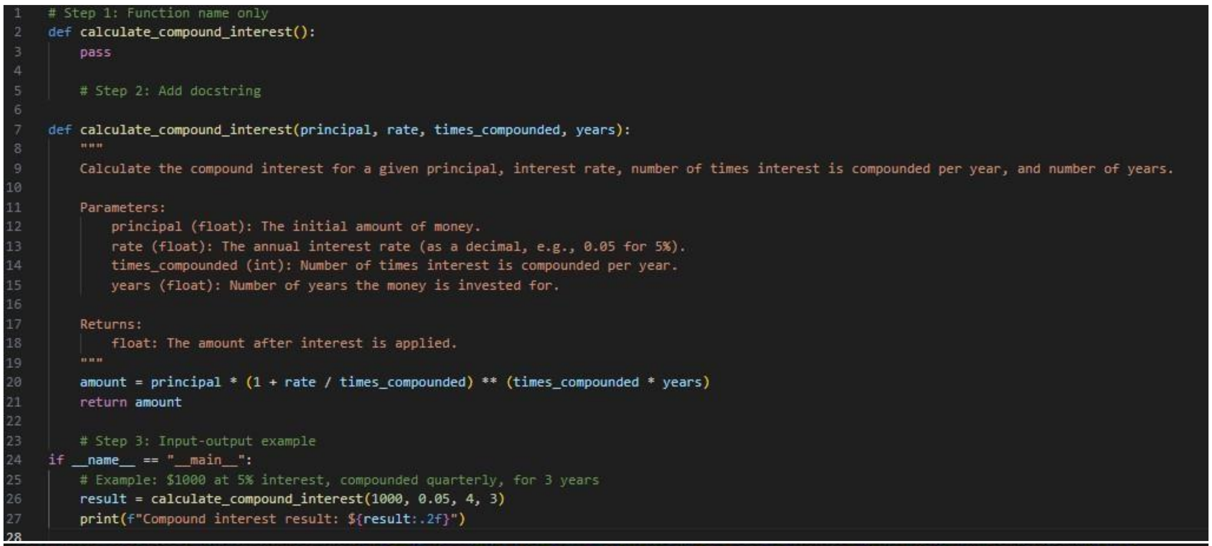
Task 1

Task Overview:
Use AI assistance to develop a Python function for calculating compound interest. Begin by defining only the function name. Then, enhance the function with:

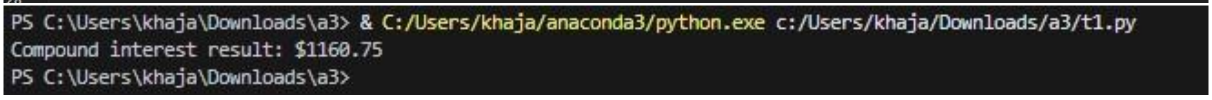
1. **A descriptive docstring explaining the purpose, input parameters, and return value.**
2. **An example showing how the function can be used along with the corresponding output.**

Prompt Given:
"Create a Python function to compute compound interest. Start by showing only the function name. Then, add a docstring explaining its functionality, parameters, and output. Finally, provide a usage example with input values and the resulting output."

CODE:-



```python
# Step 1: Function name only
def calculate_compound_interest():
    pass

    # Step 2: Add docstring

def calculate_compound_interest(principal, rate, times_compounded, years):
    """
    Calculate the compound interest for a given principal, interest rate, number of times interest is compounded per year, and number of years.

    Parameters:
        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 for.

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

    # Step 3: Input-output example
if __name__ == "__main__":
    # Example: $1000 at 5% interest, compounded quarterly, for 3 years
    result = calculate_compound_interest(1000, 0.05, 4, 3)
    print(f"Compound interest result: ${result:.2f}")

```



```
PS C:\Users\khaja\Downloads\a3> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/a3/t1.py
Compound interest result: $1160.75
PS C:\Users\khaja\Downloads\a3>
```

OUTPUT:-

Task 2

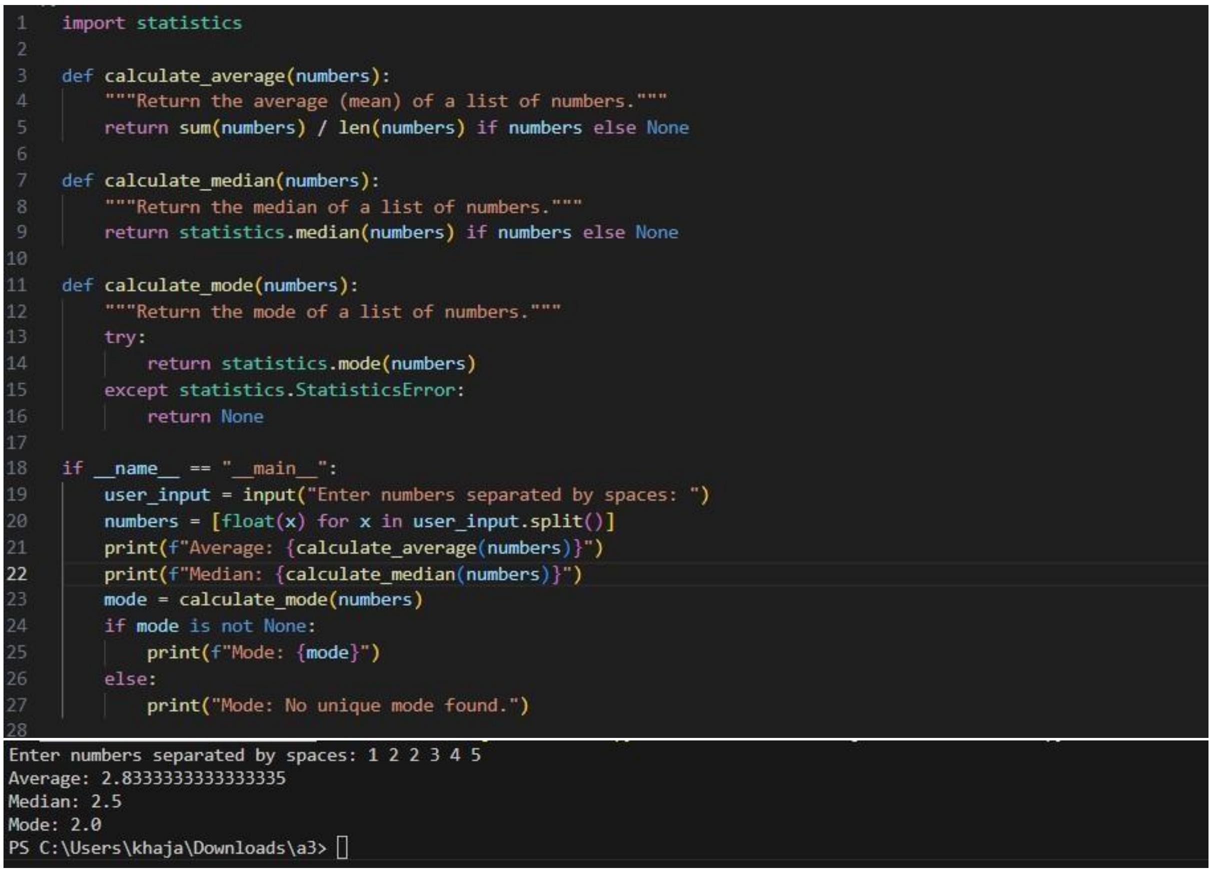
Task Overview:
Begin with a general mathematical computation task, then refine it to specifically:

Write a Python function to calculate the average, median, and mode of a list of numbers.

The function should accept the numbers entered by the user at runtime.

Prompt Given:
"Create a Python function that calculates the average, median, and mode for a list of numbers entered by the user during program execution."

CODE:-



```python
import statistics

def calculate_average(numbers):
    """Return the average (mean) of a list of numbers."""
    return sum(numbers) / len(numbers) if numbers else None

def calculate_median(numbers):
    """Return the median of a list of numbers."""
    return statistics.median(numbers) if numbers else None

def calculate_mode(numbers):
    """Return the mode of a list of numbers."""
    try:
        return statistics.mode(numbers)
    except statistics.StatisticsError:
        return None

if __name__ == "__main__":
    user_input = input("Enter numbers separated by spaces: ")
    numbers = [float(x) for x in user_input.split()]
    print(f"Average: {calculate_average(numbers)}")
    print(f"Median: {calculate_median(numbers)}")
    mode = calculate_mode(numbers)
    if mode is not None:
        print(f"Mode: {mode}")
    else:
        print("Mode: No unique mode found.")

```

```
Enter numbers separated by spaces: 1 2 2 3 4 5
Average: 2.8333333333333335
Median: 2.5
Mode: 2.0
PS C:\Users\khaja\Downloads\a3>
```

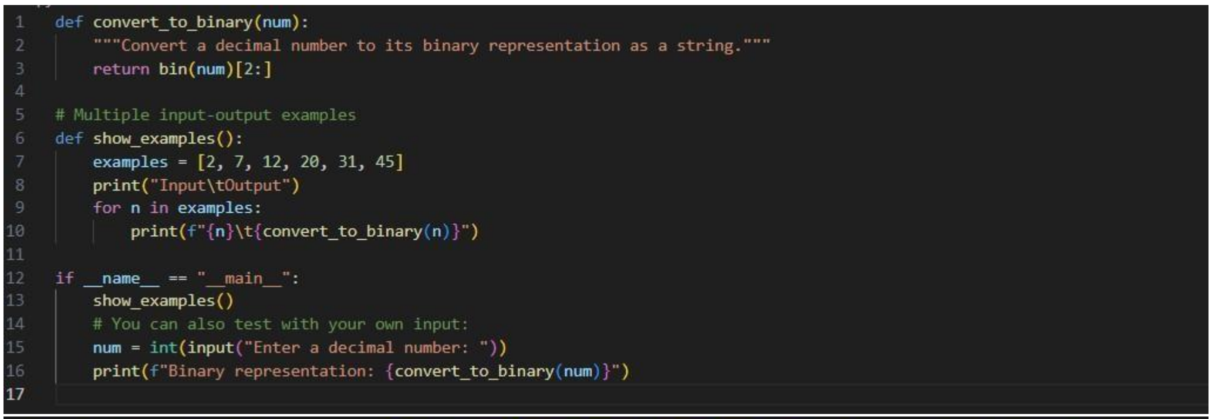
OUTPUT:-

Task 3

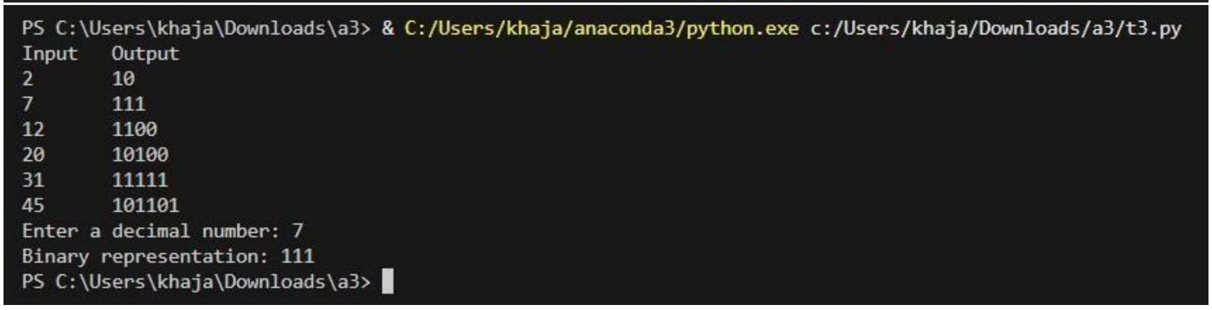
Task Overview:
Demonstrate few-shot prompting by supplying the AI with several input–output pairs for a function named convert_to_binary(num). This function is intended to transform a decimal number into its binary equivalent. By reviewing the examples, observe how the AI learns to generalize the pattern.

Prompt Used:
"Write a Python program that includes multiple sample inputs and outputs for a convert_to_binary(num) function, where the function converts a given decimal number into its binary representation."

CODE:-



```python
def convert_to_binary(num):
    """Convert a decimal number to its binary representation as a string."""
    return bin(num)[2:]

# Multiple input-output examples
def show_examples():
    examples = [2, 7, 12, 20, 31, 45]
    print("Input\tOutput")
    for n in examples:
        print(f"{n}\t{convert_to_binary(n)}")

if __name__ == "__main__":
    show_examples()
    # You can also test with your own input:
    num = int(input("Enter a decimal number: "))
    print(f"Binary representation: {convert_to_binary(num)}")
```



```
PS C:\Users\khaja\Downloads\a3> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/a3/t3.py
Input   Output
2       10
7       111
12      1100
20      10100
31      11111
45      101101
Enter a decimal number: 7
Binary representation: 111
PS C:\Users\khaja\Downloads\a3> 
```

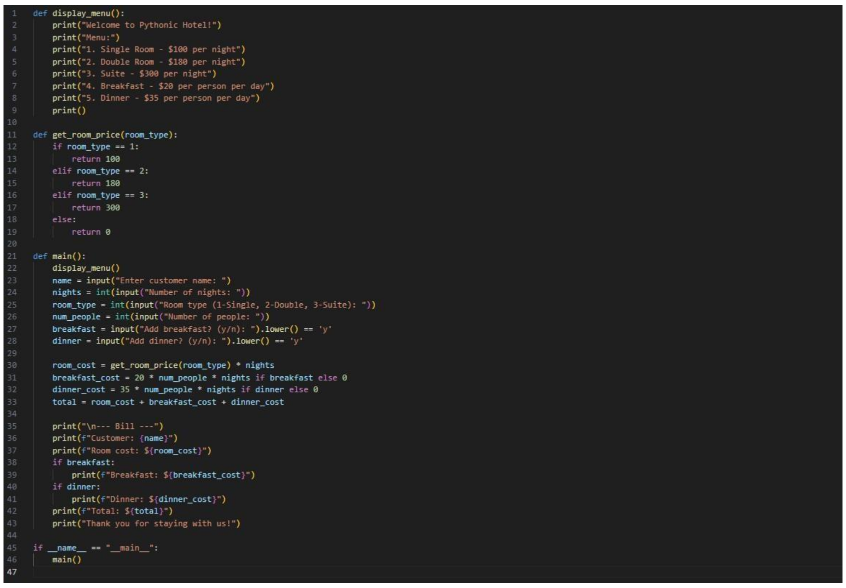
OUTPUT:-

Task 4

Task Overview:
Develop a Python-based user interface for a hotel billing system. The program should allow customers to select services, enter quantities, and automatically calculate the final bill according to their choices.

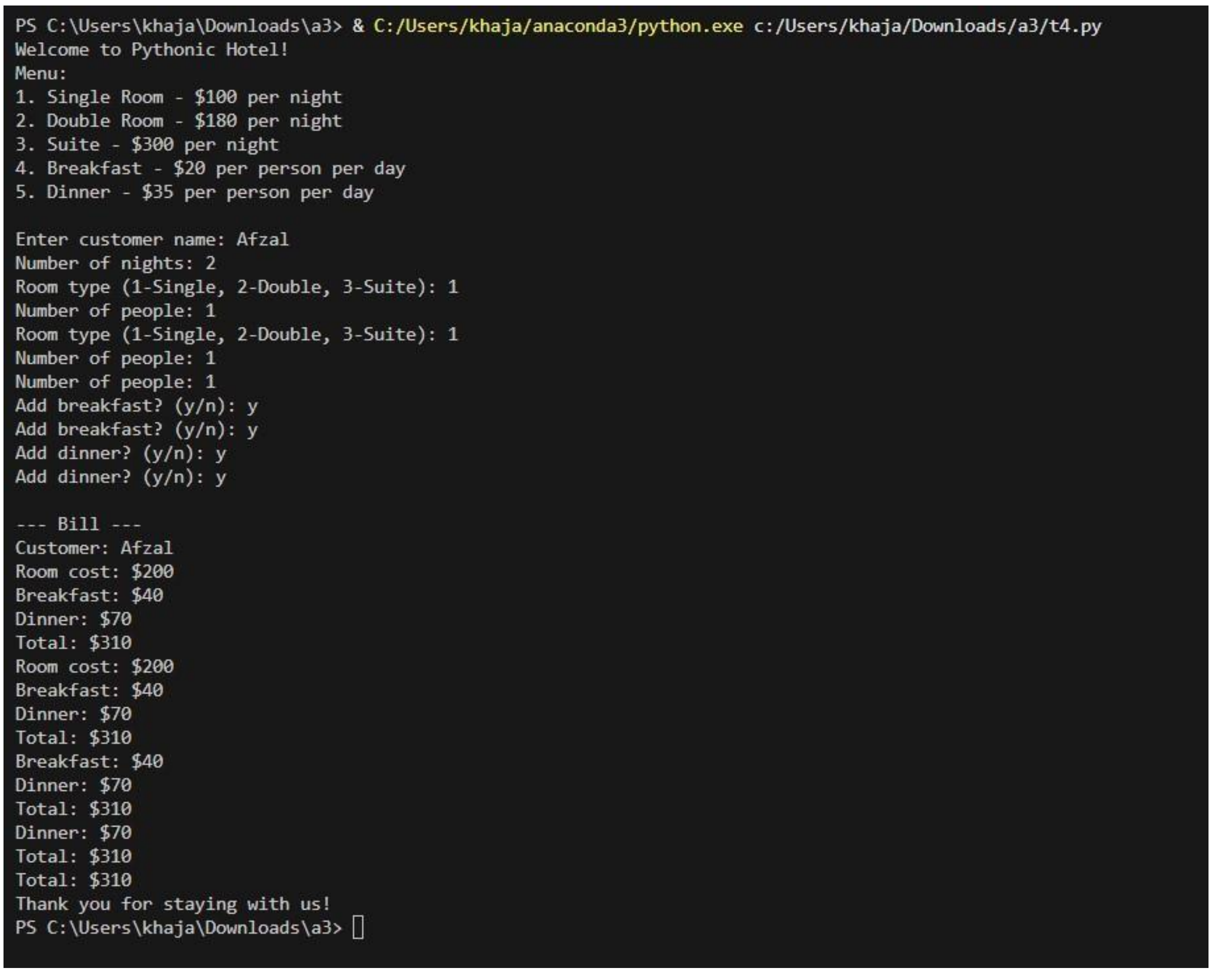
Prompt Used:
"Write a Python program to build a user interface for a hotel that generates a bill based on customer selections and requirements."

CODE



```python
def display_menu():
    print("Welcome to Pythonic Hotel!")
    print("Menu:")
    print("1. Single Room - $100 per night")
    print("2. Double Room - $180 per night")
    print("3. Suite - $300 per night")
    print("4. Breakfast - $20 per person per day")
    print("5. Dinner - $35 per person per day")
    print()

def get_room_price(room_type):
    if room_type == 1:
        return 100
    elif room_type == 2:
        return 180
    elif room_type == 3:
        return 300
    else:
        return 0

def main():
    display_menu()
    name = input("Enter customer name: ")
    nights = int(input("Number of nights: "))
    room_type = int(input("Room type (1-Single, 2-Double, 3-Suite): "))
    num_people = int(input("Number of people: "))
    breakfast = input("Add breakfast? (y/n): ").lower() == 'y'
    dinner = input("Add dinner? (y/n): ").lower() == 'y'

    room_cost = get_room_price(room_type) * nights
    breakfast_cost = 20 * num_people * nights if breakfast else 0
    dinner_cost = 35 * num_people * nights if dinner else 0
    total = room_cost + breakfast_cost + dinner_cost

    print("\n--- Bill ---")
    print(f"Customer: {name}")
    print(f"Room cost: ${room_cost}")
    if breakfast:
        print(f"Breakfast: ${breakfast_cost}")
    if dinner:
        print(f"Dinner: ${dinner_cost}")
    print(f"Total: ${total}")
    print("Thank you for staying with us!")

if __name__ == "__main__":
    main()

```

OUTPUT

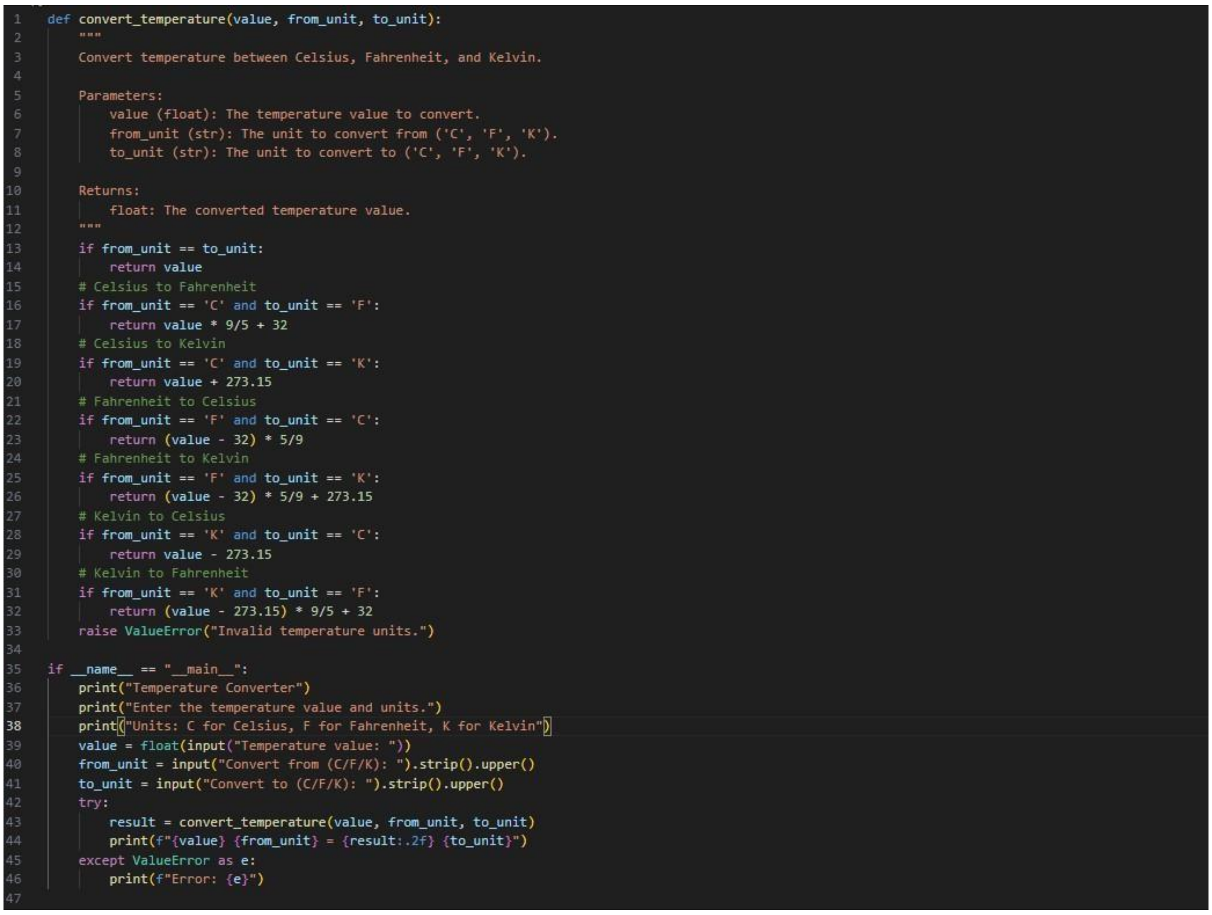

```
PS C:\Users\khaja\Downloads\a3> & C:/Users/khaja/anaconda3/python.exe c:/Users/khaja/Downloads/a3/t4.py
Welcome to Pythonic Hotel!
Menu:
1. Single Room - $100 per night
2. Double Room - $180 per night
3. Suite - $300 per night
4. Breakfast - $20 per person per day
5. Dinner - $35 per person per day

Enter customer name: Afzal
Number of nights: 2
Room type (1-Single, 2-Double, 3-Suite): 1
Number of people: 1
Room type (1-Single, 2-Double, 3-Suite): 1
Number of people: 1
Number of people: 1
Add breakfast? (y/n): y
Add breakfast? (y/n): y
Add dinner? (y/n): y
Add dinner? (y/n): y

--- Bill ---
Customer: Afzal
Room cost: $200
Breakfast: $40
Dinner: $70
Total: $310
Room cost: $200
Breakfast: $40
Dinner: $70
Total: $310
Breakfast: $40
Dinner: $70
Total: $310
Dinner: $70
Total: $310
Total: $310
Thank you for staying with us!
PS C:\Users\khaja\Downloads\a3>
```

TASK5

TASK5 DESCRIPTION:- Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions

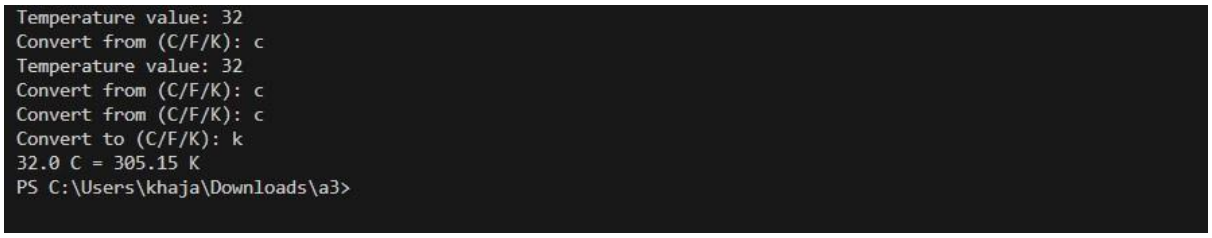
PROMPT:- Generate a python program to Improving Temperature Conversion Function with Clear Instructions

CODE:-



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

    Parameters:
        value (float): The temperature value to convert.
        from_unit (str): The unit to convert from ('C', 'F', 'K').
        to_unit (str): The unit to convert to ('C', 'F', 'K').

    Returns:
        float: The converted temperature value.
    """
    if from_unit == to_unit:
        return value
    # Celsius to Fahrenheit
    if from_unit == 'C' and to_unit == 'F':
        return value * 9/5 + 32
    # Celsius to Kelvin
    if from_unit == 'C' and to_unit == 'K':
        return value + 273.15
    # Fahrenheit to Celsius
    if from_unit == 'F' and to_unit == 'C':
        return (value - 32) * 5/9
    # Fahrenheit to Kelvin
    if from_unit == 'F' and to_unit == 'K':
        return (value - 32) * 5/9 + 273.15
    # Kelvin to Celsius
    if from_unit == 'K' and to_unit == 'C':
        return value - 273.15
    # Kelvin to Fahrenheit
    if from_unit == 'K' and to_unit == 'F':
        return (value - 273.15) * 9/5 + 32
    raise ValueError("Invalid temperature units.")

if __name__ == "__main__":
    print("Temperature Converter")
    print("Enter the temperature value and units.")
    print("Units: C for Celsius, F for Fahrenheit, K for Kelvin")
    value = float(input("Temperature value: "))
    from_unit = input("Convert from (C/F/K): ").strip().upper()
    to_unit = input("Convert to (C/F/K): ").strip().upper()
    try:
        result = convert_temperature(value, from_unit, to_unit)
        print(f"{value} {from_unit} = {result:.2f} {to_unit}")
    except ValueError as e:
        print(f"Error: {e}")
```

OUTPUT:-



```
Temperature value: 32
Convert from (C/F/K): c
Temperature value: 32
Convert from (C/F/K): c
Convert from (C/F/K): c
Convert to (C/F/K): k
32.0 C = 305.15 K
PS C:\Users\khaja\Downloads\a3>
```

OBSERVATION:- From this assignment, I observed the practical role of prompt engineering and AI-assisted coding in generating Python programs. By giving different levels of instructions, the AI was able to produce complete implementations, examples, and even user interfaces.

- In Task 1, starting with only a function name and gradually adding docstrings and examples demonstrated how AI understands step-by-step instructions and builds code systematically.
- In Task 2, I observed how runtime inputs can be used for statistical calculations (average, median, mode), showing AI's capability to handle mathematical logic on user-provided data.
- In Task 3, by providing multiple input–output examples for the convert_to_binary(num) function, I noticed how AI applied few-shot prompting to generalize and generate correct binary conversions for any decimal input.
- In Task 4, the hotel billing program highlighted how AI can extend beyond simple functions to build user-oriented applications, combining logic with interface design.
- In Task 5, refining the temperature conversion function showed how prompt specificity directly affects the accuracy, clarity, and usability of AI-generated code.