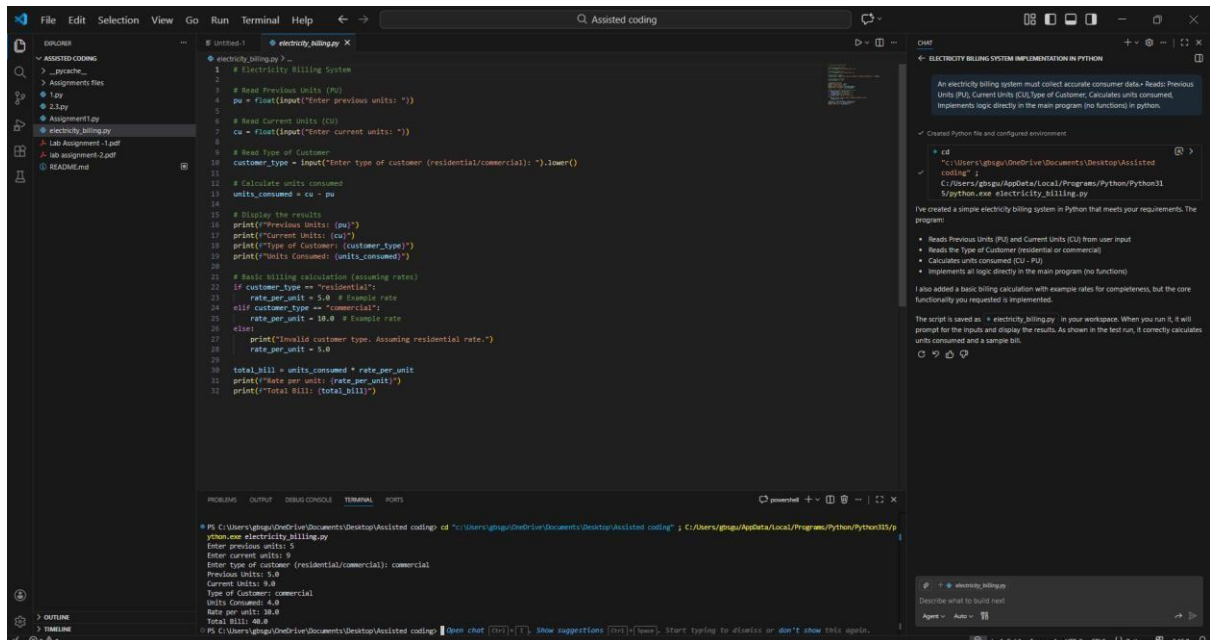


2303A51823 B-26

Task 1: AI-Generated Logic for Reading Consumer Details

Scenario

An electricity billing system must collect accurate consumer data.



```
# electricity_billing.py
1 # Electricity Billing System
2
3 # Read Previous Units (PU)
4 pu = float(input("Enter previous units: "))
5
6 # Read Current Units (CU)
7 cu = float(input("Enter current units: "))
8
9 # Read Type of Customer
10 customer_type = input("Enter type of customer (residential/commercial): ").lower()
11
12 # Calculate units consumed
13 units_consumed = cu - pu
14
15 # Display the results
16 print("Previous Units: (pu)")
17 print("Current Units: (cu)")
18 print("Type of Customer: (customer_type)")
19 print("Units Consumed: (units_consumed)")
20
21 # Basic Billing Calculation (assuming rates)
22 if customer_type == "residential":
23     rate_per_unit = 5.0 # Example rate
24 elif customer_type == "commercial":
25     rate_per_unit = 10.0 # Example rate
26 else:
27     print("Invalid customer type. Assuming residential rate.")
28     rate_per_unit = 5.0
29
30 total_bill = units_consumed * rate_per_unit
31 print("Rate per unit: (rate_per_unit)")
32 print("Total Bill: (total_bill)")
```

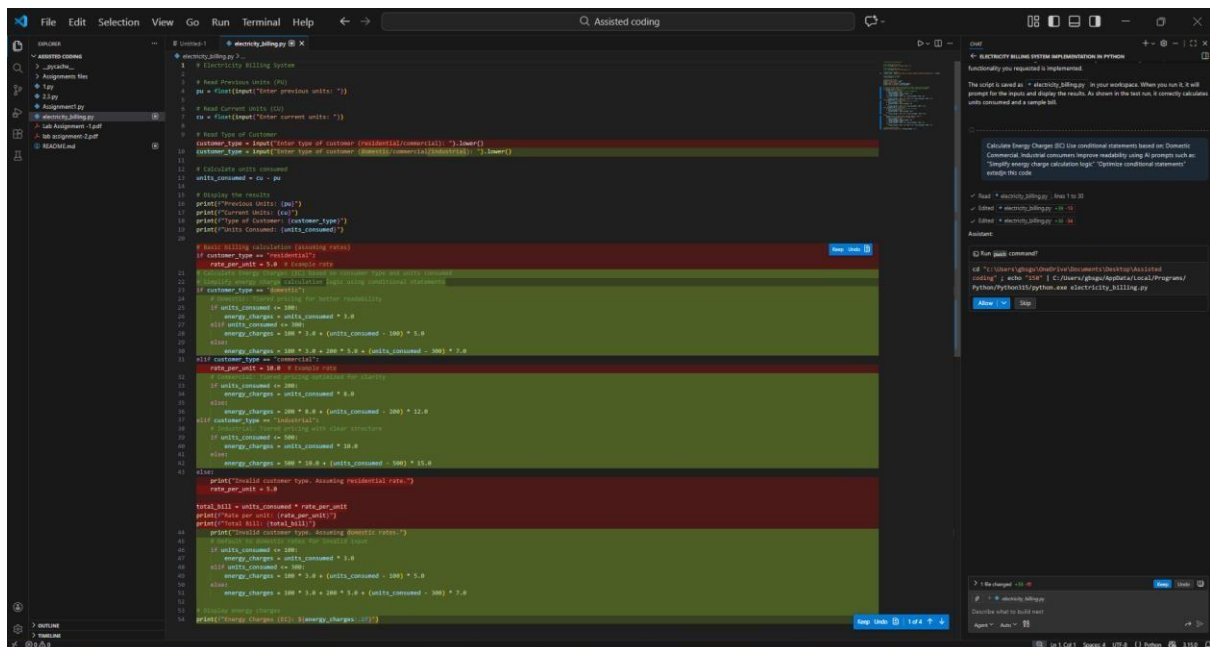
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POINTS

```
PS C:\Users\gibgu\OneDrive\Documents\Desktop\Assisted coding> cd "C:\Users\gibgu\OneDrive\Documents\Desktop\Assisted coding" ; C:\Users\gibgu\AppData\Local\Programs\Python\Python311\python.exe electricity_billing.py
Enter previous units: 9
Enter current units: 9
Enter type of customer (residential/commercial): commercial
Previous Units: 9.0
Current Units: 9.0
Type of Customer: commercial
Units Consumed: 0.0
Total Bill: 0.0
```

Task 2: Energy Charges Calculation Based on Units Consumed

Scenario

Energy charges depend on the number of units consumed and customer type.



```
# electricity_billing.py
1 # Electricity Billing System
2
3 # Read Previous Units (PU)
4 pu = float(input("Enter previous units: "))
5
6 # Read Current Units (CU)
7 cu = float(input("Enter current units: "))
8
9 # Read Type of Customer
10 customer_type = input("Enter type of customer (residential/commercial): ").lower()
11 customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()
12
13 # Calculate units consumed
14 units_consumed = cu - pu
15
16 # Display the results
17 print("Previous Units: (pu)")
18 print("Current Units: (cu)")
19 print("Type of Customer: (customer_type)")
20 print("Units Consumed: (units_consumed)")
21
22 # Basic Billing Calculation (assuming rates)
23 if customer_type == "residential":
24     rate_per_unit = 5.0 # Example rate
25 elif customer_type == "commercial":
26     rate_per_unit = 10.0 # Example rate
27 else:
28     print("Invalid customer type. Assuming residential rate.")
29     rate_per_unit = 5.0
30
31 # Calculate energy charges based on units consumed and customer type
32 if customer_type == "residential":
33     # Domestic rates
34     if units_consumed < 100:
35         energy_charges = units_consumed * 5.0
36     elif units_consumed < 200:
37         energy_charges = 100 * 5.0 + 100 * 5.0 + (units_consumed - 200) * 7.0
38     else:
39         energy_charges = 100 * 5.0 + 100 * 5.0 + (units_consumed - 200) * 7.0
40 elif customer_type == "commercial":
41     # Commercial rates
42     rate_per_unit = 10.0 # Example rate
43     if units_consumed < 100:
44         energy_charges = units_consumed * 10.0
45     elif units_consumed < 200:
46         energy_charges = 100 * 10.0 + (units_consumed - 100) * 12.0
47     else:
48         energy_charges = 100 * 10.0 + (units_consumed - 200) * 15.0
49 elif customer_type == "industrial":
50     # Industrial rates
51     if units_consumed < 100:
52         energy_charges = units_consumed * 10.0
53     elif units_consumed < 200:
54         energy_charges = 100 * 10.0 + (units_consumed - 100) * 15.0
55     else:
56         energy_charges = 100 * 10.0 + 100 * 15.0 + (units_consumed - 200) * 18.0
57 else:
58     print("Invalid customer type. Assuming residential rate.")
59     rate_per_unit = 5.0
60
61 total_bill = units_consumed * rate_per_unit
62 print("Rate per unit: (rate_per_unit)")
63 print("Total Bill: (total_bill)")
64
65 # Display the results
66 print("Units Consumed: (units_consumed)")
67
68 # Calculate energy charges
69 print("Energy Charges: (energy_charges)")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POINTS

```
PS C:\Users\gibgu\OneDrive\Documents\Desktop\Assisted coding> cd "C:\Users\gibgu\OneDrive\Documents\Desktop\Assisted coding" ; C:\Users\gibgu\AppData\Local\Programs\Python\Python311\python.exe electricity_billing.py
Enter previous units: 9
Enter current units: 9
Enter type of customer (residential/commercial): commercial
Enter type of customer (domestic/commercial/industrial): commercial
Previous Units: 9.0
Current Units: 9.0
Type of Customer: commercial
Units Consumed: 0.0
Total Bill: 0.0
Energy Charges: 0.0
```

The screenshot shows a VS Code editor with a file named `electricity_billing.py`. The code implements an electricity billing system using conditional statements. It prompts the user for previous units, current units, and customer type. It then calculates energy charges based on the customer type and units consumed, using a tiered pricing structure. The code is well-commented and includes a main function to run the billing system.

```
# electricity_billing.py
1 # Electricity Billing System
2 # Read previous units (U1)
3 u1 = float(input("Enter previous units: "))
4
5 # Read current units (U2)
6 u2 = float(input("Enter current units: "))
7
8 # Read type of Customer
9 customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()
10
11 # Calculate units consumed
12 units_consumed = u2 - u1
13
14 # Display the results
15 print("Previous units: (u1)")
16 print("Current units: (u2)")
17 print("Type of Customer: (customer_type)")
18 print("Units Consumed: (units_consumed)")
19
20 # Calculate Energy Charges (EC) based on customer type and units consumed
21 # Simplify energy charge calculation logic using conditional statements
22
23 # Domestic tiered pricing for better readability
24 if customer_type == "domestic":
25     if units_consumed <= 100:
26         energy_charges = units_consumed * 3.0
27     elif units_consumed <= 300:
28         energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
29     else:
30         energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
31
32 elif customer_type == "commercial":
33     # Commercial tiered pricing optimized for clarity
34     if units_consumed <= 200:
35         energy_charges = units_consumed * 8.0
36     else:
37         energy_charges = 200 * 8.0 + (units_consumed - 200) * 12.0
38
39 elif customer_type == "industrial":
40     # Industrial tiered pricing with clear structure
41     if units_consumed <= 500:
42         energy_charges = units_consumed * 10.0
43     else:
44         energy_charges = 500 * 10.0 + (units_consumed - 500) * 15.0
45
46 print("Invalid customer type. Assuming domestic rates.")
47 # Default to domestic rates for invalid input
48 if units_consumed <= 100:
49     energy_charges = units_consumed * 3.0
50 elif units_consumed <= 300:
51     energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
52 else:
53     energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
54
55 # Display energy charges
56 print("Energy Charges (EC): $", energy_charges, ".00")
57
58 # Main function to run the billing system
59 def main():
60     # Read previous units (U1)
61     u1 = float(input("Enter previous units: "))
62     # Read current units (U2)
63     u2 = float(input("Enter current units: "))
64     # Read type of Customer
65     customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()
66     # Calculate units consumed
67     units_consumed = u2 - u1
68     # Calculate Energy Charges (EC)
69     energy_charges = calculate_energy_charges(units_consumed, customer_type)
70     # Display results
71     print("Previous units: ", u1)
72     print("Current units: ", u2)
73     print("Type of Customer: ", customer_type)
74     print("Units Consumed: ", units_consumed)
75     print("Energy Charges (EC): $", energy_charges, ".00")
76
77 # Run the main function
78 if __name__ == "__main__":
79     main()
```

Task 3: Modular Design Using AI Assistance (Using Functions)

Scenario

Billing logic must be reusable for multiple consumers.

The screenshot shows a VS Code editor with a file named `33.py`. The code implements an electricity billing system using user-defined functions. It defines a `calculate_energy_charges` function that takes units consumed and customer type as arguments and returns the calculated energy charges. The main function prompts the user for previous units, current units, and customer type, and then calls the `calculate_energy_charges` function to calculate the energy charges. The code is well-commented and includes a main function to run the billing system.

```
# 33.py
1 # Electricity Billing System with User-Defined Functions
2 # This program calculates electricity bills using modular functions for energy charges and fixed charges
3
4 def calculate_energy_charges(units_consumed, customer_type):
5     """
6     Calculate energy charges based on customer type and units consumed.
7     Uses tiered pricing structure for different consumption levels.
8     """
9     # Args:
10     # units_consumed (float): Number of units consumed
11     # customer_type (str): Type of customer ('domestic', 'commercial', 'industrial')
12
13     # Returns:
14     # float: Calculated energy charges
15
16     if customer_type == "domestic":
17         # Domestic tiered pricing: 3.0 for first 100, 5.0 for next 200, 7.0 thereafter
18         if units_consumed <= 100:
19             return units_consumed * 3.0
20         elif units_consumed <= 300:
21             return 100 * 3.0 + (units_consumed - 100) * 5.0
22         else:
23             return 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
24
25     elif customer_type == "commercial":
26         # Commercial tiered pricing: 8.0 for first 200, 12.0 thereafter
27         if units_consumed <= 200:
28             return units_consumed * 8.0
29         else:
30             return 200 * 8.0 + (units_consumed - 200) * 12.0
31
32     elif customer_type == "industrial":
33         # Industrial tiered pricing: 10.0 for first 500, 15.0 thereafter
34         if units_consumed <= 500:
35             return units_consumed * 10.0
36         else:
37             return 500 * 10.0 + (units_consumed - 500) * 15.0
38
39     # Default to domestic rates for invalid customer type
40     print("Invalid customer type. Using domestic rates.")
41     if units_consumed <= 100:
42         return units_consumed * 3.0
43     elif units_consumed <= 300:
44         return 100 * 3.0 + (units_consumed - 100) * 5.0
45     else:
46         return 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
47
48 # Main function to run the billing system
49 def main():
50     # Read previous units (U1)
51     u1 = float(input("Enter previous units: "))
52     # Read current units (U2)
53     u2 = float(input("Enter current units: "))
54     # Read type of Customer
55     customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()
56     # Calculate units consumed
57     units_consumed = u2 - u1
58     # Calculate Energy Charges (EC)
59     energy_charges = calculate_energy_charges(units_consumed, customer_type)
60     # Display results
61     print("Previous units: ", u1)
62     print("Current units: ", u2)
63     print("Type of Customer: ", customer_type)
64     print("Units Consumed: ", units_consumed)
65     print("Energy Charges (EC): $", energy_charges, ".00")
66
67 # Run the main function
68 if __name__ == "__main__":
69     main()
```

```
File Edit Selection View Go Run Terminal Help
3.3.py X
def calculate_energy_charges(units_consumed, customer_type):
    """
    Calculate energy charges based on units consumed and customer type.
    """
    if units_consumed <= 300:
        return units_consumed * 3.0
    elif units_consumed > 300:
        return 300 * 3.0 + (units_consumed - 300) * 5.0
    else:
        return 300 * 3.0 + 300 * 5.0 + (units_consumed - 300) * 7.0

def calculate_fixed_charges(customer_type):
    """
    Calculate fixed charges based on customer type.
    Fixed charges are monthly service fees.
    """
    Args:
        customer_type (str): Type of customer ('domestic', 'commercial', 'industrial')
    Returns:
        float: Fixed charges amount
    """
    if customer_type == "domestic":
        return 50.0 # Monthly service fee for domestic customers
    elif customer_type == "commercial":
        return 100.0 # Monthly service fee for commercial customers
    elif customer_type == "industrial":
        return 200.0 # Monthly service fee for industrial customers
    else:
        # Default to domestic fixed charges
        return 50.0

def main():
    """
    Main function to handle user input and display billing information.
    """
    # Read Previous Units (PU)
    pu = float(input("Enter previous units: "))
    # Read Current Units (CU)
    cu = float(input("Enter current units: "))
    # Read Type of Customer
    customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()

    # Calculate units consumed
    units_consumed = cu - pu

    # Calculate charges using functions
    energy_charges = calculate_energy_charges(units_consumed, customer_type)
    fixed_charges = calculate_fixed_charges(customer_type)

    # Calculate total bill
    total_bill = energy_charges + fixed_charges

    # Display the results
    print("\n--- Electricity Bill ---")
    print(f"Previous Units: {pu}")
    print(f"Current Units: {cu}")
    print(f"Units Consumed: {units_consumed}")
    print(f"Customer Type: {customer_type.title}")
    print(f"Energy Charges: ${energy_charges:.2f}")
    print(f"Fixed Charges: ${fixed_charges:.2f}")
    print(f"Total Bill: ${total_bill:.2f}")

    # Return calculated values as a tuple
    return energy_charges, fixed_charges, total_bill

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

PS C:\Users\ghogu\OneDrive\Documents\Desktop\Assisted coding> C:\Users\ghogu\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\ghogu\OneDrive\Documents\Desktop\Assisted coding\3.3.py"

--- Electricity Bill ---
Previous Units: 10.0
Current Units: 12.0
Units Consumed: 2.0
Customer Type: Commercial
Energy Charges: \$16.00
Fixed Charges: \$100.00
Total Bill: \$116.00

PS C:\Users\ghogu\OneDrive\Documents\Desktop\Assisted coding> Open chat [Ctrl]+[I], Show suggestions [Ctrl]+[Tab], Start typing to discuss or don't show this again.

CHAT

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

Uses user-defined functions to Calculate Energy Charges, Calculate Fixed Charges. Returns calculated values, includes meaningful comments in python code

✓ Read file and created new Python file

✓ C:\Users\ghogu\AppData\Local\Programs\Python\Python311\python.exe
✓ 3/python.exe
✓ "C:\Users\ghogu\OneDrive\Documents\Desktop\Assisted coding\3.3.py"

✓ Checked + 3.3.py, no problems found

I've successfully refactored the electricity billing code in + 3.3.py to use user-defined functions as requested. Here's what I implemented:

Key Features Added:

- 1. **calculate_energy_charges**(units_consumed, customer_type) - A function that calculates energy charges based on tiered pricing for different customer types (domestic, commercial, industrial).
- 2. **calculate_fixed_charges**(customer_type) - A function that returns fixed monthly service fees based on customer type.
- 3. **main()** - A main function that handles user input, calls the calculation functions, displays results, and returns the calculated values as a tuple.

Code Structure:

- **Modular Design:** Separated calculations into reusable functions.
- **Meaningful Comments:** Added docstrings and inline comments explaining each function's purpose and logic.
- **Error Handling:** Includes fallback to domestic rates for invalid customer types.
- **Return Values:** Functions return calculated amounts as specified.
- **Clear Output:** Formatted bill display with all charges and totals.

The code runs successfully without errors and produces accurate billing calculations. The functions are well-documented and follow Python best practices for readability and maintainability.

Describe what to build next

Agents Auto

```
File Edit Selection View Go Run Terminal Help
3.3.py X
def main():
    """
    Main function to handle user input and display billing information.
    """
    # Read Previous Units (PU)
    pu = float(input("Enter previous units: "))
    # Read Current Units (CU)
    cu = float(input("Enter current units: "))
    # Read Type of Customer
    customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()

    # Calculate units consumed
    units_consumed = cu - pu

    # Calculate charges using functions
    energy_charges = calculate_energy_charges(units_consumed, customer_type)
    fixed_charges = calculate_fixed_charges(customer_type)

    # Calculate total bill
    total_bill = energy_charges + fixed_charges

    # Display the results
    print("\n--- Electricity Bill ---")
    print(f"Previous Units: {pu}")
    print(f"Current Units: {cu}")
    print(f"Units Consumed: {units_consumed}")
    print(f"Customer Type: {customer_type.title}")
    print(f"Energy Charges: ${energy_charges:.2f}")
    print(f"Fixed Charges: ${fixed_charges:.2f}")
    print(f"Total Bill: ${total_bill:.2f}")

    # Return calculated values as a tuple
    return energy_charges, fixed_charges, total_bill

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

PS C:\Users\ghogu\OneDrive\Documents\Desktop\Assisted coding> C:\Users\ghogu\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\ghogu\OneDrive\Documents\Desktop\Assisted coding\3.3.py"

--- Electricity Bill ---
Previous Units: 10.0
Current Units: 12.0
Units Consumed: 2.0
Customer Type: Commercial
Energy Charges: \$16.00
Fixed Charges: \$100.00
Total Bill: \$116.00

PS C:\Users\ghogu\OneDrive\Documents\Desktop\Assisted coding> Open chat [Ctrl]+[I], Show suggestions [Ctrl]+[Tab], Start typing to discuss or don't show this again.

CHAT

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

Uses user-defined functions to Calculate Energy Charges, Calculate Fixed Charges. Returns calculated values, includes meaningful comments in python code

✓ Read file and created new Python file

✓ C:\Users\ghogu\AppData\Local\Programs\Python\Python311\python.exe
✓ 3/python.exe
✓ "C:\Users\ghogu\OneDrive\Documents\Desktop\Assisted coding\3.3.py"

✓ Checked + 3.3.py, no problems found

I've successfully refactored the electricity billing code in + 3.3.py to use user-defined functions as requested. Here's what I implemented:

Key Features Added:

- 1. **calculate_energy_charges**(units_consumed, customer_type) - A function that calculates energy charges based on tiered pricing for different customer types (domestic, commercial, industrial).
- 2. **calculate_fixed_charges**(customer_type) - A function that returns fixed monthly service fees based on customer type.
- 3. **main()** - A main function that handles user input, calls the calculation functions, displays results, and returns the calculated values as a tuple.

Code Structure:

- **Modular Design:** Separated calculations into reusable functions.
- **Meaningful Comments:** Added docstrings and inline comments explaining each function's purpose and logic.
- **Error Handling:** Includes fallback to domestic rates for invalid customer types.
- **Return Values:** Functions return calculated amounts as specified.
- **Clear Output:** Formatted bill display with all charges and totals.

The code runs successfully without errors and produces accurate billing calculations. The functions are well-documented and follow Python best practices for readability and maintainability.

Describe what to build next

Agents Auto

Task 4: Calculation of Additional Charges

Scenario

Electricity bills include multiple additional charges.

```
File Edit Selection View Go Run Terminal Help
electric_bill.py X 3.py

# Electricity Billing System - Enhanced Calculations
# Read previous units (pu)
pu = float(input("Enter previous units: "))

# Read current units (cu)
cu = float(input("Enter current units: "))

# Read type of customer
customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()

# Calculate units consumed
units_consumed = cu - pu

# Display the results
print(f"Previous units: {pu}")
print(f"Current units: {cu}")
print(f"Type of customer: {customer_type}")
print(f"Units Consumed: {units_consumed}")

# Calculate Energy Charges (EC) based on customer type and units consumed
# Electricity energy charge calculation logic using conditional statements
if customer_type == "domestic":
    # Domestic: tiered pricing for better readability
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif 100 < units_consumed <= 200:
        energy_charges = 100 * 3.0 + 3.0 * (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 3.0 + 3.0 * (units_consumed - 300) * 7.0
elif customer_type == "commercial":
    # Commercial: tiered pricing optimized for clarity
    if units_consumed <= 100:
        energy_charges = units_consumed * 8.0
    else:
        energy_charges = 100 * 8.0 + 8.0 * (units_consumed - 100) * 12.0
elif customer_type == "industrial":
    # Industrial: tiered pricing
    if units_consumed <= 100:
        energy_charges = units_consumed * 10.0
    elif 100 < units_consumed <= 200:
        energy_charges = 100 * 10.0 + 10.0 * (units_consumed - 100) * 15.0
    else:
        print("Invalid customer type. Assuming domestic rates.")
        # Default to domestic rates for invalid input
        if units_consumed <= 100:
            energy_charges = units_consumed * 3.0
        else:
            energy_charges = 100 * 3.0 + 3.0 * (units_consumed - 100) * 5.0

# Calculate Fixed Charges (FC) based on customer type
if customer_type == "domestic":
    fixed_charges = 10.0 # Monthly service fee for domestic customers
elif customer_type == "commercial":
    fixed_charges = 100.0 # Monthly service fee for commercial customers
elif customer_type == "industrial":
    fixed_charges = 200.0 # Monthly service fee for industrial customers
else:
    fixed_charges = 10.0 # Default to domestic fixed charges

# Calculate Electricity Duty (ED) as percentage of energy charges (EC)
# Electricity duty rates vary by customer type for improved billing accuracy
if customer_type == "domestic":
    electricity_duty_rate = 0.05 # 5% for domestic
elif customer_type == "commercial":
    electricity_duty_rate = 0.08 # 8% for commercial
elif customer_type == "industrial":
    electricity_duty_rate = 0.10 # 10% for industrial
else:
    electricity_duty_rate = 0.05 # Default 5%

electricity_duty = energy_charges * electricity_duty_rate

# Calculate Customer Charges (CC) - Total charges including all components
customer_charges = energy_charges + fixed_charges + electricity_duty

# Display all charges (formatted for better readability)
print(f"--- Electricity Bill Details ---")
print(f"Energy Charges (EC): $10.00")
print(f"Fixed Charges (FC): $100.00")
print(f"Electricity Duty (ED): $1.50 (5%)")
print(f"Customer Charges (CC): $111.50
```

```
File Edit Selection View Go Run Terminal Help
electric_bill.py X 3.py

# Electricity Billing System - Enhanced Calculations
# Read previous units (pu)
pu = float(input("Enter previous units: "))

# Read current units (cu)
cu = float(input("Enter current units: "))

# Read type of customer
customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()

# Calculate units consumed
units_consumed = cu - pu

# Display the results
print(f"Previous units: {pu}")
print(f"Current units: {cu}")
print(f"Type of customer: {customer_type}")
print(f"Units Consumed: {units_consumed}")

# Calculate Energy Charges (EC) based on customer type and units consumed
# Electricity energy charge calculation logic using conditional statements
if customer_type == "domestic":
    # Domestic: tiered pricing for better readability
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif 100 < units_consumed <= 200:
        energy_charges = 100 * 3.0 + 3.0 * (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 3.0 + 3.0 * (units_consumed - 300) * 7.0
elif customer_type == "commercial":
    # Commercial: tiered pricing optimized for clarity
    if units_consumed <= 100:
        energy_charges = units_consumed * 8.0
    else:
        energy_charges = 100 * 8.0 + 8.0 * (units_consumed - 100) * 12.0
elif customer_type == "industrial":
    # Industrial: tiered pricing
    if units_consumed <= 100:
        energy_charges = units_consumed * 10.0
    elif 100 < units_consumed <= 200:
        energy_charges = 100 * 10.0 + 10.0 * (units_consumed - 100) * 15.0
    else:
        print("Invalid customer type. Assuming domestic rates.")
        # Default to domestic rates for invalid input
        if units_consumed <= 100:
            energy_charges = units_consumed * 3.0
        else:
            energy_charges = 100 * 3.0 + 3.0 * (units_consumed - 100) * 5.0

# Calculate Fixed Charges (FC) based on customer type
if customer_type == "domestic":
    fixed_charges = 10.0 # Monthly service fee for domestic customers
elif customer_type == "commercial":
    fixed_charges = 100.0 # Monthly service fee for commercial customers
elif customer_type == "industrial":
    fixed_charges = 200.0 # Monthly service fee for industrial customers
else:
    fixed_charges = 10.0 # Default to domestic fixed charges

# Calculate Electricity Duty (ED) as percentage of energy charges (EC)
# Electricity duty rates vary by customer type for improved billing accuracy
if customer_type == "domestic":
    electricity_duty_rate = 0.05 # 5% for domestic
elif customer_type == "commercial":
    electricity_duty_rate = 0.08 # 8% for commercial
elif customer_type == "industrial":
    electricity_duty_rate = 0.10 # 10% for industrial
else:
    electricity_duty_rate = 0.05 # Default 5%

electricity_duty = energy_charges * electricity_duty_rate

# Calculate Customer Charges (CC) - Total charges including all components
customer_charges = energy_charges + fixed_charges + electricity_duty

# Display all charges (formatted for better readability)
print(f"--- Electricity Bill Details ---")
print(f"Energy Charges (EC): $10.00")
print(f"Fixed Charges (FC): $100.00")
print(f"Electricity Duty (ED): $1.50 (5%)")
print(f"Customer Charges (CC): $111.50
```

Task 5: Final Bill Generation and Output Analysis

Scenario

The final electricity bill must present all values clearly.

The image shows a VS Code editor with a Python script named `electricity_billing.py` and its output in the terminal.

Python Script: `electricity_billing.py`

```
#!/usr/bin/env python3
# Electricity Billing System with Extended Accuracy and Real-world Features
# This program calculates comprehensive electricity bills with multiple charge components

def main():
    # Input validation and error handling for real-world applicability
    try:
        # Read Previous units (PU) with validation
        pu = float(input("Enter previous units: "))
        if pu < 0:
            raise ValueError("Previous units cannot be negative")

        # Read Current units (CU) with validation
        cu = float(input("Enter current units: "))
        if cu < 0:
            raise ValueError("Current units cannot be negative")

        # Read type of customer with validation
        customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower().strip()
        valid_types = ['domestic', 'commercial', 'industrial']
        if customer_type not in valid_types:
            print(f"Invalid customer type '{customer_type}'. Valid options: {', '.join(valid_types)}")
            customer_type = 'domestic' # default fallback
            print("Using default customer type: {customer_type}")

    except ValueError as e:
        print(f"Error: {e}")
        print("Please enter valid numeric values for units.")
        return

    # Calculate units consumed with validation
    units_consumed = cu - pu

    # Handle negative consumption (possible meter reset or error)
    if units_consumed < 0:
        print(f"Warning: Current units are less than previous units.")
        print("This may indicate a meter reading error or meter reset.")
        print("Setting energy charges to $0.00 for this billing cycle.")
        electricity_bill = 0.0
        units_consumed_display = f"Units consumed: 0 (adjusted to 0 for a billing)"
    else:
        units_consumed_display = f"Units consumed: {units_consumed}"

    # Calculate Energy Charge (EC) based on customer type and units consumed
    # Domestic: $0.12 per kWh, Commercial: $0.15 per kWh, Industrial: $0.20 per kWh
    energy_charge = units_consumed * (0.12 if customer_type == 'domestic' else 0.15 if customer_type == 'commercial' else 0.20)

    # Billing Summary
    print("\n--- Meter Reading Summary ---")
    print(f"Previous Units: {pu}")
    print(f"Current Units: {cu}")
    print(f"Customer Type: {customer_type}")
    print(f"Units Consumed: {units_consumed}")

    # Electricity Bill Details
    print("\n--- Electricity Bill Details ---")
    print(f"Energy Charge (EC): ${energy_charge:.2f}")
    print(f"Fixed Charge (FC): $200.00")
    print(f"Customer Charge (CC): $50.00")
    print(f"Electricity Duty (ED): $0.00")
    print(f"Total Bill Amount: $250.00")

    # Bill Summary for Industrial Customer
    print("\n--- Bill Summary for Industrial Customer ---")
    print("Note: Structure: Tiered pricing applied")
```

Terminal Output:

```
PS C:\Users\gking> cd "C:\Users\gking\Documents\Desktop\Assisted coding" & C:\Users\gking\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\gking\Documents\Desktop\Assisted coding\electricity_billing.py"
Enter previous units: 12
Enter current units: 35
Enter type of customer (domestic/commercial/industrial): industrial

--- Meter Reading Summary ---
Previous Units: 12.0
Current Units: 35.0
Customer Type: Industrial
Units Consumed: 23.0

--- Electricity Bill Details ---
Energy Charge (EC): $345.00
Fixed Charge (FC): $200.00
Customer Charge (CC): $50.00
Electricity Duty (ED): $0.00
Total Bill Amount: $595.00

--- Bill Summary for Industrial Customer ---
Note: Structure: Tiered pricing applied
```

The screenshot displays a Windows 10 desktop environment with a code editor, a terminal, and a file explorer.

Code Editor (Left Panel): The editor shows a Python script named `electricity_bill.py`. The script calculates electricity bills based on units consumed and customer type (domestic, commercial, industrial). It includes comments for each step and uses conditional logic to apply different rates and charges.

```

1 # electricity_bill.py
2
3 # Customer type and units consumed
4 units_consumed = 150 # (adjusted to 1 hr billing)
5
6 # Calculate electricity bill
7 if units_consumed < 10:
8     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
9     # Customer type = "domestic"
10    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
11
12 elif units_consumed < 30:
13     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
14     # Customer type = "domestic"
15    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
16
17 elif units_consumed < 50:
18     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
19     # Customer type = "domestic"
20    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
21
22 elif units_consumed < 100:
23     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
24     # Customer type = "domestic"
25    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
26
27 elif units_consumed < 200:
28     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
29     # Customer type = "domestic"
30    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
31
32 elif units_consumed < 300:
33     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
34     # Customer type = "domestic"
35    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
36
37 elif units_consumed < 500:
38     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
39     # Customer type = "domestic"
40    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
41
42 elif units_consumed < 1000:
43     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
44     # Customer type = "domestic"
45    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
46
47 elif units_consumed < 2000:
48     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
49     # Customer type = "domestic"
50    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
51
52 elif units_consumed < 3000:
53     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
54     # Customer type = "domestic"
55    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
56
57 elif units_consumed < 5000:
58     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
59     # Customer type = "domestic"
60    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
61
62 elif units_consumed < 10000:
63     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
64     # Customer type = "domestic"
65    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
66
67 elif units_consumed < 20000:
68     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
69     # Customer type = "domestic"
70    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
71
72 elif units_consumed < 30000:
73     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
74     # Customer type = "domestic"
75    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
76
77 elif units_consumed < 50000:
78     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
79     # Customer type = "domestic"
80    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
81
82 elif units_consumed < 100000:
83     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
84     # Customer type = "domestic"
85    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
86
87 elif units_consumed < 200000:
88     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
89     # Customer type = "domestic"
90    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
91
92 elif units_consumed < 300000:
93     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
94     # Customer type = "domestic"
95    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
96
97 elif units_consumed < 500000:
98     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
99     # Customer type = "domestic"
100   energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
101
102 elif units_consumed < 1000000:
103     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
104     # Customer type = "domestic"
105    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
106
107 elif units_consumed < 2000000:
108     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
109     # Customer type = "domestic"
110    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
111
112 elif units_consumed < 3000000:
113     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
114     # Customer type = "domestic"
115    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
116
117 elif units_consumed < 5000000:
118     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
119     # Customer type = "domestic"
120    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
121
122 elif units_consumed < 10000000:
123     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
124     # Customer type = "domestic"
125    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
126
127 elif units_consumed < 20000000:
128     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
129     # Customer type = "domestic"
130    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
131
132 elif units_consumed < 30000000:
133     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
134     # Customer type = "domestic"
135    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
136
137 elif units_consumed < 50000000:
138     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
139     # Customer type = "domestic"
140    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
141
142 elif units_consumed < 100000000:
143     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
144     # Customer type = "domestic"
145    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
146
147 elif units_consumed < 200000000:
148     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
149     # Customer type = "domestic"
150    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
151
152 elif units_consumed < 300000000:
153     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
154     # Customer type = "domestic"
155    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
156
157 elif units_consumed < 500000000:
158     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
159     # Customer type = "domestic"
160    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
161
162 elif units_consumed < 1000000000:
163     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
164     # Customer type = "domestic"
165    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
166
167 elif units_consumed < 2000000000:
168     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
169     # Customer type = "domestic"
170    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
171
172 elif units_consumed < 3000000000:
173     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
174     # Customer type = "domestic"
175    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
176
177 elif units_consumed < 5000000000:
178     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
179     # Customer type = "domestic"
180    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
181
182 elif units_consumed < 10000000000:
183     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
184     # Customer type = "domestic"
185    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
186
187 elif units_consumed < 20000000000:
188     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
189     # Customer type = "domestic"
190    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
191
192 elif units_consumed < 30000000000:
193     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
194     # Customer type = "domestic"
195    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
196
197 elif units_consumed < 50000000000:
198     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
199     # Customer type = "domestic"
200    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
201
202 elif units_consumed < 100000000000:
203     # Domestic: flat rate pricing - 10.0 for first 10, 1.0 thereafter
204     # Customer type = "domestic"
205    energy_charges = 10.0 + 1.0 * (units_consumed - 10) * 1.0
206
207 elif units_consumed < 200000000000:
208     # Domestic: flat rate pricing - 10.
```

