

# SR UNIVERSITY

## AI ASSIST CODING

## Lab-3.2

E.RAVALI

2503A51L18

## **TASK #1:**

### **Prompt Used:**

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

## **Code Generated:**

The screenshot shows a Visual Studio Code (VS Code) interface. The top bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help' menus. The title bar says 'AI\_CODING'. The left sidebar has icons for Explorer, Open Editors (1 unsaved), AI CODING, calculator.py, compound.py (selected), count.py, data.txt, fibonacci.py, file\_handling.py, file\_reading.py, list\_sort.py, palindrome.py, reverse\_String.py, Task1.py, and task2.py. The main editor area contains Python code for calculating compound interest. Below the editor are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (selected), and 'PORTS'. The terminal tab shows command-line output for running the script. The bottom status bar displays file paths and other system information.

```
def calculate_compound_interest(principal, rate, time, n):
    """
    Calculate compound interest.

    Parameters:
    principal : float : initial amount of money
    rate : float : annual interest rate (as decimal, e.g., 0.05 for 5%)
    time : float : time in years
    n : int : number of times interest is compounded per year

    Returns:
    float : total amount after interest
    """
    return principal * (1 + rate/n)**(n*time)

# Take input from the user
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the annual interest rate (as a decimal, e.g., 0.05): "))
time = float(input("Enter the time in years: "))
n = int(input("Enter the number of times interest is compounded per year: "))

# Calculate compound interest
amount = calculate_compound_interest(principal, rate, time, n)
print("Total amount after interest:", round(amount, 2))
```

## **Output After executing Code:**

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + × ... | [] ×

Drive/Documents/AI\_CODING/compound.py"

```
Enter the principal amount: 1000
Enter the annual interest rate (as a decimal, e.g., 0.05): 0.05
Enter the time in years: 2
Enter the number of times interest is compounded per year: 4
Total amount after interest: 1104.49
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> 
```

[powershell] Python Python

In 16. Col 27 (58 selected) Spaces: 4 UTF-8 CRLF {} Python Chat quota reached 3.7.6 Go live

## Observations:

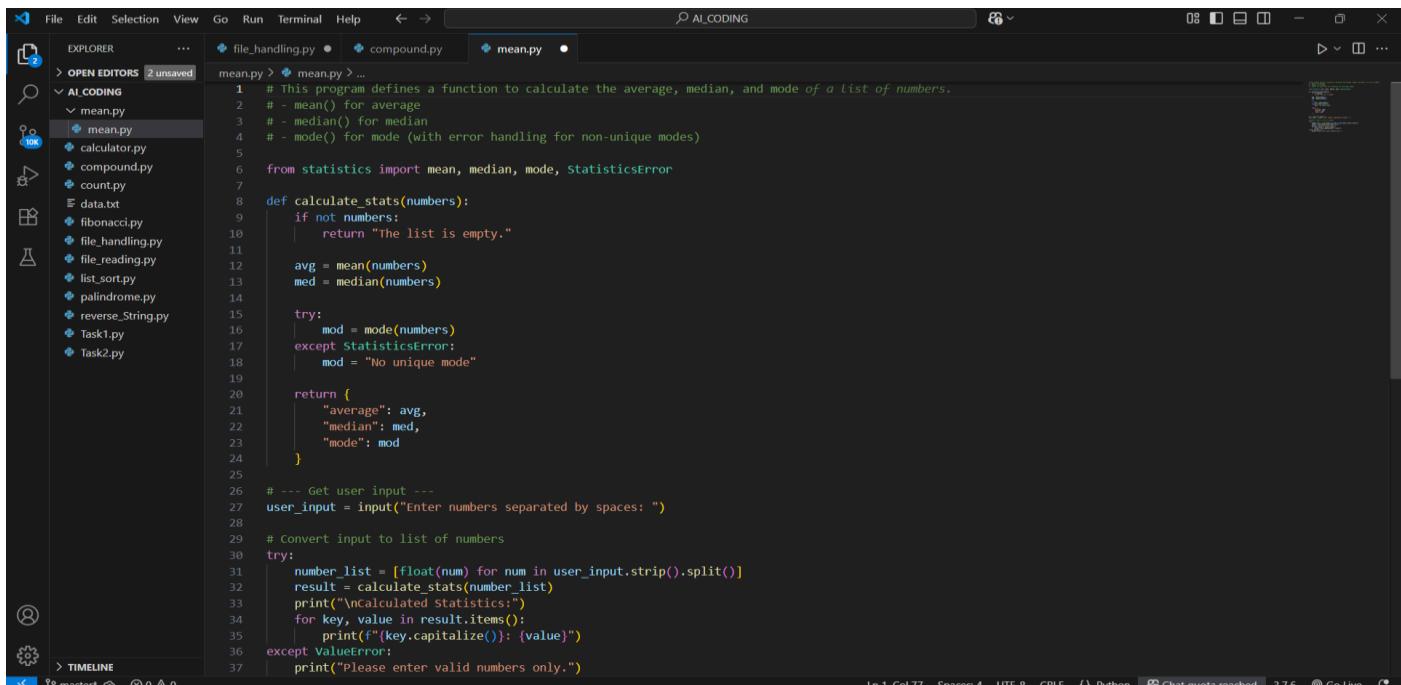
- The code correctly implements the compound interest formula using inputs (principal, rate, time, compounds per year) and returns the calculated final amount.
- The docstring explains the function with parameter details, return type, and an example, making the code clear and professional.

## TASK #2:

### Prompt Used:

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

### Code Generated:



```
# This program defines a function to calculate the average, median, and mode of a list of numbers.
# - mean() for average
# - median() for median
# - mode() for mode (with error handling for non-unique modes)

from statistics import mean, median, mode, StatisticsError

def calculate_stats(numbers):
    if not numbers:
        return "The list is empty."
    avg = mean(numbers)
    med = median(numbers)

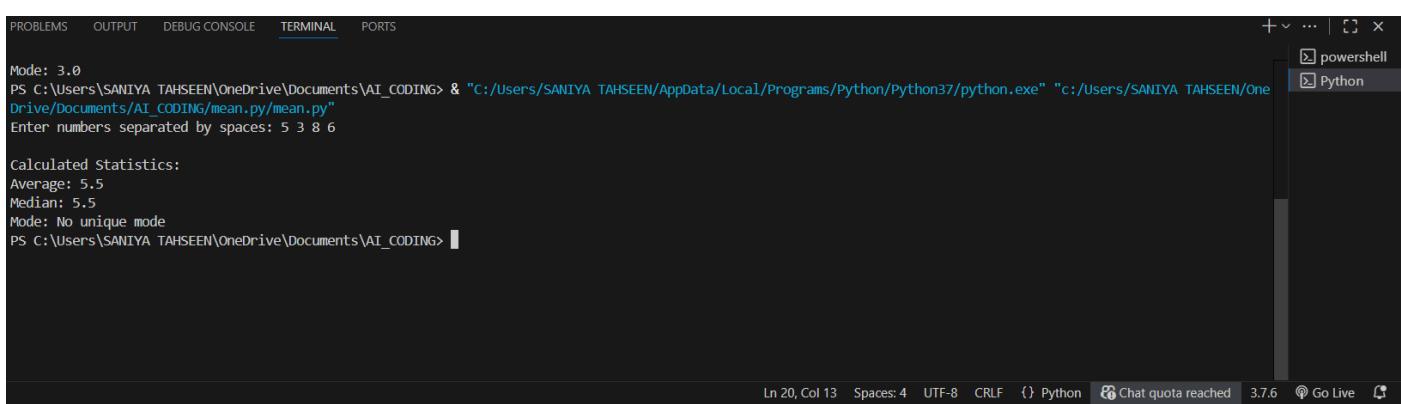
    try:
        mod = mode(numbers)
    except StatisticsError:
        mod = "No unique mode"

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

# --- Get user input ---
user_input = input("Enter numbers separated by spaces: ")

# Convert input to list of numbers
try:
    number_list = [float(num) for num in user_input.strip().split()]
    result = calculate_stats(number_list)
    print("\nCalculated statistics:")
    for key, value in result.items():
        print(f"\t{key.capitalize()}: {value}")
except ValueError:
    print("Please enter valid numbers only.")
```

### Output After executing Code:



```
Mode: 3.0
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/mean.py"
Enter numbers separated by spaces: 5 3 8 6

Calculated Statistics:
Average: 5.5
Median: 5.5
Mode: No unique mode
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

## Observations:

- The program defines a function to calculate the average, median, and mode of a list of numbers using Python's built-in statistics functions: mean(), median(), and mode().
- A try-except block handles cases where there is no unique mode, preventing errors.

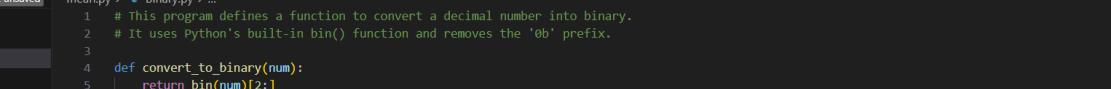
- User input is taken as a space-separated string, converted into floats, and results are displayed in a structured dictionary format.

### **TASK #3:**

## Prompt Used:

Provide multiple examples of input-output to the AI for convert\_to\_binary(num) function. Observe how AI uses few-shot prompting to generalize.

## **Code Generated:**



The screenshot shows a VS Code interface with the following details:

- File Explorer:** On the left, it lists files under the "AI\_CODING" folder. The "binary.py" file is currently selected.
- Code Editor:** The main area displays Python code for converting decimal numbers to binary. The code uses the built-in `bin()` function and handles user input.
- Status Bar:** At the bottom, there are status icons and a message: "Python: Set as Default Language".

```
1 # This program defines a function to convert a decimal number into binary.
2 # It uses Python's built-in bin() function and removes the '0b' prefix.
3
4 def convert_to_binary(num):
5     return bin(num)[2:]
6
7 # ---- Get user input ---
8 try:
9     user_input = int(input("Enter a decimal number to convert to binary: "))
10    binary_result = convert_to_binary(user_input)
11    print(f"Binary representation of {user_input} is: {binary_result}")
12 except ValueError:
13     print("Please enter a valid integer.")
14
```

## **Output After executing Code:**

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... [ ] x

PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/mean.py/binary.py"
Enter a decimal number to convert to binary: 10
Binary representation of 10 is: 1010
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/mean.py/binary.py"
Enter a decimal number to convert to binary: 135
Binary representation of 135 is: 1000011
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/mean.py/binary.py"
Enter a decimal number to convert to binary: 100
Binary representation of 100 is: 1100100
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

## **Observations:**

- `bin(num)` converts a decimal number to a binary string prefixed with '0b'.
  - Using `[2:]` slices off the '0b' to return only the binary digits.
  - The function uses a try-except block to check whether the user's input is a valid number and to handle errors if it is not.

## **TASK #4:**

### **Prompt Used:**

Create an user interface for an hotel to generate bill based on customer requirements

## **Code Generated:**

The screenshot shows the Visual Studio Code interface. The left sidebar displays a file tree with several Python files and other files like 'data.txt'. The main editor area is titled 'bill.py' and contains the following code:

```
# --- Hotel Billing System ---
menu = {
    "Burger": 120,
    "Pizza": 250,
    "Pasta": 180,
    "Coke": 50,
    "Coffee": 70
}

# Function to display menu
def display_menu():
    print("\n----- Hotel Menu -----")
    for item, price in menu.items():
        print(f"{item}: ₹{price}")
    print("-----")

# Function to calculate cost of selected item
def calculate_cost(item, quantity):
    return menu[item] * quantity

# Function to generate bill
def generate_bill(order):
    total = 0
    print("\n----- Final Bill -----")
    for item, qty in order.items():
        cost = calculate_cost(item, qty)
        print(f"[item] x {qty} = ₹{cost}")
        total += cost
    print("-----")
    print(f"Total Amount: ₹{total}")
    print("Thank you for visiting!")

# --- Main Program ---
display_menu()
```

## Output After executing Code:

The terminal window shows the command to run the script and the resulting output:

```
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/mean.py/bill.py"

----- Hotel Menu -----
Burger: ₹120
Pizza: ₹250
Pasta: ₹180
Coke: ₹50
Coffee: ₹70
-----

Enter item name (or 'done' to finish): pizza
Enter quantity for Pizza: 3

Enter item name (or 'done' to finish): coffee
Enter quantity for Coffee: 1

Enter item name (or 'done' to finish): done

----- Final Bill -----
Pizza x 3 = ₹750
Coffee x 1 = ₹70
-----
Total Amount: ₹820
Thank you for visiting!
```

## Observations:

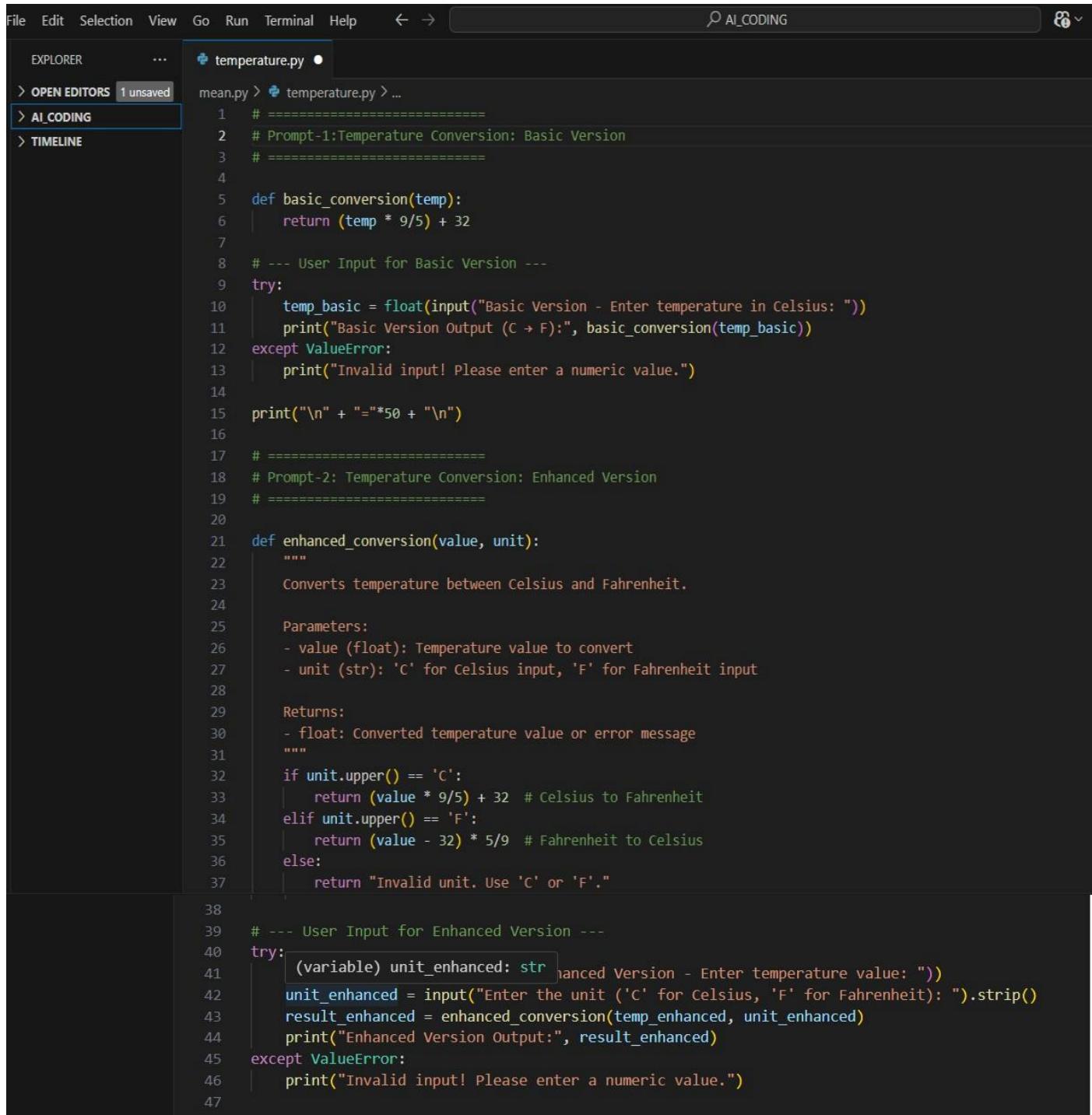
- The menu is stored as a dictionary with items as keys (capitalized) and their prices as values.
- User input is processed with `.capitalize()` so it matches the menu keys, making input case-insensitive for practical purposes.
- Separate functions are defined for repeated tasks like displaying the menu and generating the bill, ensuring code reusability.

## TASK #5:

## Prompt Used:

Analyzing Prompt Specificity: Improving Temperature Conversion Function with Clear Instructions

## Code Generated:



The screenshot shows a code editor interface with a dark theme. The top bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help' menus. A search bar is positioned above the main workspace. The left sidebar features 'EXPLORER', 'OPEN EDITORS' (with one unsaved file), 'AI CODING' (selected), and 'TIMELINE'. The main workspace displays the following Python script:

```
# =====#
# Prompt-1:Temperature Conversion: Basic Version
# =====#
def basic_conversion(temp):
    return (temp * 9/5) + 32
# --- User Input for Basic Version ---
try:
    temp_basic = float(input("Basic Version - Enter temperature in Celsius: "))
    print("Basic Version Output (C → F):", basic_conversion(temp_basic))
except ValueError:
    print("Invalid input! Please enter a numeric value.")

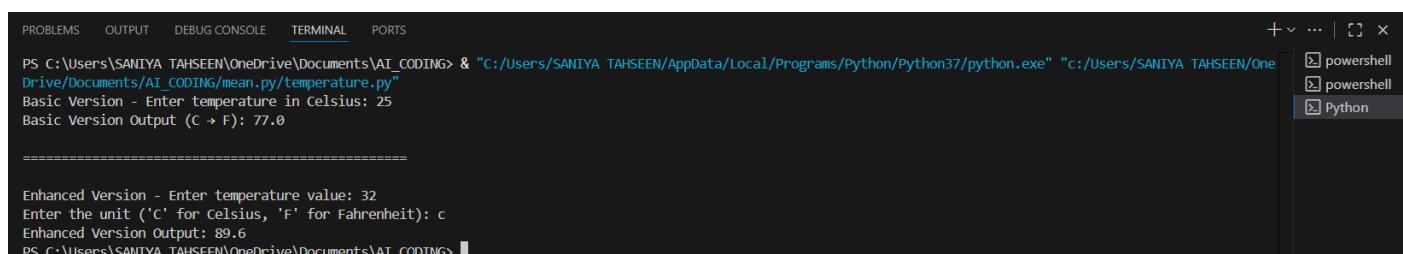
print("\n" + "="*50 + "\n")
# =====#
# Prompt-2: Temperature Conversion: Enhanced Version
# =====#
def enhanced_conversion(value, unit):
    """
    Converts temperature between Celsius and Fahrenheit.

    Parameters:
    - value (float): Temperature value to convert
    - unit (str): 'C' for Celsius input, 'F' for Fahrenheit input

    Returns:
    - float: Converted temperature value or error message
    """
    if unit.upper() == 'C':
        return (value * 9/5) + 32 # Celsius to Fahrenheit
    elif unit.upper() == 'F':
        return (value - 32) * 5/9 # Fahrenheit to Celsius
    else:
        return "Invalid unit. Use 'C' or 'F'."

# --- User Input for Enhanced Version ---
try:
    (variable) unit_enhanced: str = input("Enhanced Version - Enter temperature value: ")
    unit_enhanced = input("Enter the unit ('C' for Celsius, 'F' for Fahrenheit): ").strip()
    result_enhanced = enhanced_conversion(temp_enhanced, unit_enhanced)
    print("Enhanced Version Output:", result_enhanced)
except ValueError:
    print("Invalid input! Please enter a numeric value.")
```

## Output After executing Code:



The screenshot shows a terminal window with the following output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ×
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING> & "C:/Users/SANIYA TAHSEEN/AppData/Local/Programs/Python/Python37/python.exe" "c:/Users/SANIYA TAHSEEN/OneDrive/Documents/AI_CODING/mean.py/temperature.py"
Basic Version - Enter temperature in Celsius: 25
Basic Version Output (C → F): 77.0
=====
Enhanced Version - Enter temperature value: 32
Enter the unit ('C' for Celsius, 'F' for Fahrenheit): c
Enhanced Version Output: 89.6
PS C:\Users\SANIYA TAHSEEN\OneDrive\Documents\AI_CODING>
```

## **Observations:**

### **Prompt-1: Temperature Conversion: Basic Version**

- Handles only Celsius → Fahrenheit conversion
- Code is simple, minimal, and easy to follow
- No error handling

### **Prompt-2: Temperature Conversion: Enhanced Version**

- Converts both Celsius ↔ Fahrenheit based on user input unit.
- Includes docstring, comments, and validation for invalid inputs.
- Robust, readable, and user-friendly.