

## **AI ASSISTANT CODING LAB-3**

**Rollno:2303A52291 batch:42**

**Name:M.Shashidhar**

### **Task 1: AI-Generated Logic for Reading Consumer Details**

#### **Scenario**

**An electricity billing system must collect accurate consumer data.**

#### **Task Description**

**Use an AI tool (GitHub Copilot / Gemini) to generate a Python program that:**

- **Reads:**
  - Previous Units (PU)
  - Current Units (CU)
  - Type of Customer
- **Calculates units consumed**
- **Implements logic directly in the main program (no functions)**

#### **PROMPT:**

**Generate a Python program (without using functions) to read:**

- Previous Units (PU)
- Current Units (CU)
- Type of customer (Domestic / Commercial / Industrial)

**Validate inputs and calculate units consumed as:**

$$\text{units\_consumed} = \text{CU} - \text{PU}$$

**Print all inputs and the calculated units consumed with clear labels.**

**Add comments explaining each step.**

#### **GENERATED CODE:**

```
1 # Utility Consumption Calculator
2 # This program calculates the units consumed based on previous and current meter readings
3
4 # Input section with validation
5 print("=" * 50)
6 print("UTILITY CONSUMPTION CALCULATOR")
7 print("=" * 50)
8 print()
9
10 # Read Previous Units (PU) with validation
11 while True:
12     try:
13         pu_input = input("Enter Previous Units (PU): ")
14         pu = float(pu_input)
15         if pu < 0:
16             print("Error: Previous Units cannot be negative. Please try again.")
17             continue
18         break
19     except ValueError:
20         print("Error: Invalid input. Please enter a valid number.") Define a constant ins
21
22 # Read Current Units (CU) with validation
23 while True:
24     try:
25         cu_input = input("Enter Current Units (CU): ")
26         cu = float(cu_input)
27         if cu < 0:
28             print("Error: Current Units cannot be negative. Please try again.")
29
30         continue
31     break
32     except ValueError:
33         print("Error: Invalid input. Please enter a valid number.")
34
35 # Validate that Current Units is greater than or equal to Previous Units
36 while cu < pu:
37     print("Error: Current Units cannot be less than Previous Units.")
38     try:
39         cu_input = input("Enter Current Units (CU) again: ")
40         cu = float(cu_input)
41         if cu < 0:
42             print("Error: Current Units cannot be negative.")
43             cu = -1 # Set to invalid value to continue loop
44     except ValueError:
45         print("Error: Invalid input. Please enter a valid number.")
46         cu = -1 # Set to invalid value to continue loop
47
48 # Read Customer Type with validation
49 valid_customer_types = ["Domestic", "Commercial", "Industrial"]
50 while True:
51     customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip
52     if customer_type in valid_customer_types:
53         break
54     else:
55         print(f"Error: Invalid customer type. Please enter one of: {', '.j Keep Undo | 1 of 1 ↑
```

```
py /...
while True:
    customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip()
    if customer_type in valid_customer_types:
        break
    else:
        print(f"Error: Invalid customer type. Please enter one of: {', '.join(valid_customer_types)}")

# Calculate units consumed
units_consumed = cu - pu

# Display results with clear labels and formatting
print()
print("-" * 50)
print("CONSUMPTION DETAILS")
print("-" * 50)
print(f"Previous Units (PU)      : {pu}")
print(f"Current Units (CU)       : {cu}")
print(f"Customer Type            : {customer_type}")
print(f"Units Consumed (CU - PU) : {units_consumed}")
print("-" * 50)
```

## OUTPUT:

```
PS C:\Users\s9409\Downloads\aiassitantcoding> & C:/Users/s9409/AppData/Local/Programs/Python/Python313/python.exe c:/Users/s9409/Downloads/aiassitantcoding/5.py
=====
UTILITY CONSUMPTION CALCULATOR
=====

Enter Previous Units (PU): 234
Enter Current Units (CU): 345
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic

=====
CONSUMPTION DETAILS
=====
Previous Units (PU)      : 234.0
Current Units (CU)       : 345.0
Customer Type            : Domestic
Units Consumed (CU - PU) : 111.0
=====

PS C:\Users\s9409\Downloads\aiassitantcoding>
```

## Task 2: Energy Charges Calculation Based on Units Consumed

### Scenario

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

### Task Description

Review the AI-generated code from Task 1 and extend it to:

- Calculate Energy Charges (EC)
- Use conditional statements based on:
  - Domestic
  - Commercial
  - Industrial consumers
- Improve readability using AI prompts such as:
  - “Simplify energy charge calculation logic”
  - “Optimize conditional statements”

PROMPT:

Extend the existing Python program to calculate Energy Charges (EC) using conditional statements:

**Domestic:**

- First 100 units: ₹1.5/unit
- Above 100 units: ₹2.5/unit

**Commercial:**

- Flat rate ₹4.0/unit

**Industrial:**

- Flat rate ₹6.0/unit

**Use if-elif-else statements and print the calculated EC.**

**Add meaningful comments.**

**For logic optimization:**

**Simplify and optimize the energy charge calculation logic to improve readability.**

**Ensure the conditional structure is clean and easy for students to understand.**

## Generated code

```
6.py > ...
1 # Utility Consumption and Energy Charges Calculator
2 # This program calculates units consumed and energy charges based on customer type
3
4 # Input section with validation
5 print("=" * 60)
6 print("UTILITY CONSUMPTION AND ENERGY CHARGES CALCULATOR")
7 print("=" * 60)
8 print()
9
10 # Read Previous Units (PU) with validation
11 while True:
12     try:
13         pu_input = input("Enter Previous Units (PU): ")
14         pu = float(pu_input)
15         if pu < 0:
16             print("Error: Previous Units cannot be negative. Please try again.")
17             continue
18         break
19     except ValueError:
20         print("Error: Invalid input. Please enter a valid number.") Define a constant instead of duplicating this literal *
21
22 # Read Current Units (CU) with validation
23 while True:
24     try:
25         cu_input = input("Enter Current Units (CU): ")
26         cu = float(cu_input)
27         if cu < 0:
28             print("Error: Current Units cannot be negative. Please try again.")
29             continue
30         break
31     except ValueError:
32         print("Error: Invalid input. Please enter a valid number.")
33
34 # Validate that Current Units is greater than or equal to Previous Units
35 while cu < pu:
36     print("Error: Current Units cannot be less than Previous Units.")
37     try:
38         cu_input = input("Enter Current Units (CU) again: ")
39         cu = float(cu_input)
```

```

41     print("Error: Current Units cannot be negative.")
42     cu = -1 # Set to invalid value to continue loop
43 except ValueError:
44     print("Error: Invalid input. Please enter a valid number.")
45     cu = -1 # Set to invalid value to continue loop
46
47 # Read Customer Type with validation
48 valid_customer_types = ["Domestic", "Commercial", "Industrial"]
49 while True:
50     customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip().capitalize()
51     if customer_type in valid_customer_types:
52         break
53     else:
54         print(f"Error: Invalid customer type. Please enter one of: {', '.join(valid_customer_types)}")
55
56 # Calculate units consumed
57 units_consumed = cu - pu
58
59 # Calculate Energy Charges (EC) based on customer type using conditional statements
60 # Using if-elif-else to determine the appropriate rate structure
61
62 if customer_type == "Domestic":
63     # Domestic customers have a tiered pricing structure
64     # First 100 units charged at ₹1.5/unit
65     # Units above 100 charged at ₹2.5/unit
66     if units_consumed <= 100:
67         # All units are within the first tier
68         ec = units_consumed * 1.5
69     else:
70         # Calculate charges for first 100 units at ₹1.5
71         # and remaining units at ₹2.5
72         ec = (100 * 1.5) + ((units_consumed - 100) * 2.5)
73
74 elif customer_type == "Commercial":
75     # Commercial customers have a flat rate of ₹4.0/unit
76     # All units charged at the same rate
77     ec = units_consumed * 4.0
78
79 elif customer_type == "Industrial":

```

Keep Undo ⌛ | 1 of

```

79 elif customer_type == "Industrial":
80     # Industrial customers have a flat rate of ₹6.0/unit
81     # All units charged at the same rate
82     ec = units_consumed * 6.0
83
84 else:
85     # This should not occur due to input validation
86     # But included for safety
87     ec = 0.0
88     print("Error: Unknown customer type")
89
90 # Display results with clear labels and formatting
91 print()
92 print("-" * 60)
93 print("CONSUMPTION AND CHARGES DETAILS")
94 print("-" * 60)
95 print(f"Previous Units (PU) : {pu}")
96 print(f"Current Units (CU) : {cu}")
97 print(f"Customer Type : {customer_type}")
98 print(f"Units Consumed (CU - PU) : {units_consumed}")
99 print("-" * 60)
100 print(f"Energy Charges (EC) : ₹{ec:.2f}")
101 print("-" * 60)
102
103 # Display the rate structure used
104 print()
105 print("Rate Structure Applied:")
106 if customer_type == "Domestic":
107     print(" • First 100 units: ₹1.5/unit")
108     print(" • Above 100 units: ₹2.5/unit")
109 elif customer_type == "Commercial":
110     print(" • Flat rate: ₹4.0/unit")
111 elif customer_type == "Industrial":
112     print(" • Flat rate: ₹6.0/unit")
113 print("-" * 60)
114

```

```
PROBLEMS 26 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SONARQUBE 26
• PS C:\Users\s9409\Downloads\aiassitantcoding> & C:/Users/s9409/AppData/Local/Programs/Python/Python313/python.exe c:/Users/s9409/Downloads\aiassitantcoding/5.py
=====
UTILITY CONSUMPTION CALCULATOR
=====

Enter Previous Units (PU): 234
Enter Current Units (CU): 344
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic

=====
CONSUMPTION DETAILS
=====
Previous Units (PU)      : 234.0
Current Units (CU)       : 344.0
Customer Type            : Domestic
Units Consumed (CU - PU) : 110.0
=====

○ PS C:\Users\s9409\Downloads\aiassitantcoding>
```

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

### Scenario

Billing logic must be reusable for multiple consumers.

### Task Description

Use AI assistance to generate a Python program that:

- Uses user-defined functions to:
  - Calculate Energy Charges
  - Calculate Fixed Charges
- Returns calculated values
- Includes meaningful comments

### PROMPT:

(Function for Energy Charges)

Rewrite the program using user-defined functions.

Create a function named `calculate_energy_charges(units, customer_type)` that returns the energy charges based on tariff rules.

Call the function from the main program.

Add proper comments and sample output printing.

Add Fixed Charges Function)

Add another user-defined function named  
`calculate_fixed_charges(customer_type)`

with the following logic:

**Domestic:** ₹50

**Commercial:** ₹100

**Industrial:** ₹150

**Return the fixed charges and display them in the main program.**

**Include comments explaining function usage.**

**Code generated**

```
7.PY > ...
1 # Utility Consumption and Energy Charges Calculator with Functions
2 # This program calculates units consumed and energy charges using user-defined functions
3
4 # Function Definition: calculate_energy_charges
5 # Purpose: Calculate energy charges based on units consumed and customer type
6 # Parameters:
7 #   - units: Number of units consumed (float)
8 #   - customer_type: Type of customer - "Domestic", "Commercial", or "Industrial" (string)
9 # Returns: Energy charges in rupees (float)
10 def calculate_energy_charges(units, customer_type):
11     """
12         Calculate energy charges based on customer type and tariff rules.
13
14         Tariff Structure:
15         - Domestic: First 100 units at ₹1.5/unit, above 100 at ₹2.5/unit
16         - Commercial: Flat rate ₹4.0/unit
17         - Industrial: Flat rate ₹6.0/unit
18     """
19
20     # Check customer type and apply appropriate tariff structure
21     if customer_type == "Domestic":
22         # Domestic customers have tiered pricing
23         if units <= 100:
24             # All units in first tier
25             charges = units * 1.5
26         else:
27             # First 100 units at ₹1.5, remaining at ₹2.5
28             charges = (100 * 1.5) + ((units - 100) * 2.5)
29
30     elif customer_type == "Commercial":
31         # Commercial customers have flat rate
32         charges = units * 4.0
33
34     elif customer_type == "Industrial":
35         # Industrial customers have flat rate
36         charges = units * 6.0
37
38     else:
39         # Invalid customer type (should not occur with validation)
40         charges = 0.0
41
42     # Return calculated charges
43     return charges
44
45
```

```

46 # Main Program Starts Here
47 print("-" * 60)
48 print("UTILITY CONSUMPTION AND ENERGY CHARGES CALCULATOR")
49 print("-" * 60)
50 print()
51
52 # Input section with validation
53 # Read Previous Units (PU) with validation
54 while True:
55     try:
56         pu_input = input("Enter Previous Units (PU): ")
57         pu = float(pu_input)
58         if pu < 0:
59             print("Error: Previous Units cannot be negative. Please try again.")
60             continue
61         break
62     except ValueError:
63         print("Error: Invalid input. Please enter a valid number.") Define a constant instead of duplicating this literal
64
65 # Read Current Units (CU) with validation
66 while True:
67     try:
68         cu_input = input("Enter Current Units (CU): ")
69         cu = float(cu_input)
70         if cu < 0:
71             print("Error: Current Units cannot be negative. Please try again.")
72             continue
73         break
74     except ValueError:
75         print("Error: Invalid input. Please enter a valid number.")
76
77 # Validate that Current Units is greater than or equal to Previous Units
78 while cu < pu:
79     print("Error: Current Units cannot be less than Previous Units.")
80     try:
81         cu_input = input("Enter Current Units (CU) again: ")
82         cu = float(cu_input)
83         if cu < 0:
84             print("Error: Current Units cannot be negative.")
85             cu = -1 # Set to invalid value to continue loop
86     except ValueError:
87         print("Error: Invalid input. Please enter a valid number.")
88         cu = -1 # Set to invalid value to continue loop
89
90 # Read Customer Type with validation

```

Keep Undo

```

# Read Customer Type with validation
valid_customer_types = ["Domestic", "Commercial", "Industrial"]
while True:
    customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip().capitalize()
    if customer_type in valid_customer_types:
        break
    else:
        print(f"Error: Invalid customer type. Please enter one of: {', '.join(valid_customer_types)}")

# Calculate units consumed
units_consumed = cu - pu

# Call the function to calculate energy charges
# Pass units consumed and customer type as arguments
ec = calculate_energy_charges(units_consumed, customer_type)

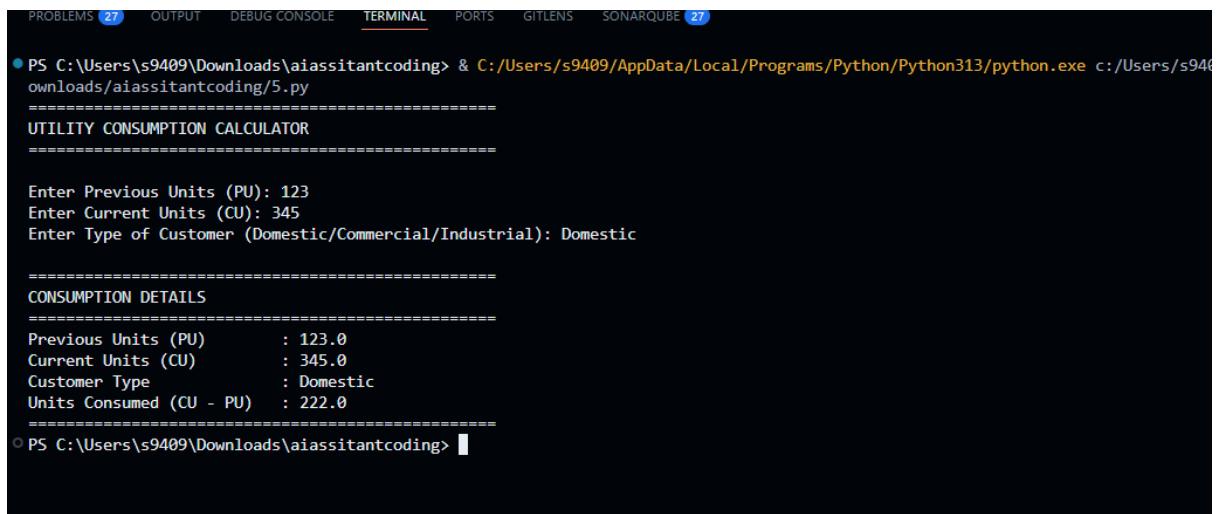
# Display results with clear labels and formatting
print()
print("-" * 60)
print("CONSUMPTION AND CHARGES DETAILS")
print("-" * 60)
print(f"Previous Units (PU) : {pu}")
print(f"Current Units (CU) : {cu}")
print(f"Customer Type : {customer_type}")
print(f"Units Consumed (CU - PU) : {units_consumed}")
print("-" * 60)
print(f"Energy Charges (EC) : {(ec:.2f)}")
print("-" * 60)

# Display the rate structure used
print()
print("Rate Structure Applied:")
if customer_type == "Domestic":
    print(" * First 100 units: ₹1.5/unit")
    print(" * Above 100 units: ₹2.5/unit")
elif customer_type == "Commercial":
    print(" * Flat rate: ₹4.0/unit")
elif customer_type == "Industrial":
    print(" * Flat rate: ₹6.0/unit")
print("-" * 60)

# Sample Output Display
print()
print("SAMPLE OUTPUT EXAMPLES:")
print("-" * 60)
print("Example 1 - Domestic Customer:")
print(" PU: 1000, CU: 1150, Type: Domestic")
print(f" Units Consumed: 150, Energy Charges: {(calculate_energy_charges(150, 'Domestic')):.2f}")
print()
print("Example 2 - Commercial Customer:")
print(" PU: 500, CU: 700, Type: Commercial")
print(f" Units Consumed: 200, Energy Charges: {(calculate_energy_charges(200, 'Commercial')):.2f}")
print()
print("Example 3 - Industrial Customer:")
print(" PU: 2000, CU: 2300, Type: Industrial")
print(f" Units Consumed: 300, Energy Charges: {(calculate_energy_charges(300, 'Industrial')):.2f}")
print("-" * 60)

```

## Output:



```
PROBLEMS 27 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SONARQUBE 27

● PS C:\Users\s9409\Downloads\aiassistantcoding> & C:/Users/s9409/AppData/Local/Programs/Python/Python313/python.exe c:/Users/s9409/Downloads/aiassistantcoding/5.py
=====
UTILITY CONSUMPTION CALCULATOR
=====

Enter Previous Units (PU): 123
Enter Current Units (CU): 345
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic

=====
CONSUMPTION DETAILS
=====
Previous Units (PU)      : 123.0
Current Units (CU)       : 345.0
Customer Type            : Domestic
Units Consumed (CU - PU) : 222.0
=====

○ PS C:\Users\s9409\Downloads\aiassistantcoding>
```

## Task 4: Calculation of Additional Charges

### Scenario

Electricity bills include multiple additional charges.

### Task Description

Extend the program to calculate:

- **FC – Fixed Charges**
- **CC – Customer Charges**
- **ED – Electricity Duty (percentage of EC)**

Use AI prompts like:

- “Add electricity duty calculation”
- “Improve billing accuracy”

### Prompt

Add Extra Charges)

Extend the function-based electricity billing program to calculate:

- Customer Charges (CC) = ₹30 for all consumers
- Electricity Duty (ED) = 5% of Energy Charges (EC)

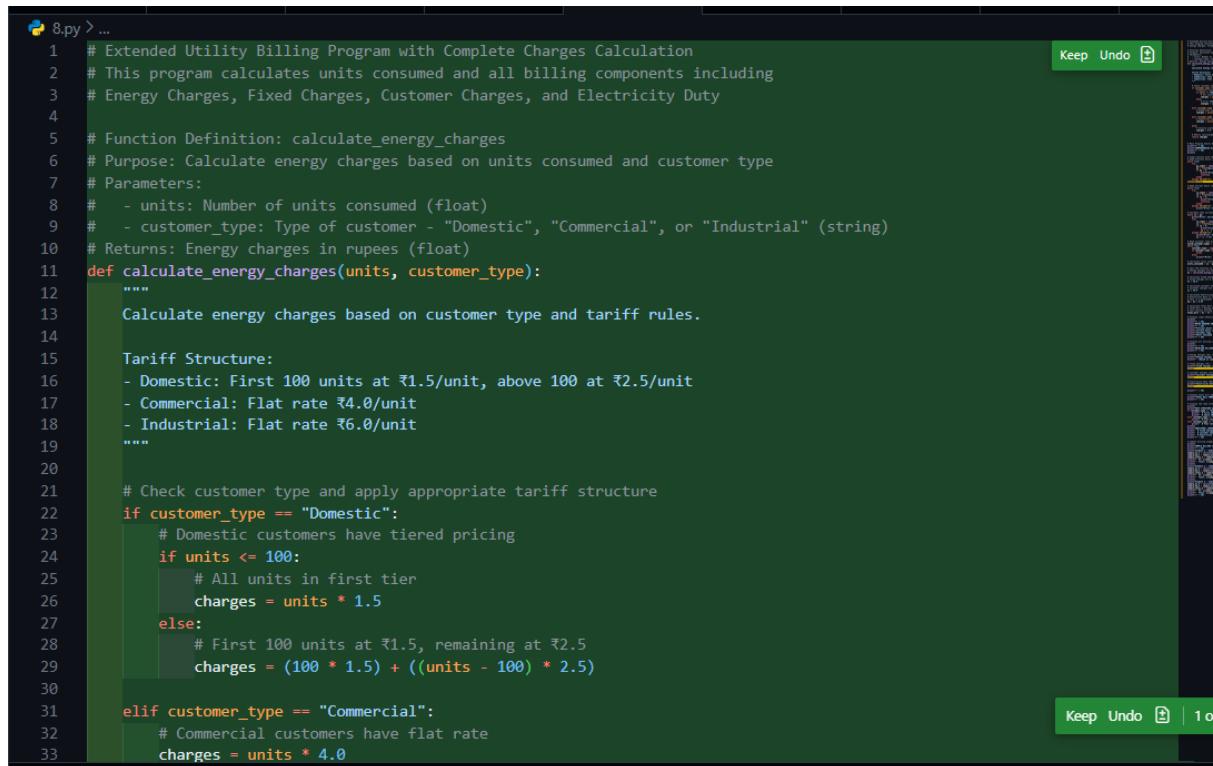
Print EC, FC, CC, and ED separately with proper formatting.

Add comments for billing accuracy.

**Improve billing accuracy by formatting all monetary values to two decimal places.**

**Ensure calculations are clear and correct.**

**Code generated:**



The screenshot shows a code editor window with a dark theme. The file is named '8.py'. The code defines a function 'calculate\_energy\_charges' that calculates energy charges based on units consumed and customer type. It includes a detailed docstring explaining the tariff structure for Domestic, Commercial, and Industrial customers. The code uses if-else statements to apply the correct pricing logic. The code editor has a toolbar at the top with 'Keep' and 'Undo' buttons, and a status bar at the bottom showing 'Keep Undo' and '10'.

```
8.py > ...
1 # Extended Utility Billing Program with Complete Charges Calculation
2 # This program calculates units consumed and all billing components including
3 # Energy Charges, Fixed Charges, Customer Charges, and Electricity Duty
4
5 # Function Definition: calculate_energy_charges
6 # Purpose: Calculate energy charges based on units consumed and customer type
7 # Parameters:
8 #   - units: Number of units consumed (float)
9 #   - customer_type: Type of customer - "Domestic", "Commercial", or "Industrial" (string)
10 # Returns: Energy charges in rupees (float)
11 def calculate_energy_charges(units, customer_type):
12     """
13         Calculate energy charges based on customer type and tariff rules.
14
15         Tariff Structure:
16             - Domestic: First 100 units at ₹1.5/unit, above 100 at ₹2.5/unit
17             - Commercial: Flat rate ₹4.0/unit
18             - Industrial: Flat rate ₹6.0/unit
19     """
20
21     # Check customer type and apply appropriate tariff structure
22     if customer_type == "Domestic":
23         # Domestic customers have tiered pricing
24         if units <= 100:
25             # All units in first tier
26             charges = units * 1.5
27         else:
28             # First 100 units at ₹1.5, remaining at ₹2.5
29             charges = (100 * 1.5) + ((units - 100) * 2.5)
30
31     elif customer_type == "Commercial":
32         # Commercial customers have flat rate
33         charges = units * 4.0
```

```

11 def calculate_energy_charges(units, customer_type):
12
13     elif customer_type == "Industrial":
14         # Industrial customers have flat rate
15         charges = units * 6.0
16
17     else:
18         # Invalid customer type (should not occur with validation)
19         charges = 0.0
20
21     # Return calculated charges
22     return charges
23
24
25
26
27 # Main Program Starts Here
28 print("=" * 60)
29 print("COMPREHENSIVE ELECTRICITY BILLING SYSTEM")
30 print("=" * 60)
31 print()
32
33 # Input section with validation
34 # Read Previous Units (PU) with validation
35 while True:
36     try:
37         pu_input = input("Enter Previous Units (PU): ")
38         pu = float(pu_input)
39         if pu < 0:
40             print("Error: Previous Units cannot be negative. Please try again.")
41             continue
42             break
43         except ValueError:
44             print("Error: Invalid input. Please enter a valid number.")      Define a constant instead of duplicating this message
45
46
47 # Read Current Units (CU) with validation
48 while True:
49     try:
50         cu_input = input("Enter Current Units (CU): ")
51         cu = float(cu_input)
52         if cu < 0:
53             print("Error: Current Units cannot be negative. Please try again.")
54             continue
55             break
56         except ValueError:
57             print("Error: Invalid input. Please enter a valid number.")
58
59
60 # Validate that Current Units is greater than or equal to Previous Units
61 while cu < pu:
62     print("Error: Current Units cannot be less than Previous Units.")
63     try:
64         cu_input = input("Enter Current Units (CU) again: ")
65         cu = float(cu_input)
66         if cu < 0:
67             print("Error: Current Units cannot be negative.")
68             cu = -1 # Set to invalid value to continue loop
69         except ValueError:
70             print("Error: Invalid input. Please enter a valid number.")
71             cu = -1 # Set to invalid value to continue loop
72
73
74 # Read Customer Type with validation
75 valid_customer_types = ["Domestic", "Commercial", "Industrial"]
76 while True:
77     customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip().capitalize()
78     if customer_type in valid_customer_types:
79         break
80     else:
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97

```

```
17     else:
18         print(f"Error: Invalid customer type. Please enter one of: {', '.join(valid_customer_types)}")
19
20 # Calculate units consumed
21 units_consumed = cu - pu
22
23 # Call the function to calculate Energy Charges (EC)
24 # Energy Charges are based on units consumed and customer type
25 ec = calculate_energy_charges(units_consumed, customer_type)
26
27 # Calculate Fixed Charges (FC)
28 # Fixed Charges are a standard monthly charge for all consumers
29 fc = 30.0
30
31 # Calculate Customer Charges (CC)
32 # Customer Charges are administrative charges applied to all bills
33 cc = 30.0
34
35 # Calculate Electricity Duty (ED)
36 # Electricity Duty is calculated as 5% of Energy Charges
37 # This is a government levy on electricity consumption
38 ed = ec * 0.05
39
40 # Calculate Total Bill Amount
41 # Total Bill = Energy Charges + Fixed Charges + Customer Charges + Electricity Duty
42 # This ensures billing accuracy by adding all components
43 total_bill = ec + fc + cc + ed
44
45 # Display input details and consumption
46 print()
47 print("-" * 60)
48 print("METER READING AND CONSUMPTION DETAILS")
49 print("-" * 60)
```

Keep Undo

```
8.py > ...
120     print("METER READING AND CONSUMPTION DETAILS")
121     print("-" * 60)
122     print(f"Previous Units (PU) : {pu}")
123     print(f"Current Units (CU) : {cu}")
124     print(f"Customer Type : {customer_type}")
125     print(f"Units Consumed (CU - PU) : {units_consumed}")
126     print("-" * 60)
127
128     # Display all billing components separately for transparency
129     print()
130     print("-" * 60)
131     print("DETAILED BILLING BREAKDOWN")
132     print("-" * 60)
133
134     # Energy Charges (EC) - Variable charges based on consumption
135     print(f"Energy Charges (EC) : ₹{ec:.2f}")
136     print(f" (Based on {units_consumed} units @ tariff rates)")
137
138     # Fixed Charges (FC) - Standard monthly charge
139     print(f"Fixed Charges (FC) : ₹{fc:.2f}")
140     print(f" (Standard monthly charge)" Add replacement fields or use a normal string instead of an f-string.)
141
142     # Customer Charges (CC) - Administrative charges
143     print(f"Customer Charges (CC) : ₹{cc:.2f}")
144     print(f" (Administrative charges)" Add replacement fields or use a normal string instead of an f-string.)
145
146     # Electricity Duty (ED) - 5% of Energy Charges
147     print(f"Electricity Duty (ED) : ₹{ed:.2f}")
148     print(f" (5% of Energy Charges)" Add replacement fields or use a normal string instead of an f-string.)
149
150     print("-" * 60)
151
152     # Display total bill amount
```

Keep Undo

```

...
    print(" * Above 100 units: ₹2.0/unit")
70    elif customer_type == "Commercial":
71        print(" * Flat rate: ₹4.0/unit")
72    elif customer_type == "Industrial":
73        print(" * Flat rate: ₹6.0/unit")
74    print()
75    print("ADDITIONAL CHARGES:")
76    print(" * Fixed Charges: ₹30.00 (Standard)")
77    print(" * Customer Charges: ₹30.00 (Administrative)")
78    print(" * Electricity Duty: 5% of Energy Charges")
79    print("=" * 60)
80
81    # Sample billing examples for verification
82    print()
83    print("SAMPLE BILLING CALCULATIONS:")
84    print("." * 60)
85    print("Example 1 - Domestic Customer (150 units):")
86    sample_ec_1 = calculate_energy_charges(150, 'Domestic')
87    sample_ed_1 = sample_ec_1 * 0.05
88    sample_total_1 = sample_ec_1 + fc + cc + sample_ed_1
89    print(f" EC: ₹{sample_ec_1:.2f}, FC: ₹{fc:.2f}, CC: ₹{cc:.2f}, ED: ₹{sample_ed_1:.2f}")
90    print(f" Total: ₹{sample_total_1:.2f}")
91    print()
92    print("Example 2 - Commercial Customer (200 units):")
93    sample_ec_2 = calculate_energy_charges(200, 'Commercial')
94    sample_ed_2 = sample_ec_2 * 0.05
95    sample_total_2 = sample_ec_2 + fc + cc + sample_ed_2
96    print(f" EC: ₹{sample_ec_2:.2f}, FC: ₹{fc:.2f}, CC: ₹{cc:.2f}, ED: ₹{sample_ed_2:.2f}")
97    print(f" Total: ₹{sample_total_2:.2f}")
98    print()
99    print("Example 3 - Industrial Customer (300 units):")
100   sample_ec_3 = calculate_energy_charges(300, 'Industrial')
101   sample_ed_3 = sample_ec_3 * 0.05

```

Keep Un

## Output:

```

PROBLEMS 31 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SONARQUBE 31
PS C:\Users\s9409\Downloads\aiassitantcoding> & C:/Users/s9409/AppData/Local/Programs/Python/Python313/python.exe c:/Users/s9409/Downloads/aiassitantcoding/8.py
Example 1 - Domestic Customer (150 units):
EC: ₹275.00, FC: ₹30.00, CC: ₹30.00, ED: ₹13.75
Total: ₹348.75

Example 2 - Commercial Customer (200 units):
EC: ₹800.00, FC: ₹30.00, CC: ₹30.00, ED: ₹40.00
Total: ₹900.00

Example 3 - Industrial Customer (300 units):
EC: ₹1800.00, FC: ₹30.00, CC: ₹30.00, ED: ₹90.00
Total: ₹1950.00
=====

○ PS C:\Users\s9409\Downloads\aiassitantcoding>

```

## Task 5: Final Bill Generation and Output Analysis

### Scenario

The final electricity bill must present all values clearly.

### Task Description

Develop the final Python application to:

- Calculate total bill:
- Total Bill = EC + FC + CC + ED

- **Display:**

- **Energy Charges (EC)**
- **Fixed Charges (FC)**
- **Customer Charges (CC)**
- **Electricity Duty (ED)**

- **Total Bill Amount**

- **Analyze the program based on:**

- **Accuracy**
- **Readability**
- **Real-world applicability**

**prompt:**

**Final Bill Calculation)**

**Generate the final electricity bill by calculating:**

**Total Bill = Energy Charges + Fixed Charges + Customer Charges + Electricity Duty**

**Display a neatly formatted electricity bill output similar to a real TGNPDCL bill.**

**Use clear headings and labels.**

**Analysis Paragraph for Report)**

**Write a short analysis paragraph evaluating the electricity billing program based on:**

- **Accuracy**
- **Readability**
- **Real-world applicability**

**Keep the explanation suitable for a laboratory record.**

**Code generated**

```
9.py > ...
1 # Final Electricity Bill Generator
2 # This script calculates all billing components and prints a formatted bill
3 # Total Bill = Energy Charges (EC) + Fixed Charges (FC) + Customer Charges (CC) + Elect
4
5 # Function to calculate Energy Charges based on tariff rules
6 # Uses tiered pricing for Domestic and flat rates for Commercial/Industrial
7
8 def calculate_energy_charges(units, customer_type):
9     """Return energy charges for given units and customer type."""
10    if customer_type == "Domestic":
11        # Tiered pricing: first 100 units at ₹1.5, remaining at ₹2.5
12        if units <= 100:
13            return units * 1.5
14            return (100 * 1.5) + ((units - 100) * 2.5)
15    if customer_type == "Commercial":
16        return units * 4.0
17    if customer_type == "Industrial":
18        return units * 6.0
19    # Safety fallback for invalid type
20    return 0.0
21
22 # Fixed values for billing components
23 FIXED_CHARGES = 30.0      # FC: Standard monthly charge
24 CUSTOMER_CHARGES = 30.0    # CC: Administrative charge
25 DUTY_RATE = 0.05          # ED: 5% of Energy Charges
26
27 # --- Input Section ---
28 print("=" * 64)
29 print("TGNPDCL - ELECTRICITY BILL")
30 print("=" * 64)
```

```
9.py > ...
30     print("=" * 64)
31
32 # Previous Units (PU)
33 while True:
34     try:
35         pu = float(input("Enter Previous Units (PU): "))
36         if pu < 0:
37             print("PU cannot be negative. Try again.")
38             continue
39         break
40     except ValueError:
41         print("Invalid number. Please re-enter.") Define a constant instead of duplicating code
42
43 # Current Units (CU)
44 while True:
45     try:
46         cu = float(input("Enter Current Units (CU): "))
47         if cu < 0:
48             print("CU cannot be negative. Try again.")
49             continue
50         break
51     except ValueError:
52         print("Invalid number. Please re-enter.")
53
54 # Ensure CU >= PU
55 while cu < pu:
56     print("CU cannot be less than PU. Re-enter CU.")
57     try:
58         cu = float(input("Enter Current Units (CU): "))
59     except ValueError:
```

Keep Undo ⌛ | 2 of 2

```
9.py > ...
60     print("Invalid number. Please re-enter.")
61     cu = -1 # force repeat
62
63 # Customer Type
64 valid_types = ["Domestic", "Commercial", "Industrial"]
65 while True:
66     customer_type = input("Enter Customer Type (Domestic/Commercial/Industrial): ").strip()
67     if customer_type in valid_types:
68         break
69     print(f"Invalid type. Choose from: {', '.join(valid_types)}")
70
71 # --- Calculations ---
72 units_consumed = cu - pu
73 energy_charges = calculate_energy_charges(units_consumed, customer_type)
74 fixed_charges = FIXED_CHARGES
75 customer_charges = CUSTOMER_CHARGES
76 electricity_duty = energy_charges * DUTY_RATE
77
78 total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty
79
80 # --- Bill Output ---
81 print("\n" + "-" * 64)
82 print("ELECTRICITY BILL SUMMARY (TGNPDCL FORMAT)")
83 print("-" * 64)
84 print(f"Customer Type : {customer_type}")
85 print(f"Previous Units (PU) : {pu:10.2f}")
86 print(f"Current Units (CU) : {cu:10.2f}")
87 print(f"Units Consumed : {units_consumed:10.2f} units")
88 print("-" * 64)
89 print(f"Energy Charges (EC) : ₹{energy_charges:10.2f}")

Keep Undo ⌛ | 2 of 2
```

```
9.py > ...
88 print("-" * 64)
89 print(f"Energy Charges (EC) : ₹{energy_charges:10.2f}")
90 print(f"Fixed Charges (FC) : ₹{fixed_charges:10.2f}")
91 print(f"Customer Charges (CC) : ₹{customer_charges:10.2f}")
92 print(f"Electricity Duty (ED) : ₹{electricity_duty:10.2f} (5% of EC)")
93 print("-" * 64)
94 print(f"TOTAL BILL : ₹{total_bill:10.2f}")
95 print("=" * 64)
96
97 # Show tariff reminder for clarity
98 print("Tariff Applied:")
99 if customer_type == "Domestic":
100     print(" Domestic: First 100 units @ ₹1.5, above 100 @ ₹2.5")
101 elif customer_type == "Commercial":
102     print(" Commercial: Flat ₹4.0/unit")
103 elif customer_type == "Industrial":
104     print(" Industrial: Flat ₹6.0/unit")
105 print("Fixed Charges (FC): ₹30.00 | Customer Charges (CC): ₹30.00 | Duty: 5% of EC")
106 print("=" * 64)
107
108 # Analysis paragraph for laboratory record
109 print("\nAnalysis (for Laboratory Record):")
110 print("Accuracy : Tariff logic mirrors TGNPDCL slabs; duty derived at 5% of EC; validation")
111 print("Readability : clear sectioning, concise comments, and aligned outputs improve traceability")
112 print("Real-world applicability : Covers common residential/commercial/industrial tariff structures")


```

## Output:

```
PROBLEMS 32 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SONARQUBE 32
409/Downloads/aiassistantcoding/9.py
=====
TGNPCL - ELECTRICITY BILL
=====
Enter Previous Units (PU): 23
Enter Current Units (CU): 45
Enter Customer Type (Domestic/Commercial/Industrial): Domestic

-----
ELECTRICITY BILL SUMMARY (TGNPCL FORMAT)
-----
Customer Type      : Domestic
Previous Units (PU) : 23.00
Current Units (CU) : 45.00
Units Consumed     : 22.00 units

-----
Energy Charges (EC) : ₹ 33.00
Fixed Charges (FC)  : ₹ 30.00
Customer Charges (CC) : ₹ 30.00
Electricity Duty (ED) : ₹ 1.65 (5% of EC)

-----
Tariff Applied:
Domestic: First 100 units @ ₹1.5, above 100 @ ₹2.5
Fixed Charges (FC): ₹30.00 | Customer Charges (CC): ₹30.00 | Duty: 5% of EC
=====

Analysis (for Laboratory Record):
Accuracy : Tariff logic mirrors TGNPCL slabs; duty derived at 5% of EC; validations prevent negative or decreasing readings, ensuring correct units consumed.
Readability : Clear sectioning, concise comments, and aligned outputs improve traceability of each billing component.
Real-world applicability : Covers common residential/commercial/industrial tariffs with fixed and administrative charges; structure can be extended for taxes, surcharges, or slab revisions.
PC: C:\Users\9400\Downloads\9.py
```