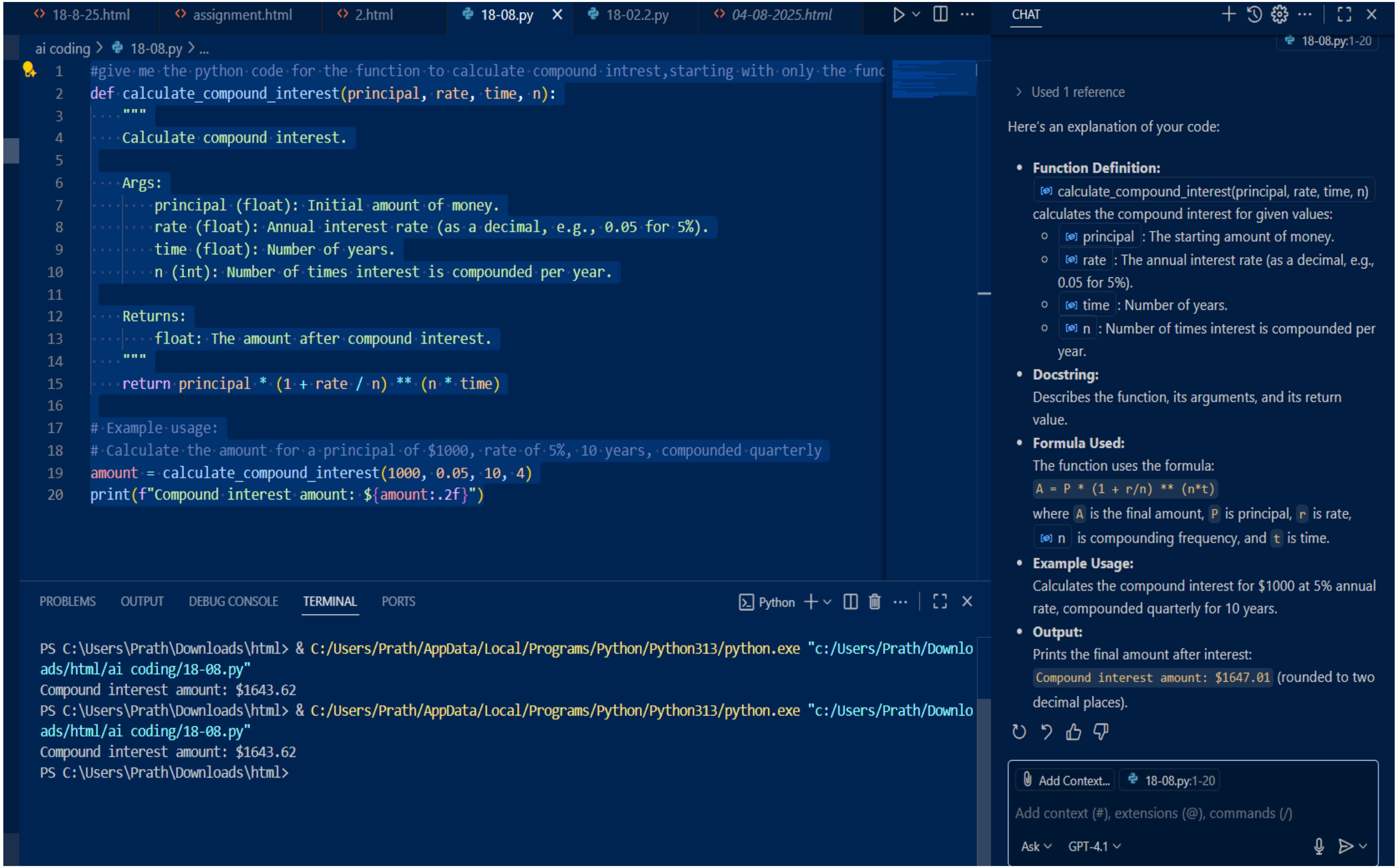
AI ASSISTED CODING ASSAIGNMENT3

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



```python
#give me the python code for the function to calculate compound intrest,starting with only the func
def calculate_compound_interest(principal, rate, time, n):
    """
    Calculate compound interest.

    Args:
        principal (float): Initial amount of money.
        rate (float): Annual interest rate (as a decimal, e.g., 0.05 for 5%).
        time (float): Number of years.
        n (int): Number of times interest is compounded per year.

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

# Example usage:
# Calculate the amount for a principal of $1000, rate of 5%, 10 years, compounded quarterly
amount = calculate_compound_interest(1000, 0.05, 10, 4)
print(f"Compound interest amount: ${amount:.2f}")
```

```
PS C:\Users\Prath\Downloads\html> & C:/Users/Prath/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Prath/Downlo
ads/html/ai coding/18-08.py"
Compound interest amount: $1643.62
PS C:\Users\Prath\Downloads\html> & C:/Users/Prath/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Prath/Downlo
ads/html/ai coding/18-08.py"
Compound interest amount: $1643.62
PS C:\Users\Prath\Downloads\html>
```

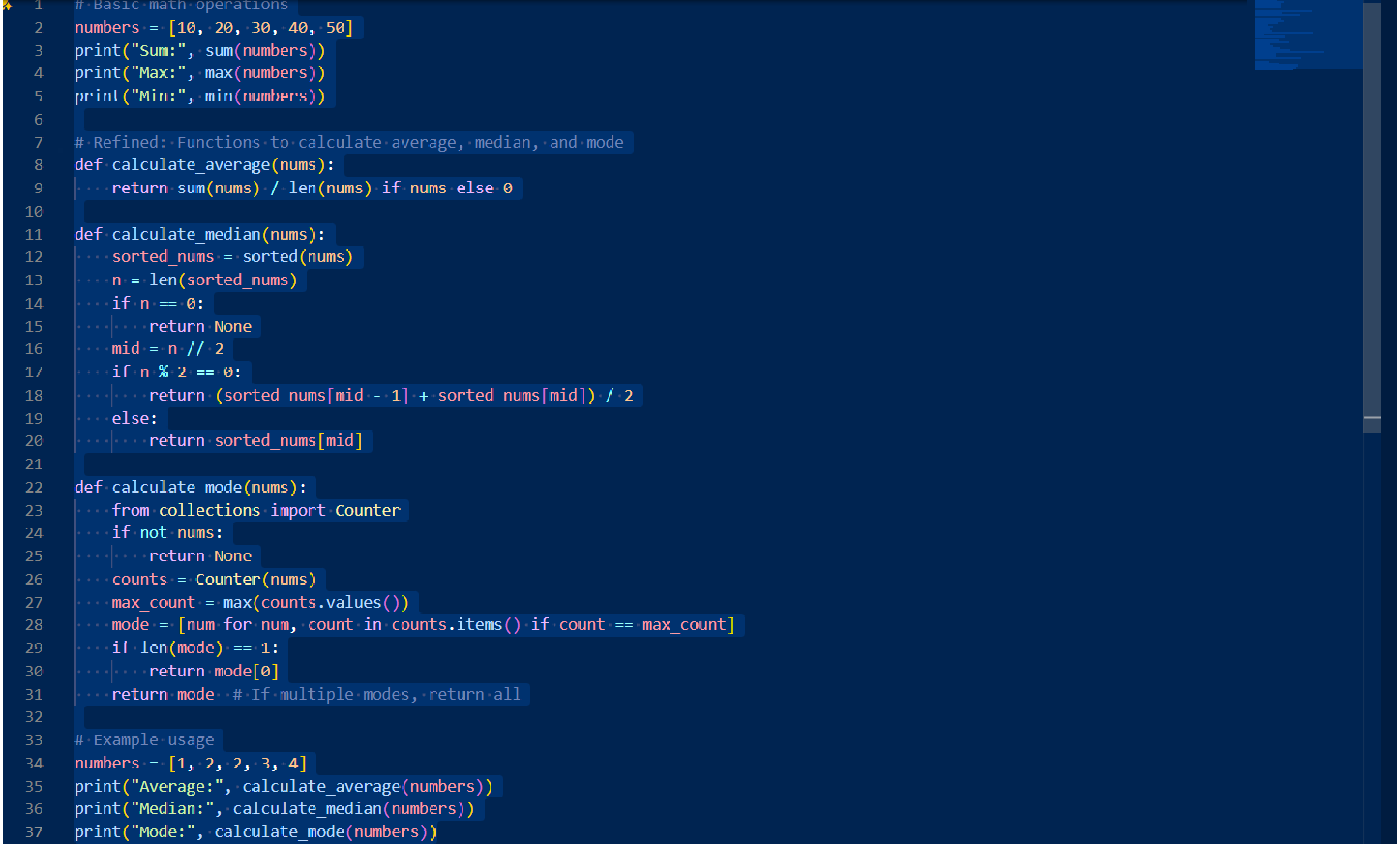
> Used 1 reference

Here's an explanation of your code:

- **Function Definition:**
  `calculate_compound_interest(principal, rate, time, n)` calculates the compound interest for given values:
  - `principal` : The starting amount of money.
  - `rate` : The annual interest rate (as a decimal, e.g., 0.05 for 5%).
  - `time` : Number of years.
  - `n` : Number of times interest is compounded per year.
- **Docstring:**
  Describes the function, its arguments, and its return value.
- **Formula Used:**
  The function uses the formula:
  `A = P * (1 + r/n) ** (n*t)`
  where `A` is the final amount, `P` is principal, `r` is rate, `n` is compounding frequency, and `t` is time.
- **Example Usage:**
  Calculates the compound interest for $1000 at 5% annual rate, compounded quarterly for 10 years.
- **Output:**
  Prints the final amount after interest:
  `Compound interest amount: $1647.01` (rounded to two decimal places).

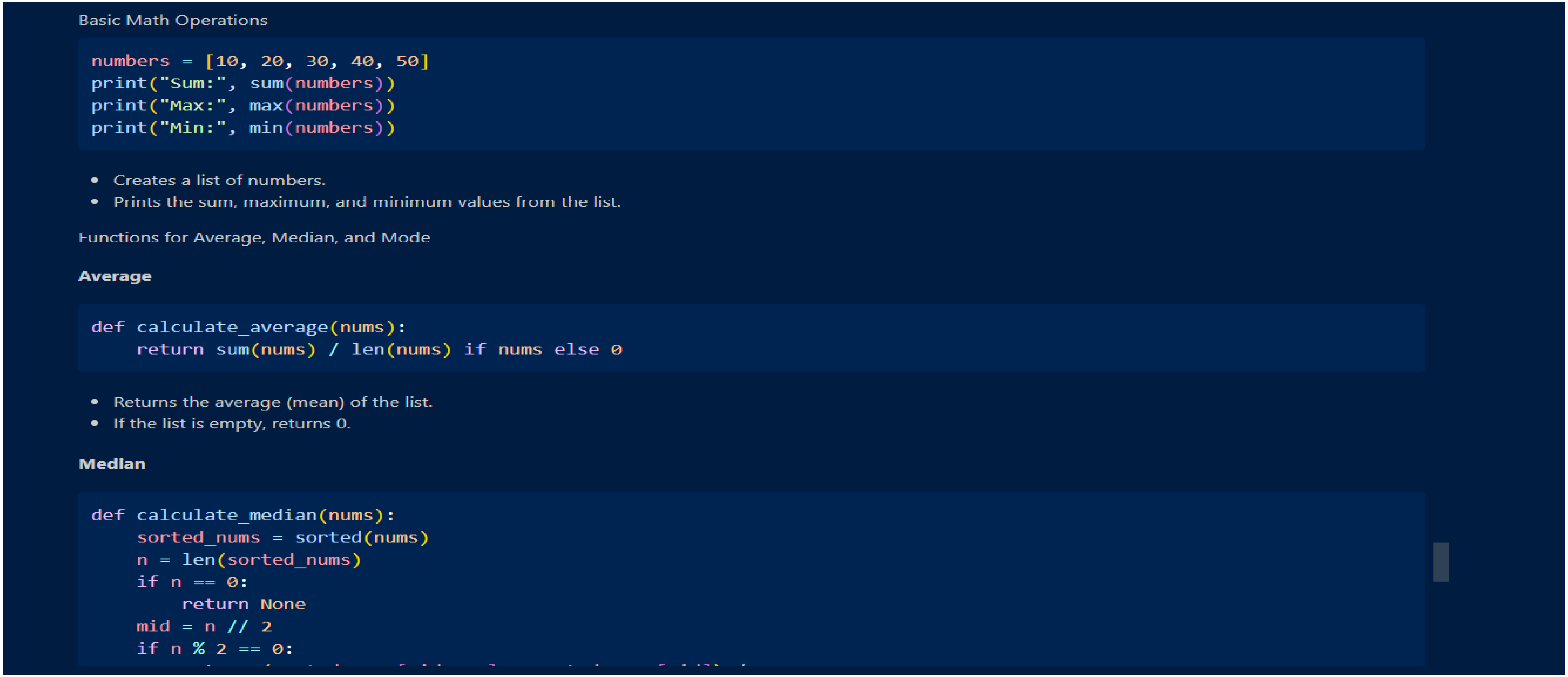
Task Description#2

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



```python
# Basic math operations
numbers = [10, 20, 30, 40, 50]
print("Sum:", sum(numbers))
print("Max:", max(numbers))
print("Min:", min(numbers))

# Refined: Functions to calculate average, median, and mode
def calculate_average(nums):
    return sum(nums) / len(nums) if nums else 0

def calculate_median(nums):
    sorted_nums = sorted(nums)
    n = len(sorted_nums)
    if n == 0:
        return None
    mid = n // 2
    if n % 2 == 0:
        return (sorted_nums[mid - 1] + sorted_nums[mid]) / 2
    else:
        return sorted_nums[mid]

def calculate_mode(nums):
    from collections import Counter
    if not nums:
        return None
    counts = Counter(nums)
    max_count = max(counts.values())
    mode = [num for num, count in counts.items() if count == max_count]
    if len(mode) == 1:
        return mode[0]
    return mode  # If multiple modes, return all

# Example usage
numbers = [1, 2, 2, 3, 4]
print("Average:", calculate_average(numbers))
print("Median:", calculate_median(numbers))
print("Mode:", calculate_mode(numbers))
```

EXPLAINATION



Basic Math Operations

```
numbers = [10, 20, 30, 40, 50]
print("Sum:", sum(numbers))
print("Max:", max(numbers))
print("Min:", min(numbers))
```

- Creates a list of numbers.
- Prints the sum, maximum, and minimum values from the list.

Functions for Average, Median, and Mode

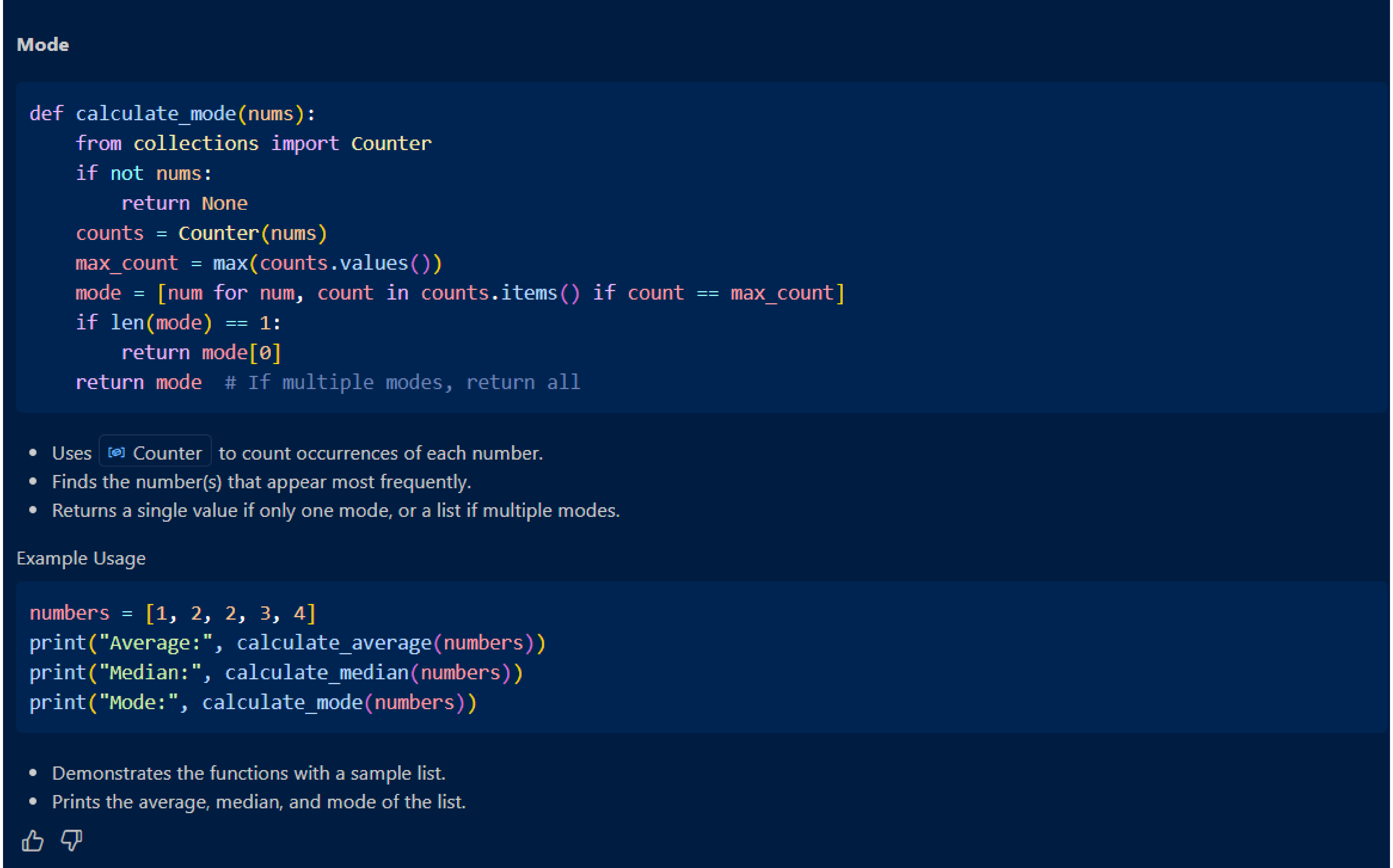
Average

```
def calculate_average(nums):
    return sum(nums) / len(nums) if nums else 0
```

- Returns the average (mean) of the list.
- If the list is empty, returns 0.

Median

```
def calculate_median(nums):
    sorted_nums = sorted(nums)
    n = len(sorted_nums)
    if n == 0:
        return None
    mid = n // 2
    if n % 2 == 0:
```



Mode

```
def calculate_mode(nums):
    from collections import Counter
    if not nums:
        return None
    counts = Counter(nums)
    max_count = max(counts.values())
    mode = [num for num, count in counts.items() if count == max_count]
    if len(mode) == 1:
        return mode[0]
    return mode  # If multiple modes, return all
```

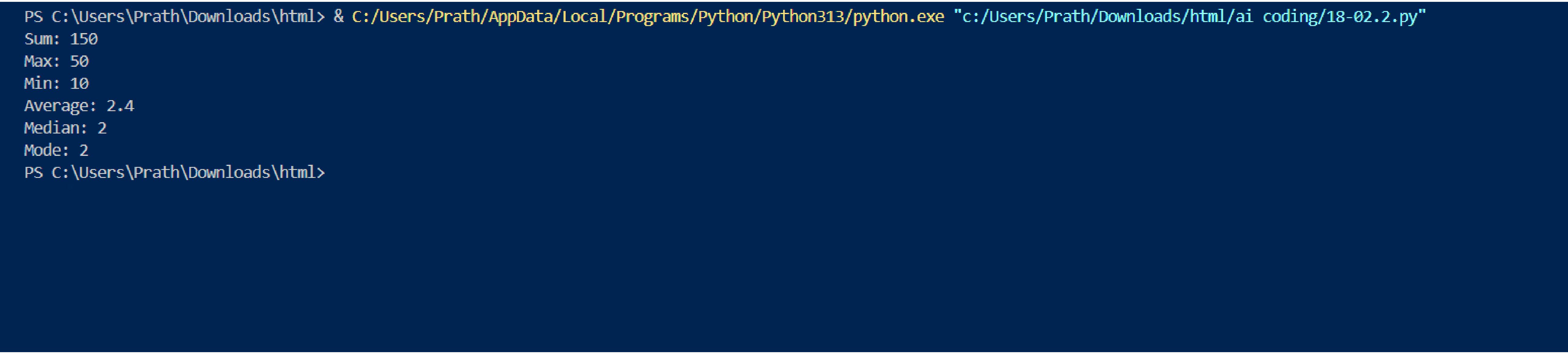
- Uses `Counter` to count occurrences of each number.
- Finds the number(s) that appear most frequently.
- Returns a single value if only one mode, or a list if multiple modes.

Example Usage

```
numbers = [1, 2, 2, 3, 4]
print("Average:", calculate_average(numbers))
print("Median:", calculate_median(numbers))
print("Mode:", calculate_mode(numbers))
```

- Demonstrates the functions with a sample list.
- Prints the average, median, and mode of the list.

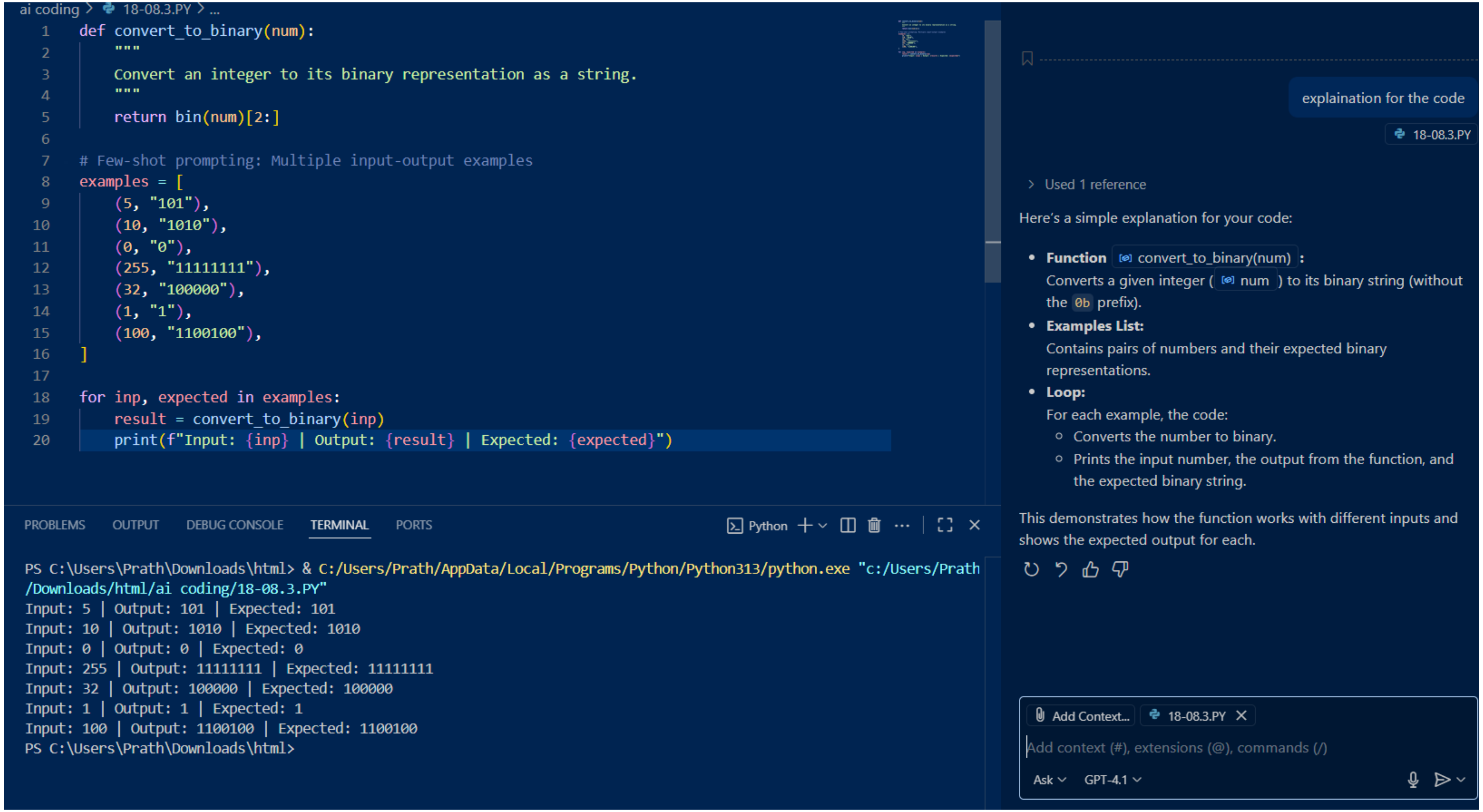
OUTPUT



```
PS C:\Users\Prath\Downloads\html> & C:/Users/Prath/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Prath/Downloads/html/ai coding/18-02.2.py"
Sum: 150
Max: 50
Min: 10
Average: 2.4
Median: 2
Mode: 2
PS C:\Users\Prath\Downloads\html>
```

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.



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

# Few-shot prompting: Multiple input-output examples
examples = [
    (5, "101"),
    (10, "1010"),
    (0, "0"),
    (255, "11111111"),
    (32, "100000"),
    (1, "1"),
    (100, "1100100"),
]

for inp, expected in examples:
    result = convert_to_binary(inp)
    print(f"Input: {inp} | Output: {result} | Expected: {expected}")
```

```
PS C:\Users\Prath\Downloads\html> & C:/Users/Prath/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Prath
/Downloads/html/ai coding/18-08.3.PY"
Input: 5 | Output: 101 | Expected: 101
Input: 10 | Output: 1010 | Expected: 1010
Input: 0 | Output: 0 | Expected: 0
Input: 255 | Output: 11111111 | Expected: 11111111
Input: 32 | Output: 100000 | Expected: 100000
Input: 1 | Output: 1 | Expected: 1
Input: 100 | Output: 1100100 | Expected: 1100100
PS C:\Users\Prath\Downloads\html>
```

explaination for the code

18-08.3.PY

Used 1 reference

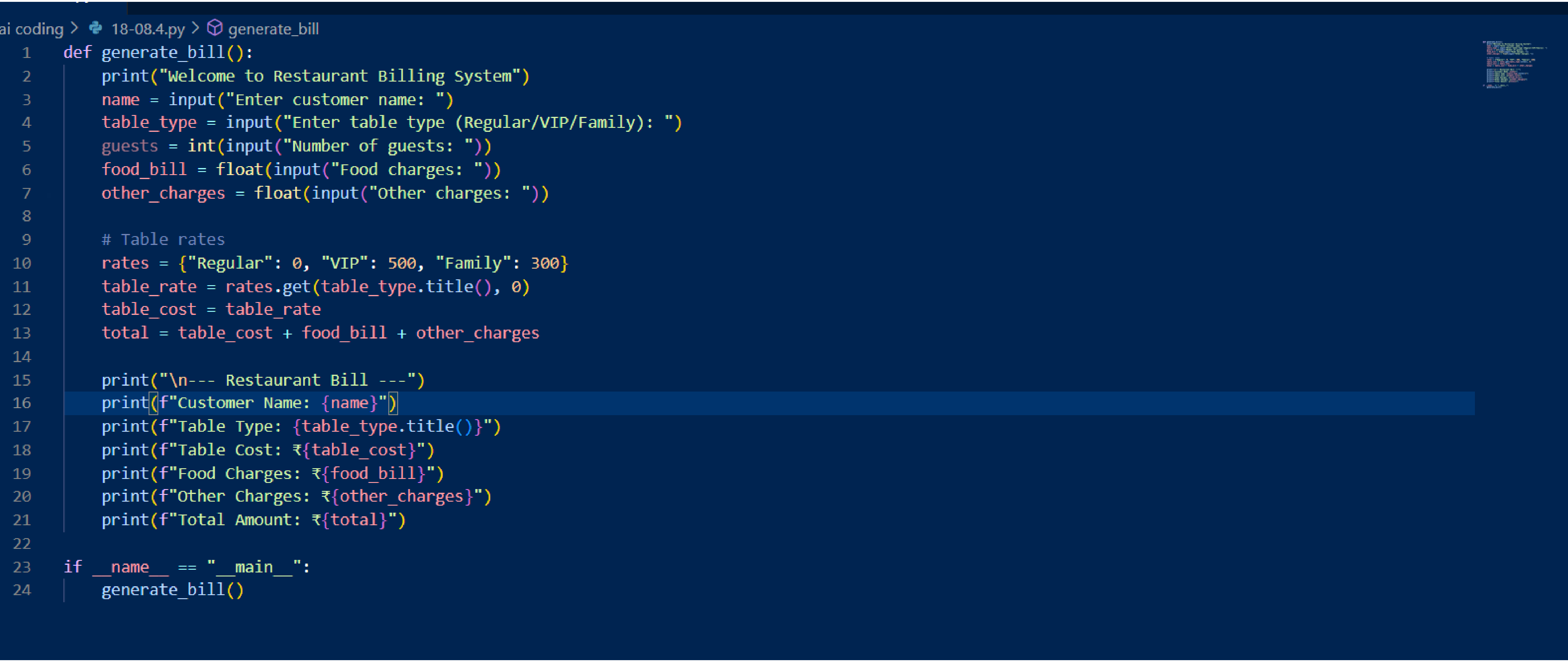
Here's a simple explanation for your code:

- **Function** `convert_to_binary(num)`**:**
  Converts a given integer (`num`) to its binary string (without the `0b` prefix).
- **Examples List:**
  Contains pairs of numbers and their expected binary representations.
- **Loop:**
  For each example, the code:
  - Converts the number to binary.
  - Prints the input number, the output from the function, and the expected binary string.

This demonstrates how the function works with different inputs and shows the expected output for each.

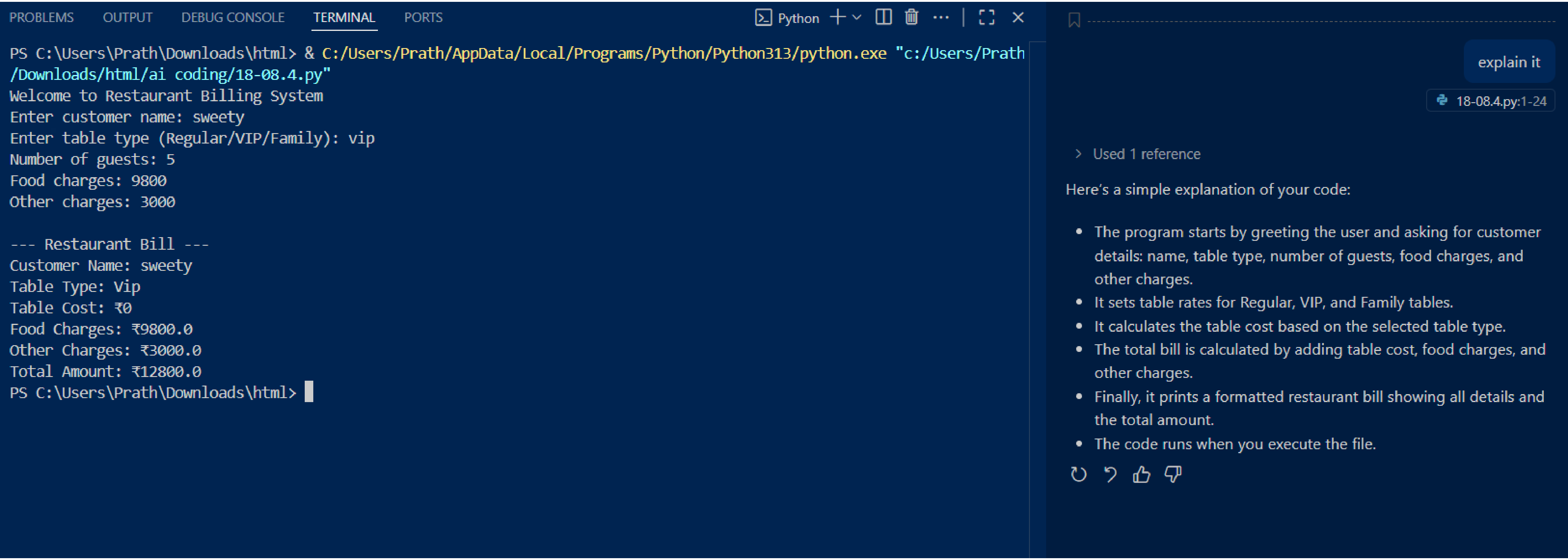
Task Description#4

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



```python
def generate_bill():
    print("Welcome to Restaurant Billing System")
    name = input("Enter customer name: ")
    table_type = input("Enter table type (Regular/VIP/Family): ")
    guests = int(input("Number of guests: "))
    food_bill = float(input("Food charges: "))
    other_charges = float(input("Other charges: "))

    # Table rates
    rates = {"Regular": 0, "VIP": 500, "Family": 300}
    table_rate = rates.get(table_type.title(), 0)
    table_cost = table_rate
    total = table_cost + food_bill + other_charges

    print("\n--- Restaurant Bill ---")
    print(f"Customer Name: {name}")
    print(f"Table Type: {table_type.title()}")
    print(f"Table Cost: ₹{table_cost}")
    print(f"Food Charges: ₹{food_bill}")
    print(f"Other Charges: ₹{other_charges}")
    print(f"Total Amount: ₹{total}")

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

Explaination and output



```
PS C:\Users\Prath\Downloads\html> & C:/Users/Prath/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Prath
/Downloads/html/ai coding/18-08.4.py"
Welcome to Restaurant Billing System
Enter customer name: sweety
Enter table type (Regular/VIP/Family): vip
Number of guests: 5
Food charges: 9800
Other charges: 3000

--- Restaurant Bill ---
Customer Name: sweety
Table Type: Vip
Table Cost: ₹0
Food Charges: ₹9800.0
Other Charges: ₹3000.0
Total Amount: ₹12800.0
PS C:\Users\Prath\Downloads\html> 
```

explain it

18-08.4.py:1-24

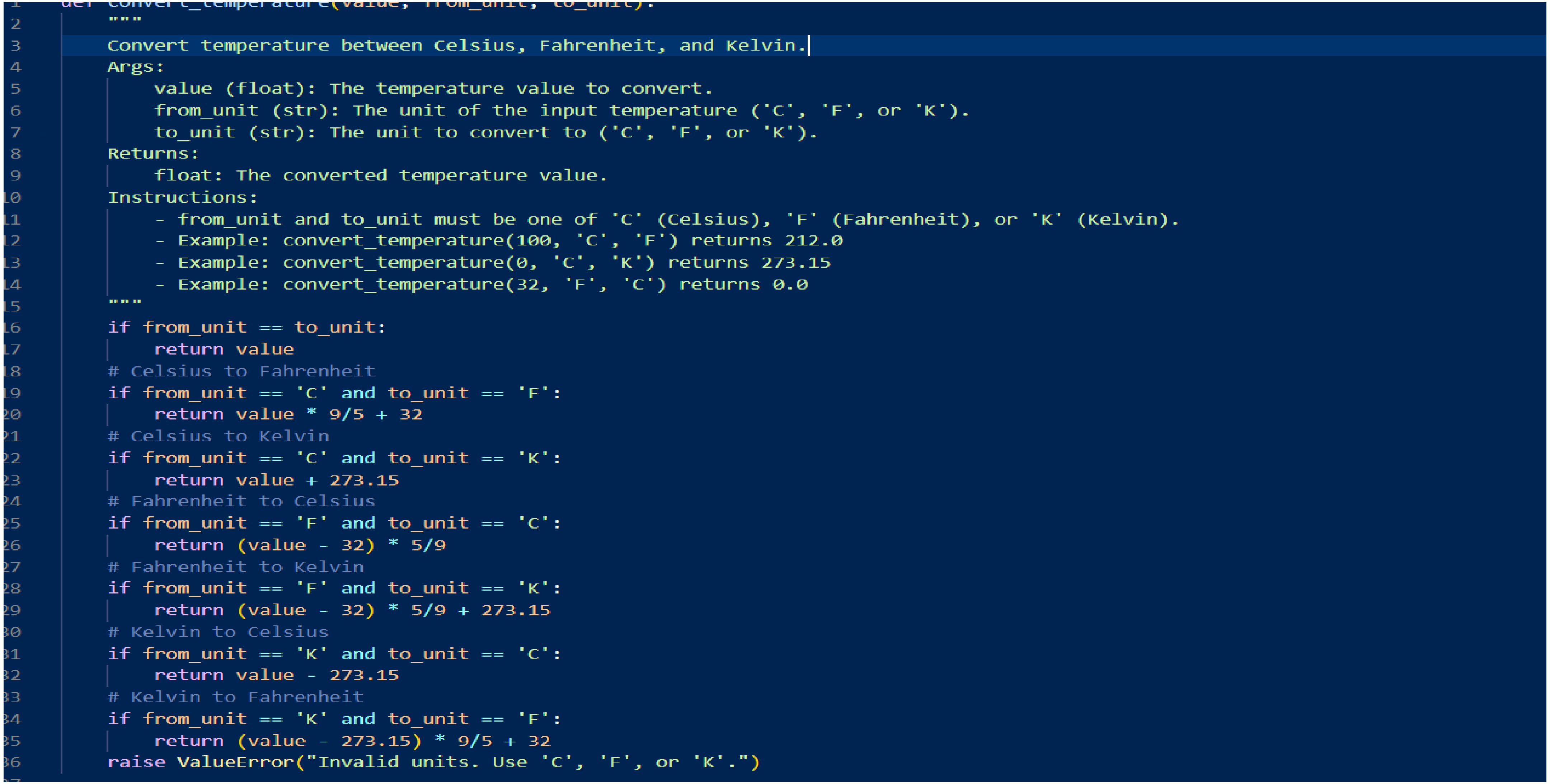
Used 1 reference

Here’s a simple explanation of your code:

- The program starts by greeting the user and asking for customer details: name, table type, number of guests, food charges, and other charges.
- It sets table rates for Regular, VIP, and Family tables.
- It calculates the table cost based on the selected table type.
- The total bill is calculated by adding table cost, food charges, and other charges.
- Finally, it prints a formatted restaurant bill showing all details and the total amount.
- The code runs when you execute the file.

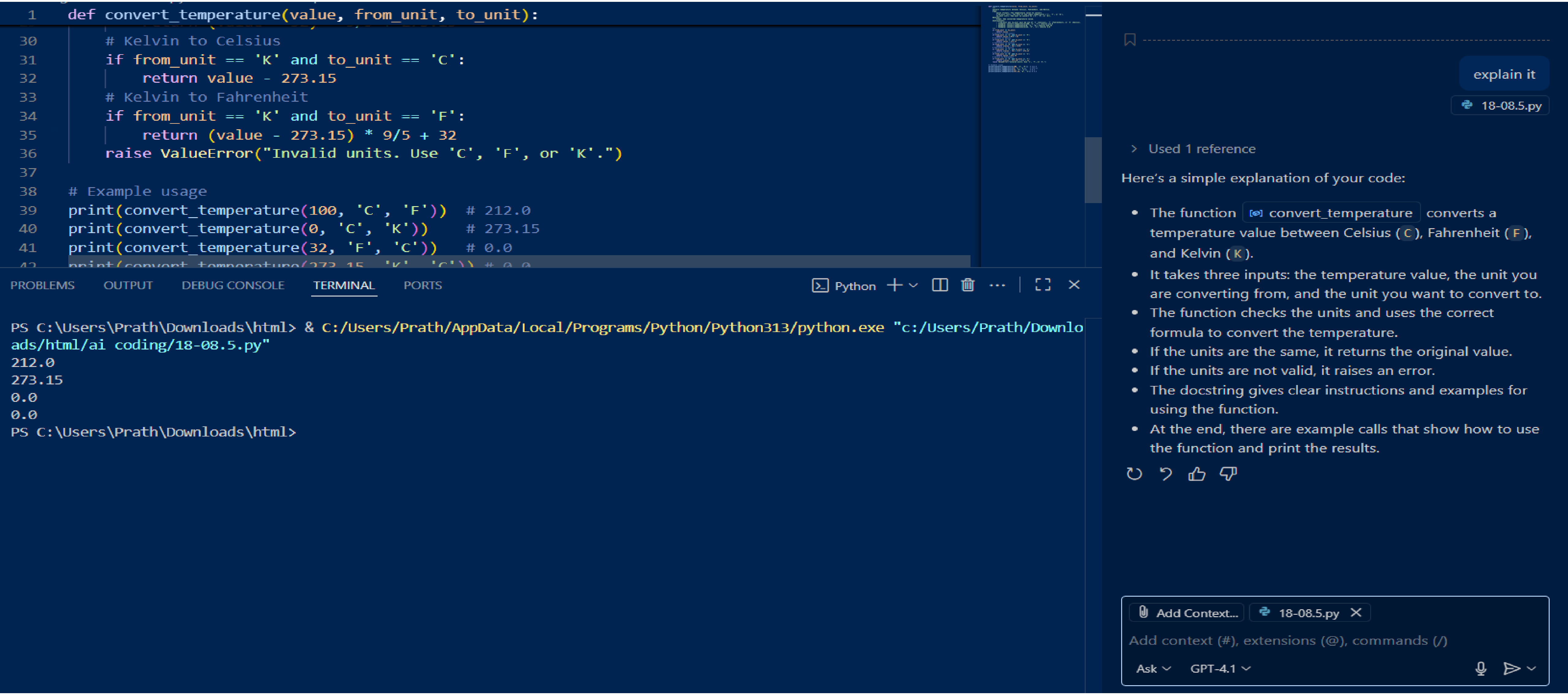
Task Description#5

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



```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', or 'K').
        to_unit (str): The unit to convert to ('C', 'F', or 'K').
    Returns:
        float: The converted temperature value.
    Instructions:
        - from_unit and to_unit must be one of 'C' (Celsius), 'F' (Fahrenheit), or 'K' (Kelvin).
        - Example: convert_temperature(100, 'C', 'F') returns 212.0
        - Example: convert_temperature(0, 'C', 'K') returns 273.15
        - Example: convert_temperature(32, 'F', 'C') returns 0.0
    """
    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 units. Use 'C', 'F', or 'K'.")
```

Explanation and result



```python
def convert_temperature(value, from_unit, to_unit):
    # 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 units. Use 'C', 'F', or 'K'.")

# Example usage
print(convert_temperature(100, 'C', 'F'))  # 212.0
print(convert_temperature(0, 'C', 'K'))    # 273.15
print(convert_temperature(32, 'F', 'C'))   # 0.0
```

```
PS C:\Users\Prath\Downloads\html> & C:/Users/Prath/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/Prath/Downloads/html/ai coding/18-08.5.py"
212.0
273.15
0.0
0.0
PS C:\Users\Prath\Downloads\html>
```

explain it

18-08.5.py

Used 1 reference

Here's a simple explanation of your code:

- The function `convert_temperature` converts a temperature value between Celsius (`C`), Fahrenheit (`F`), and Kelvin (`K`).
- It takes three inputs: the temperature value, the unit you are converting from, and the unit you want to convert to.
- The function checks the units and uses the correct formula to convert the temperature.
- If the units are the same, it returns the original value.
- If the units are not valid, it raises an error.
- The docstring gives clear instructions and examples for using the function.
- At the end, there are example calls that show how to use the function and print the results.