

Task 1: AI-Generated Logic for Reading Consumer Details

Scenario

An electricity billing system must collect accurate consumer data.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows a tree view of files. The "ASSISTED CODING" folder contains "electricity_billing.py". Other files listed include "1.yz", "2.spy", "Assignment1.py", "electricity_billng.py", "Lab Assignment 1.pdf", "lab_assignment_2.pdf", and "README.md".
- Terminal:** The terminal window shows the following command and output:

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> cd "C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding"; C:\Users\gbsgu\AppData\Local\Programs\Python\Python311\python.exe electricity_billng.py
your previous units?
Enter previous units: 5
Enter current units: 9
Enter type of customer (residential/commercial): commercial
Previous Units: 5.0
Current Units: 9.0
Type of Customer: Commercial
Units Consumed: 4.0
Rate per unit: 38.0
Total Bill: 48.0
```
- Output:** The output panel shows the generated electricity bill.
- Problems:** No problems are listed.
- Console:** The console shows the command to run the script and its path.
- Status Bar:** The status bar at the bottom right indicates "File 1 Col 1 Spaced 1 UFS CPU Python 3.11.0".

Task 2: Energy Charges Calculation Based on Units Consumed

Scenario

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

The screenshot shows a Jupyter Notebook environment with two tabs open:

- electricity_billng.py**: The active tab contains Python code for calculating electricity bills based on consumption units (U) and customer type (Residential, Commercial, Industrial). It uses nested if statements to determine charges per unit and total bill.
- electricity_billng.ipynb**: A notebook tab showing the execution history of the previous script.

The code in `electricity_billng.py` includes:

```
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)", pu)
17 print("Current units (cu)", cu)
18 print("Type of Customer", customer_type)
19 print("Units Consumed", units_consumed)
20
21 # Non-residential calculation (assuming retail)
22 rate_per_unit = 5.0 # example rate
23
24 # Residential charge calculation using consumer type and units consumed
25 # Electricity usage calculation is done using conditional otherwise
26 if customer_type == "residential":
27     # Residential charge calculation for better readability
28     if units_consumed < 100:
29         energy_charges = units_consumed * 5.0
30     else:
31         energy_charges = 100 * 5.0 + (units_consumed - 100) * 7.0
32
33     # Residential charge calculation with clear structure
34     if units_consumed < 100:
35         energy_charges = units_consumed * 5.0
36     else:
37         energy_charges = 100 * 5.0 + (units_consumed - 100) * 7.0
38
39 else:
40     print("Invalid customer type. Assuming residential rate")
41
42     rate_per_unit = 5.0
43
44 # Residential charge calculation
45 if units_consumed > 200:
46     energy_charges = 200 * 5.0
47 else:
48     energy_charges = 200 * 5.0 + (units_consumed - 200) * 12.0
49
50 # Industrial charge calculation
51 if units_consumed < 500:
52     energy_charges = units_consumed * 10.0
53 else:
54     energy_charges = 500 * 10.0 + (units_consumed - 500) * 15.0
55
56 # Overall charge calculation
57 total_bill = units_consumed * rate_per_unit
58
59 print("Rate per unit", rate_per_unit)
60 print("Total Bill", total_bill)
61
62 print("Overall charge calculation Assuming Domestic rates:")
63
64 if units_consumed < 100:
65     energy_charge = units_consumed * 5.0
66 else:
67     energy_charge = 100 * 5.0 + (units_consumed - 100) * 7.0
68
69 if units_consumed > 200:
70     energy_charge = 200 * 5.0
71 else:
72     energy_charge = 200 * 5.0 + (units_consumed - 200) * 12.0
73
74 print("Energy Charge", energy_charge)
```

```

# UNLIMITED-1 electricity_billing.py
# Electricity Billing System
# This program calculates electricity bills using conditional statements for different customer types.
# Based on the input provided, it calculates the total bill.
# Author: [REDACTED]
# Date: [REDACTED]

# Read Previous Units
pu = float(input("Enter previous units: "))

# Read Current Units
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("Previous Units: ({})".format(pu))
print("Current Units: ({})".format(cu))
print("Type of Customer: ({})".format(customer_type))
print("Units Consumed: ({})".format(units_consumed))

# Calculate Energy Charges (EC) based on customer type and units consumed
# Simplify energy charge calculation logic using conditional statements
if customer_type == "domestic":
    # Domestic tiered pricing with clear structure
    if units_consumed <= 100:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
    print("Domestic rates applied for clarity")
elif units_consumed > 200:
    energy_charges = units_consumed * 8.0
else:
    energy_charges = 100 * 3.0 + (units_consumed - 200) * 12.0
    print("Industrial tiered pricing with clear structure")
    if units_consumed <= 500:
        energy_charges = units_consumed * 10.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 500) * 15.0
else:
    print("Customer type: {}, assuming domestic rates.".format(customer_type))
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    else:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
    print("Industrial rates applied for clarity")
    if units_consumed <= 200:
        energy_charges = units_consumed * 8.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 200) * 12.0
# Display energy charges
print("Energy Charges (EC): ${energy_charges:.2f}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\gupta\OneDrive\Documents\Desktop\Assisted coding & C:\Users\gupta\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\gupta\OneDrive\Documents\Desktop\Assisted coding\electricity_billing.py"

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

Scenario

Billing logic must be reusable for multiple consumers.

```

# 3.3.py -
# Electricity Billing System with User-Defined Functions
# This program calculates electricity bills using modular functions for energy charges and Fixed charges
# Author: [REDACTED]
# Date: [REDACTED]

def calculate_energy_charges(units_consumed, customer_type):
    """
    Calculate energy charges based on customer type and units consumed.
    Uses tiered pricing structure for different consumption levels.
    """
    Args:
        units_consumed (float): Number of units consumed
        customer_type (str): Type of customer ('domestic', 'commercial', 'industrial')

    Returns:
        float: Calculated energy charges
    """
    if customer_type == "domestic":
        # Domestic tiered pricing: 3.0 for first 100, 5.0 for next 200, 7.0 thereafter
        if units_consumed <= 100:
            return units_consumed * 3.0
        elif units_consumed <= 300:
            return 100 * 3.0 + (units_consumed - 100) * 5.0
        else:
            return 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
    elif customer_type == "commercial":
        # Commercial tiered pricing: 8.0 for first 200, 12.0 thereafter
        if units_consumed <= 200:
            return units_consumed * 8.0
        else:
            return 200 * 8.0 + (units_consumed - 200) * 12.0
    elif customer_type == "industrial":
        # Industrial tiered pricing: 10.0 for first 500, 15.0 thereafter
        if units_consumed <= 500:
            return units_consumed * 10.0
        else:
            return 500 * 10.0 + (units_consumed - 500) * 15.0
    else:
        print("Default to domestic rates for invalid customer type")
        print("Invalid customer type. Using domestic rates.")
        if units_consumed <= 100:
            return units_consumed * 3.0
        elif units_consumed > 200:
            return 100 * 3.0 + 200 * 5.0 + (units_consumed - 200) * 12.0
    """
    Default to domestic rates for invalid customer type
    print("Invalid customer type. Using domestic rates.")
    if units_consumed <= 100:
        return units_consumed * 3.0
    elif units_consumed > 200:
        return 100 * 3.0 + 200 * 5.0 + (units_consumed - 200) * 12.0
    else:
        print("Default to domestic rates for invalid customer type")
        print("Invalid customer type. Using domestic rates.")

    PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
    """

    ... Electricity Bill ...
    Previous Units: 10.0
    Current Units: 15.0
    Units Consumed: 5.0
    Customer Type: Commercial
    Fixed Charges: $100.00
    Total Bill: $115.00
    PS C:\Users\gupta\OneDrive\Documents\Desktop\Assisted coding & C:\Users\gupta\AppData\Local\Programs\Python\Python311\python.exe "C:\Users\gupta\OneDrive\Documents\Desktop\Assisted coding\3.3.py"

```

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

```

3.3.py
4 def calculate_energy_charges(units_consumed, customer_type):
5     if units_consumed <= 300:
6         return 3.0 * units_consumed
7     elif units_consumed < 600:
8         return 180 + 3.0 * (units_consumed - 300)
9     else:
10        return 180 + 200 + 5.0 * (units_consumed - 600) + 7.0
11
12 def calculate_fixed_charges(customer_type):
13     Calculate fixed charges based on customer type.
14     Fixed charges are monthly service fees.
15
16     Args:
17         customer_type (str): Type of customer ('domestic', 'commercial', 'industrial')
18
19     Returns:
20         float: Fixed charges amount
21
22     if customer_type == "domestic":
23         return 50.0 # Monthly service fee for domestic customers
24     elif customer_type == "commercial":
25         return 180 # Monthly service fee for commercial customers
26     elif customer_type == "industrial":
27         return 200.0 # Monthly service fee for industrial customers
28     else:
29         # Default to domestic fixed charges
30         return 50.0
31
32 def main():
33     Main function to handle user input and display billing information.
34     calls the calculation functions and returns the calculated values.
35
36     # Read Previous Units (PU)
37     pu = float(input("Enter previous units: "))
38
39     # Read Current Units (CU)
40     cu = float(input("Enter current units: "))
41
42     # Read Type of Customer
43     customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()
44
45     # Calculate units consumed
46     units_consumed = cu - pu
47
48     # Calculate charges using functions
49     energy_charges = calculate_energy_charges(units_consumed, customer_type)
50     fixed_charges = calculate_fixed_charges(customer_type)
51
52     # Calculate total bill
53     total_bill = energy_charges + fixed_charges
54
55     # Display the results
56     print("Electricity Bill ---")
57     print(f"Previous Units: ({pu})")
58     print(f"Current Units: ({cu})")
59     print(f"Units Consumed: {units_consumed}")
60     print(f"Customer Type: {customer_type.title()}")
61     print(f"Energy Charges: ${energy_charges:.2f}")
62     print(f"Fixed Charges: ${fixed_charges:.2f}")
63     print(f"Total Bill: ${total_bill:.2f}")
64
65     # Return calculated values as a tuple
66     return energy_charges, fixed_charges, total_bill
67
68 # Run the main function if this script is executed directly
69 if __name__ == "__main__":
70     main()

```



```

3.3.2.py
67 def main():
68     calls the calculation functions and returns the calculated values.
69
70     # Read Previous Units (PU)
71     pu = float(input("Enter previous units: "))
72
73     # Read Current Units (CU)
74     cu = float(input("Enter current units: "))
75
76     # Read Type of Customer
77     customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()
78
79     # Calculate units consumed
80     units_consumed = cu - pu
81
82     # Calculate charges using functions
83     energy_charges = calculate_energy_charges(units_consumed, customer_type)
84     fixed_charges = calculate_fixed_charges(customer_type)
85
86     # Calculate total bill
87     total_bill = energy_charges + fixed_charges
88
89     # Display the results
90     print("Electricity Bill ---")
91     print(f"Previous Units: ({pu})")
92     print(f"Current Units: ({cu})")
93     print(f"Units Consumed: {units_consumed}")
94     print(f"Customer Type: {customer_type.title()}")
95     print(f"Energy Charges: ${energy_charges:.2f}")
96     print(f"Fixed Charges: ${fixed_charges:.2f}")
97     print(f"Total Bill: ${total_bill:.2f}")
98
99     # Return calculated values as a tuple
100    return energy_charges, fixed_charges, total_bill
101
102 # Run the main function if this script is executed directly
103 if __name__ == "__main__":
104    main()

```

Task 4: Calculation of Additional Charges

Scenario

Electricity bills include multiple additional charges.

```
File Edit Selection View Go Run Terminal Help ← → Q Assisted coding

electricity_bill.py ①.py

1 # Electricity Billing System [Enhanced Calculations]
2
3 pu = float(input("Enter previous units: "))
4
5 # Read Current Units (CU)
6 cu = 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 = cu - pu
13
14 # Display the Results
15 print("Previous Units: (PU) ", pu)
16 print("Current Units: (CU) ", cu)
17 print("Type of Customer: (Customer_Type) ", customer_type)
18 print("Units Consumed: (units_consumed) ", units_consumed)
19
20 # Define Functions for Energy Charge Calculation
21 def calculate_energy_charge(units_consumed):
22     if customer_type == "domestic":
23         if units_consumed < 100:
24             energy_charges = 10.0 * units_consumed
25         else:
26             energy_charges = 10.0 + (units_consumed - 100) * 5.0
27     else:
28         energy_charges = 10.0 + 3.0 * 200 + 5.0 * (units_consumed - 100) * 7.0
29
30     return energy_charges
31
32 # Define Functions for Fixed Charge Calculation
33 def calculate_fixed_charge(customer_type):
34     if customer_type == "domestic":
35         fixed_charge = 0.0
36     elif customer_type == "commercial":
37         fixed_charge = 10.0
38     else:
39         fixed_charge = 20.0
40
41     return fixed_charge
42
43 # Define Functions for Electricity Duty Calculation
44 def calculate_electricity_duty(energy_charge):
45     electricity_duty_rate = 0.05
46
47     electricity_duty = energy_charge * electricity_duty_rate
48
49     return electricity_duty
50
51 # Calculate Total Customer Charges (CCC)
52 def calculate_customer_charges(ccc):
53     total_charges = ccc + fixed_charge + electricity_duty
54
55     return total_charges
56
57 # Display All Charges [Improved Formating for Billing Accuracy]
58 def display_bill_details(ccc):
59     print("----- Electricity Bill Details -----")
60     print("Electricity Bill Details: ")
61     print("Energy Charge (EC): $", str(ccc))
62     print("Fixed Charge (FC): $", str(fixed_charge))
63     print("Electricity Duty (ED): $", str(electricity_duty))
64     print("Customer Charges (CC): $", str(total_charges))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\gupta\OneDrive\Documents\Desktop\Assisted Coding> python electricity_bill.py
Enter previous units: 10
Enter current units: 30
Enter type of customer (domestic/commercial/industrial): domestic
Customer Type: domestic
Units Consumed: 20.0
Current Units: 30.0
Type of Customer: domestic
Units Consumed: 20.0

--- Electricity Bill Details ---
Energy Charge (EC): $20.00
Fixed Charge (FC): $0.00
Electricity Duty (ED): $1.00
Customer Charges (CC): $21.00

PS C:\Users\gupta\OneDrive\Documents\Desktop\Assisted Coding>
```

Task 5: Final Bill Generation and Output Analysis

Scenario

The final electricity bill must present all values clearly.

```

# electricity_billing.py
# This program calculates comprehensive electricity bills with multiple charge components
# 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()
    valid_types = ["domestic", "commercial", "industrial"]
    if customer_type not in valid_types:
        print(f"Invalid customer type '{customer_type}'. Valid options: {', '.join(valid_types)}")
        exit(1)
    # calculate units consumed with validation
    units_consumed = cu - pu
    # handle negative consumption (possible meter reset or error)
    if units_consumed < 0:
        print("⚠️ Aaaah! 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_duty = 0.0
        units_consumed_display = f"({units_consumed} U)" # (adjusted to 0 for billing)"
    else:
        units_consumed_display = f"({units_consumed} U)"
    # calculate Energy Charges (EC) based on customer type and units consumed
    # Peter Heding Summary ...
    previous_units = 31.8
    current_units = 31.8
    customer_type = "domestic"
    units_consumed = 3.0
    # Electricity Bill Details ...
    energy_charge = 0.0
    fixed_charge = 0.0
    customer_charge = 0.0
    electricity_duty = 0.0
    total_bill_amount = 0.0
    # ... Bill Summary for Industrial Customer ...
    # Rate structure: Thread pricing applied

```

```

# electricity_billing.py
# This program calculates comprehensive electricity bills with multiple charge components
# 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()
    valid_types = ["domestic", "commercial", "industrial"]
    if customer_type not in valid_types:
        print(f"Invalid customer type '{customer_type}'. Valid options: {', '.join(valid_types)}")
        exit(1)
    # calculate units consumed with validation
    units_consumed = cu - pu
    # handle negative consumption (possible meter reset or error)
    if units_consumed < 0:
        print("⚠️ Aaaah! 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_duty = 0.0
        units_consumed_display = f"({units_consumed} U)" # (adjusted to 0 for billing)"
    else:
        units_consumed_display = f"({units_consumed} U)"
    # calculate Energy Charges (EC) based on customer type and units consumed
    # Peter Heding Summary ...
    previous_units = 31.8
    current_units = 31.8
    customer_type = "domestic"
    units_consumed = 3.0
    # Electricity Bill Details ...
    energy_charge = 0.0
    fixed_charge = 0.0
    customer_charge = 0.0
    electricity_duty = 0.0
    total_bill_amount = 0.0
    # ... Bill Summary for Industrial Customer ...
    # Rate structure: Thread pricing applied

```

The screenshot shows a Python code editor with two tabs: `electricity_bill.py` and `electricity_bill.py`. The code in `electricity_bill.py` is as follows:

```
# electric_bill.py
# This program calculates electricity bill based on meter reading and customer type.

# Define constants
DOMESTIC_CHARGE = 15.0
COMMERCIAL_CHARGE = 25.0
INDUSTRIAL_CHARGE = 40.0

# Input variables
customer_type = "domestic"
units_consumed = 5.0
customer_charge = 0.0
customer_duty = 0.0
customer_duty_rate = 0.0
customer_duty_charge = 0.0
customer_charges = 0.0
customer_duty_charge = 0.0
customer_charges = 0.0

# Calculate total bill amount
total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty

# Display the bill summary
print("(*-- Meter Reading Summary --*)")
print("Previous Units: (u.u)f")
print("Current Units: (u.u)f")
print("Customer Type: (customer_type.title())")
print("Units Consumed: (units_consumed.display())")

# Note: All charges are in rupees, except for meter reading
print("(*-- Electricity Bill Details --*)")
print("Energy Charges (EC) : Energy_charge")
print("Fixed Charges (FC) : Fixed_charge")
print("Customer Charge (CC) : Customer_charge")
print("Electricity Duty (ED) : Electricity_duty")
print("Final Bill Amount : total_bill")

# Real-world applicability notes
print("(*-- Bill Summary for customer_type.title() Customer --*)")
print("Note: This program does not consider tiered pricing applied")
if units_consumed >= 0:
    if units_consumed < 0:
        print("Error: Current units are less than previous units. Please verify meter readings.")
    if energy_charge < 0:
        print("Error: Negative energy charges detected. Check input values.")

PROBLEMS OUTPUT DEBUGGING TERMINAL PWS
```

The terminal window shows the execution of the script and its output:

```
$ python3 electricity_bill.py
Enter previous units: 15
Enter current units: 20
Enter type of customer (domestic/commercial/industrial): industrial

-- Meter Reading Summary --
Previous Units: 15.0
Current Units: 20.0
Customer Type: Industrial
Units Consumed: 5.0

-- Electricity Bill Details --
Energy Charges (EC) : 100.00
Fixed Charges (FC) : 250.00
Customer Charge (CC) : 0.00
Electricity Duty (ED) : 100.00
Total Bill amount: 375.00

-- Bill Summary for Industrial Customer --
No tiered pricing applied
```

This program accurately calculates the electricity bill by using basic arithmetic formulas. The code is easy to read because of meaningful variable names and clear print statements. It is applicable in real-world situations as it follows the standard electricity billing structure used by power departments. The formatted output helps users understand each charge clearly.