LAB NAME : **AI ASSISTED CODING**

LAB NUMBER :**02**

ROLL NO :**2503A51L16**

BRANCH : **CSE**

NAME : **K.JASHUVA**

TASK 1

Task Description: 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: Generate a Python function named calculate_compound_interest that takes principal, annual interest rate, time in years, and compounding frequency as parameters.

CODE:

```python
def calculate_compound_interest(principal: float, rate: float, time: float, frequency: int) -> float:
    amount = principal * (1 + (rate / 100) / frequency) ** (frequency * time)
    return amount


# ===== User Input Section =====
try:
    p = float(input("Enter the principal amount: "))
    r = float(input("Enter the annual interest rate (in %): "))
    t = float(input("Enter the time in years: "))
    n = int(input("Enter the compounding frequency per year (e.g., 1 for yearly, 4 for quarterly): "))

    result = calculate_compound_interest(p, r, t, n)
    print(f"\nFinal amount after {t} years: {result:.2f}")
except ValueError:
    print("Invalid input. Please enter numeric values only.")
```

OUTPUT:

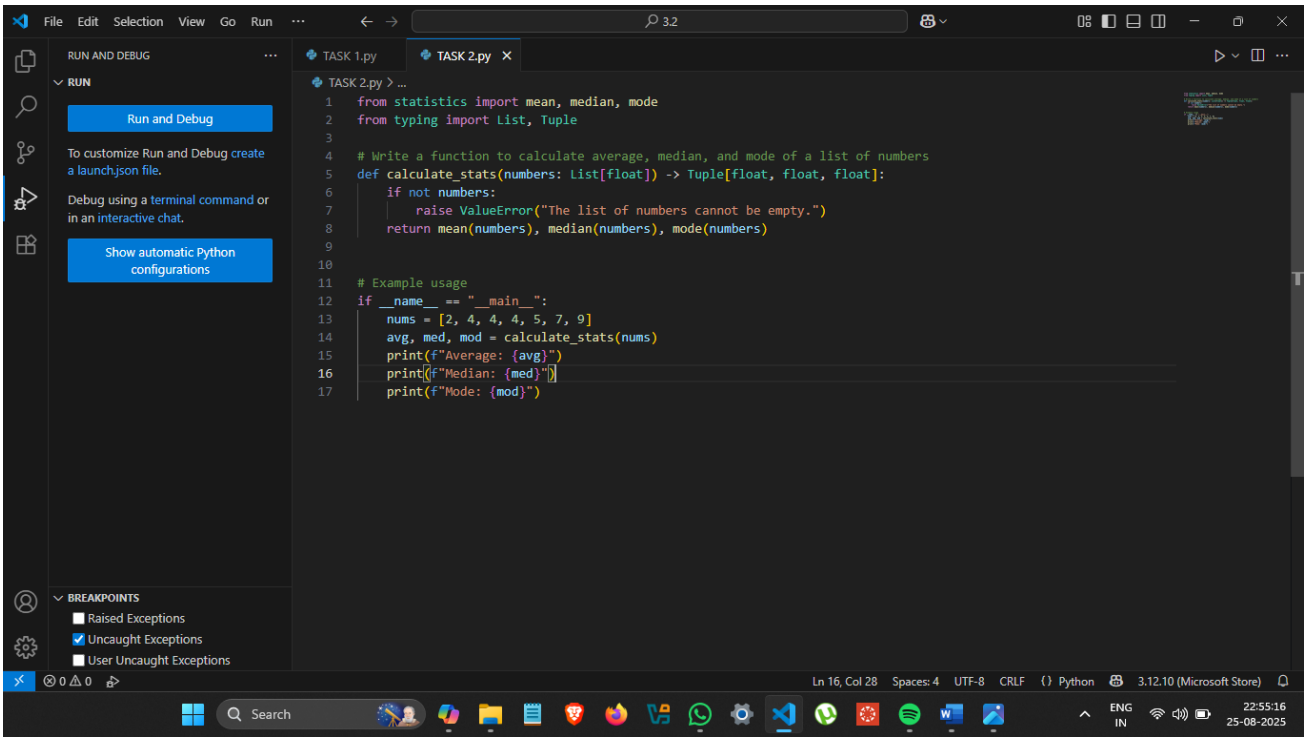
```
PS C:\B.TECH\3.2>  c:; cd 'c:\B.TECH\3.2'; & 'c:\Users\kamer\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\kamer\
.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '57971' '--' 'C:\B.TECH\3.2\TASK 1.py'
Enter the principal amount: 10000
Enter the annual interest rate (in %): 8
Enter the time in years: 5
Enter the compounding frequency per year (e.g., 1 for yearly, 4 for quarterly): 1

Final amount after 5.0 years: 14693.28
PS C:\B.TECH\3.2>
```

TASK 2

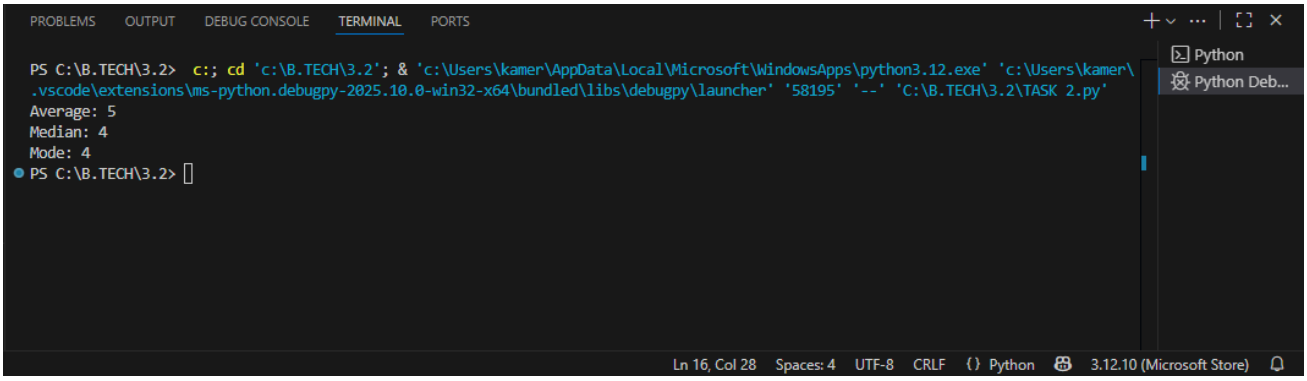
Task Description: Do math stuff, then refine it to: # Write a function to calculate average, median, and mode of a list of numbers**.**

PROMPT: Generate a Python function with type hints that calculates the average, median, and mode of a list of numbers, that includes a clear docstring, and handles empty lists or invalid inputs gracefully**.**

CODE:



OUTPUT:



TASK 3

Task Description: 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: Provide multiple examples of input-output for the convert_to_binary(num) function to help the AI generalize using few-shot prompting.

CODE:

```python
def convert_to_binary(num: int) -> str:

    if num < 0:
        raise ValueError("Only non-negative integers are supported.")
    return bin(num)[2:]  # slice off the '0b' prefix


# Testing the function with example inputs
test_values = [5, 8, 15, 0, 1, 42, 255]
for value in test_values:
    print(f"Input: {value} -> Output: {convert_to_binary(value)}")
```

OUTPUT:

```
PS C:\B.TECH\3.2>  c:; cd 'c:\B.TECH\3.2'; & 'c:\Users\kamer\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\kamer\
.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '58412' '--' 'C:\B.TECH\3.2\TASK 3.py'
Input: 5 -> Output: 101
Input: 8 -> Output: 1000
Input: 15 -> Output: 1111
Input: 0 -> Output: 0
Input: 1 -> Output: 1
Input: 42 -> Output: 101010
Input: 255 -> Output: 11111111
PS C:\B.TECH\3.2>
```

TASK 4

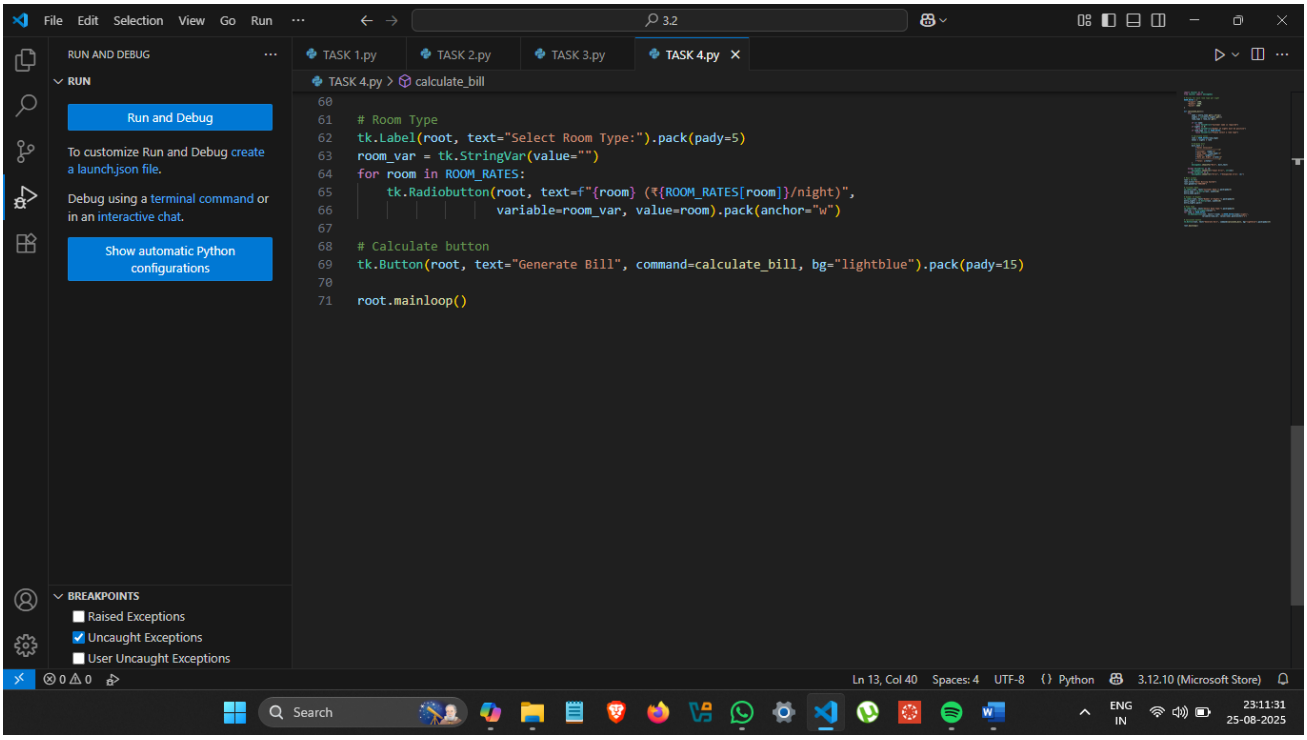
Task Description: Create a user interface for a hotel to generate bill based on customer requirements.

PROMPT: Create a user interface for a hotel billing system that generates a bill based on customer requirements.

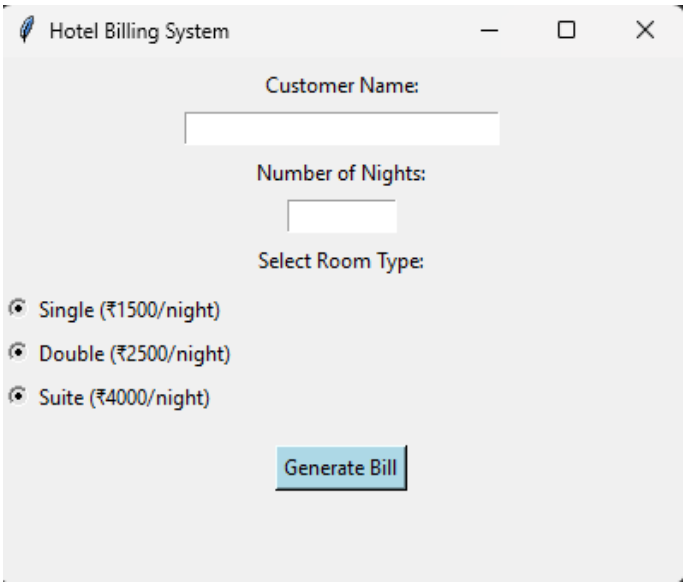
CODE:

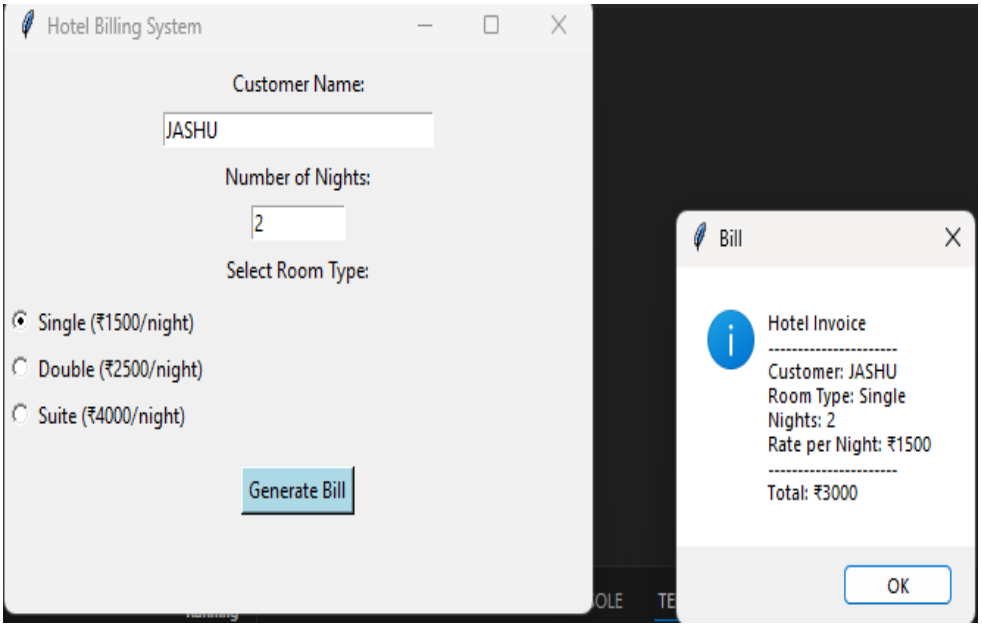
```python
import tkinter as tk
from tkinter import messagebox

# Prices for each room type per night
ROOM_RATES = {
    "Single": 1500,
    "Double": 2500,
    "Suite": 4000
}

def calculate_bill():
    try:
        name = entry_name.get().strip()
        nights = int(entry_nights.get())
        room_type = room_var.get()

        if not name:
            raise ValueError("Customer name is required")
        if nights <= 0:
            raise ValueError("Number of nights must be positive")
        if room_type not in ROOM_RATES:
            raise ValueError("Please select a room type")

        # Calculate total
        rate = ROOM_RATES[room_type]
        total = nights * rate

        # Display bill
        bill_text = (
            f"Hotel Invoice\n"
            f"----------------------\n"
            f"Customer: {name}\n"
```

```python
def calculate_bill():
            f"----------------------\n"
            f"Customer: {name}\n"
            f"Room Type: {room_type}\n"
            f"Nights: {nights}\n"
            f"Rate per Night: ₹{rate}\n"
            f"----------------------\n"
            f"Total: ₹{total}"
        )
        messagebox.showinfo("Bill", bill_text)

    except ValueError as ve:
        messagebox.showerror("Input Error", str(ve))
    except Exception as e:
        messagebox.showerror("Error", f"Unexpected error: {e}")

# Main window
root = tk.Tk()
root.title("Hotel Billing System")
root.geometry("400x300")

# Customer Name
tk.Label(root, text="Customer Name:").pack(pady=5)
entry_name = tk.Entry(root, width=30)
entry_name.pack()

# Number of Nights
tk.Label(root, text="Number of Nights:").pack(pady=5)
entry_nights = tk.Entry(root, width=10)
entry_nights.pack()

# Room Type
```


File Edit Selection View Go Run
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Debug using a terminal command or in an interactive chat.
Show automatic Python configurations
BREAKPOINTS
Raised Exceptions
Uncaught Exceptions
User Uncaught Exceptions
TASK 1.py
TASK 2.py
TASK 3.py
TASK 4.py
TASK 4.py > calculate_bill
# Room Type
tk.Label(root, text="Select Room Type:").pack(pady=5)
room_var = tk.StringVar(value="")
for room in ROOM_RATES:
tk.Radiobutton(root, text=f"{room} (₹{ROOM_RATES[room]}/night)",
variable=room_var, value=room).pack(anchor="w")
# Calculate button
tk.Button(root, text="Generate Bill", command=calculate_bill, bg="lightblue").pack(pady=15)
root.mainloop()
Ln 13, Col 40 Spaces: 4 UTF-8 Python 3.12.10 (Microsoft Store)
Search
ENG IN
23:11:31
25-08-2025

OUTPUT:


Hotel Billing System
Customer Name:
Number of Nights:
Select Room Type:
Single (₹1500/night)
Double (₹2500/night)
Suite (₹4000/night)
Generate Bill

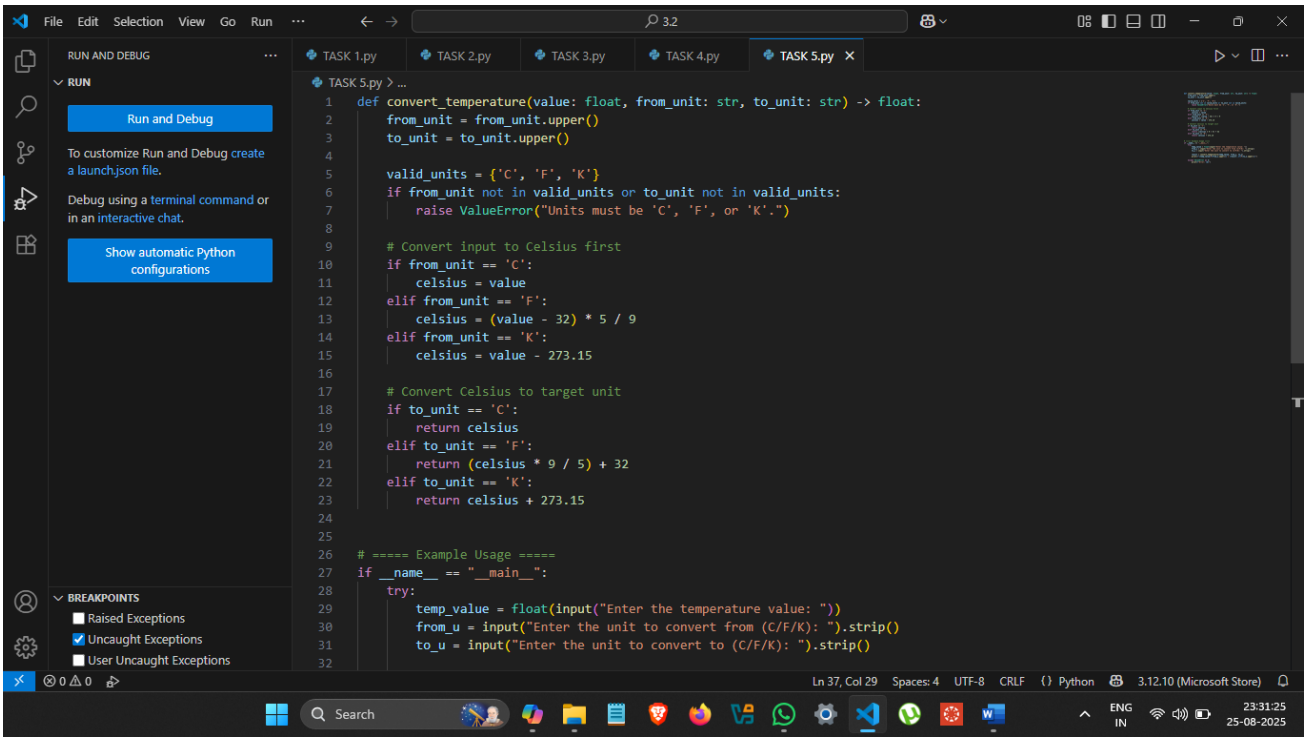


TASK 5

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

PROMPT: Give a Python Program to convert temperatures between Celsius, Fahrenheit, and Kelvin.

CODE:



```python
# ===== Example Usage =====
if __name__ == "__main__":
    try:
        temp_value = float(input("Enter the temperature value: "))
        from_u = input("Enter the unit to convert from (C/F/K): ").strip()
        to_u = input("Enter the unit to convert to (C/F/K): ").strip()

        result = convert_temperature(temp_value, from_u, to_u)
        print(f"{temp_value}°{from_u.upper()} = {result:.2f}°{to_u.upper()}")

    except ValueError as e:
        print(f"Error: {e}")
```

OUTPUT:

```
PS C:\B.TECH\3.2>  c:; cd 'c:\B.TECH\3.2'; & 'c:\Users\kamer\AppData\Local\Microsoft\WindowsApps\python3.12.exe' 'c:\Users\kamer\
.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '62401' '--' 'C:\B.TECH\3.2\TASK 5.py'
Enter the temperature value: 37
Enter the unit to convert from (C/F/K): C
Enter the unit to convert to (C/F/K): F
37.0°C = 98.60°F
PS C:\B.TECH\3.2>
```

OBSERVATION: I observed that GitHub copilot can quickly generate working code for tasks such as login systems, loan approvals, Fibonacci functions, and job applicant scoring. However, the generated code sometimes contains issues like hardcoded values, lack of encryption, or biased decision logic. This shows that AI tools are helpful for faster coding but require human review for security, fairness, and correctness. GitHub Copilot is a fascinating tool to observe—especially in how it transforms the developer experience. Here's a breakdown of key observations across its functionality, impact, and adoption