

ASSIGNMENT-3.3

2303A51546

BATCH-10

❖ TASK-1:

PROMPT:

Write a Python program to calculate the electricity bill using previous and current meter readings.

The program should ask for customer type (Domestic, Commercial, or Industrial).

Calculate units consumed and apply different rates based on unit slabs and customer type. Finally, display units consumed and total bill amount, and show an error for invalid input.

CODE:

```
previous_units = float(input("Enter Previous Units (PU): "))
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer (Domestic/Commercial):
").strip().lower()
# Calculate units consumed
units_consumed = current_units - previous_units
# Initialize bill amount
bill_amount = 0.0
# Calculate bill based on customer type
if customer_type == "domestic":
    if units_consumed <= 100:
        bill_amount = units_consumed * 1.5
    elif units_consumed <= 300:
        bill_amount = (100 * 1.5) + (units_consumed - 100) * 2.5
    else:
        bill_amount = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
elif customer_type == "commercial":
    if units_consumed <= 100:
        bill_amount = units_consumed * 2.0
    elif units_consumed <= 300:
        bill_amount = (100 * 2.0) + (units_consumed - 100) * 4.0
    else:
        bill_amount = (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
else:
    print("Invalid customer type. Please enter 'Domestic' or 'Commercial'.")
```

```

    exit()

# Display the bill amount
print(f"Units Consumed: {units_consumed}")
print(f"Bill Amount: ${bill_amount:.2f}")

OUTPUT:

```

```

day3.py
1 #!/usr/bin/python3
2
3 # Calculate units consumed
4 units_consumed = current_units - previous_units
5
6 # Initialize bill amount
7 bill_amount = 0.0
8
9 # Calculate bill based on customer type
10 if customer_type == "domestic":
11     if units_consumed <= 100:
12         bill_amount = units_consumed * 1.5
13     elif units_consumed <= 300:
14         bill_amount = (100 * 1.5) + (units_consumed - 100) * 2.5
15     else:
16         bill_amount = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
17 elif customer_type == "commercial":
18     if units_consumed <= 100:
19         bill_amount = units_consumed * 2.0
20     elif units_consumed <= 300:
21         bill_amount = (100 * 2.0) + (units_consumed - 100) * 4.0
22     else:
23         bill_amount = (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
24 else:
25     print("Invalid customer type. Please enter 'Domestic' or 'Commercial'.")
26     exit()
27
28 # Display the bill amount
29 print(f"Units Consumed: {units_consumed}")
30 print(f"Bill Amount: ${bill_amount:.2f}")
31
32
33
34
35
36
37
38
39
40

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Enter Previous Units (PU): 45
Enter Current Units (CU): 50
Enter Type of Customer (Domestic/Commercial): DOMESTIC
Units Consumed: 5.0
Bill Amount: \$7.50
PS C:\Users\sriva\OneDrive\Desktop\AI Assisted>

EXPLANATION:

This task focuses on collecting correct consumer details required for electricity billing.

The program reads previous units, current units, and consumer type from the user.

Units consumed are calculated using meter readings, which is the base for billing.

This step ensures accurate input handling for further calculations.

❖ TASK-2

PROMPT:

Create a Python program that calculates electricity charges depending on how many units a customer used and their category (Domestic, Commercial, Industrial).

Use conditional statements to apply different tariff rates.

Print the total bill clearly and explain how the calculation works.

CODE:

```
previous_units = float(input("Enter Previous Units (PU): "))
```

```

current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer
(Domestic/Commercial/Industrial): ").strip().lower()
# Calculate units consumed
units_consumed = current_units - previous_units
# Initialize bill amount
bill_amount = 0.0
# Calculate bill based on customer type
if customer_type == "domestic":
    if units_consumed <= 100:
        bill_amount = units_consumed * 1.5
    elif units_consumed <= 300:
        bill_amount = (100 * 1.5) + (units_consumed - 100) * 2.5
    else:
        bill_amount = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
elif customer_type == "commercial":
    if units_consumed <= 100:
        bill_amount = units_consumed * 2.0
    elif units_consumed <= 300:
        bill_amount = (100 * 2.0) + (units_consumed - 100) * 4.0
    else:
        bill_amount = (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
elif customer_type == "industrial":
    if units_consumed <= 100:
        bill_amount = units_consumed * 2.5
    elif units_consumed <= 300:
        bill_amount = (100 * 2.5) + (units_consumed - 100) * 5.0
    else:
        bill_amount = (100 * 2.5) + (200 * 5.0) + (units_consumed - 300) * 7.0
else:
    print("Invalid customer type. Please enter 'Domestic', 'Commercial', or
'Industrial'.")
    exit()
# Display the bill amount
print(f"Units Consumed: {units_consumed}")
print(f"Bill Amount: ${bill_amount:.2f}")

```

OUTPUT:

```
File Edit Selection View Go Run Terminal Help ← → Q AI Assisted
day1.py day2.py day3.py x
day3.py >...
units_consumed = current_units - previous_units
# initialize bill amount
bill_amount = 0.0
# calculate bill based on customer type
if customer_type == "domestic":
    if units_consumed <= 100:
        bill_amount = units_consumed * 1.5
    elif units_consumed <= 300:
        bill_amount = (100 * 1.5) + (units_consumed - 100) * 2.5
    else:
        bill_amount = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
elif customer_type == "commercial":
    if units_consumed <= 100:
        bill_amount = units_consumed * 2.0
    elif units_consumed <= 300:
        bill_amount = (100 * 2.0) + (units_consumed - 100) * 4.0
    else:
        bill_amount = (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
elif customer_type == "industrial":
    if units_consumed <= 100:
        bill_amount = units_consumed * 2.5
    elif units_consumed <= 300:
        bill_amount = (100 * 2.5) + (units_consumed - 100) * 5.0
    else:
        bill_amount = (100 * 2.5) + (200 * 5.0) + (units_consumed - 300) * 7.0
else:
    print("Invalid customer type. Please enter 'Domestic', 'Commercial', or 'Industrial'.")
exit()
# display the bill amount
```

EXPLANATION:

In this task, energy charges are calculated based on units consumed and consumer type.

Conditional logic is used to apply different tariff rates for domestic, commercial, and industrial users.

TASK-3:

PROMPT:

Write a Python electricity billing program using functions.

Create separate functions to calculate Energy Charges and Fixed Charges based on customer type.

Take inputs, compute units consumed, and display all charges with proper labels.

Add comments, sample input/output, and at least three test cases.

CODE:

```
def calculate_energy_charges(units_consumed, customer_type):
    if customer_type == "domestic":
        if units_consumed <= 100:
            return units_consumed * 1.5
        elif units_consumed <= 300:
            return (100 * 1.5) + (units_consumed - 100) * 2.5
        else:
            return (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
    elif customer_type == "commercial":
        if units_consumed <= 100:
            return units_consumed * 2.0
        elif units_consumed <= 300:
```

```

        return (100 * 2.0) + (units_consumed - 100) * 4.0
    else:
        return (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
elif customer_type == "industrial":
    if units_consumed <= 100:
        return units_consumed * 2.5
    elif units_consumed <= 300:
        return (100 * 2.5) + (units_consumed - 100) * 5.0
    else:
        return (100 * 2.5) + (200 * 5.0) + (units_consumed - 300) * 7.0
else:
    return None

# Function to calculate Fixed Charges
def calculate_fixed_charges(customer_type):
    if customer_type == "domestic":
        return 50.0
    elif customer_type == "commercial":
        return 100.0
    elif customer_type == "industrial":
        return 150.0
    else:
        return None

# Main program
previous_units = float(input("Enter Previous Units (PU): "))
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer\n(Domestic/Commercial/Industrial): ").strip().lower()

```

```
units_consumed = current_units - previous_units
energy_charges = calculate_energy_charges(units_consumed,
customer_type)
fixed_charges = calculate_fixed_charges(customer_type)
if energy_charges is None or fixed_charges is None:
    print("Invalid customer type. Please enter 'Domestic', 'Commercial', or
'Industrial'.")
    exit()
total_bill = energy_charges + fixed_charges
# Display the bill details
print(f"Units Consumed: {units_consumed}")
print(f"Energy Charges: ${energy_charges:.2f}")
print(f"Fixed Charges: ${fixed_charges:.2f}")
print(f"Total Bill Amount: ${total_bill:.2f}")
```

OUTPUT:

The screenshot shows a Python code editor with a dark theme. The main pane displays a script named 'day3.py' containing logic to calculate fixed charges for different customer types (Domestic, Commercial, Industrial) based on previous and current units consumed. The code includes error handling for invalid input and prints the total bill amount.

```
File Edit Selection View Go Run Terminal Help ← → Q AI Assisted
day1.py day2.py day3.py ×
day3.py?_ calculate_fixed_charges(customer_type):
    customer_type == "commercial":
        return 100.0
    elif customer_type == "industrial":
        return 150.0
    else:
        return None
# Main program
previous_units = float(input("Enter Previous Units (PU): "))
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter type of Customer (Domestic/Commercial/Industrial): ").strip().lower()
units_consumed = current_units - previous_units
energy_consumed = calculate_energy_charges(units_consumed, customer_type)
fixed_charges = calculate_fixed_charges(customer_type)
if energy_consumed is None or fixed_charges is None:
    print("Invalid customer type. Please enter 'Domestic', 'Commercial', or 'Industrial'.")
    exit()
total_bill = energy_consumed + fixed_charges
# display the bill details
print(f"Units consumed: {units_consumed}")
print(f"Energy Charges: ${energy_consumed:.2f}")
print(f"Fixed Charges: ${fixed_charges:.2f}")
print(f"Total Bill Amount: ${total_bill:.2f}")

```

EXPLANATION:

introduces modular programming using user-defined methods.

Separate methods are used to calculate energy charges and fixed charges.

This makes the program reusable and easier to maintain.

Modular design improves code structure and readability.

❖ **TASK-4:**

PROMPT:

Extend the electricity billing program by adding more charges like Fixed Charges and Electricity Duty.
Electricity Duty should be calculated as a percentage of Energy Charges.
Display each charge separately and make the output clear and easy to read.
Ensure all calculations are correct.

CODE:

```
def calculate_electricity_duty(energy_charges, duty_percentage):  
    return (energy_charges * duty_percentage) / 100
```

```
# Main program  
previous_units = float(input("Enter Previous Units (PU): "))  
current_units = float(input("Enter Current Units (CU): "))  
customer_type = input("Enter Type of Customer  
(Domestic/Commercial/Industrial): ").strip().lower()  
units_consumed = current_units - previous_units  
energy_charges = calculate_energy_charges(units_consumed,  
customer_type)  
fixed_charges = calculate_fixed_charges(customer_type)  
duty_percentage = 5.0 # Example duty percentage  
electricity_duty = calculate_electricity_duty(energy_charges,  
duty_percentage)  
if energy_charges is None or fixed_charges is None:  
    print("Invalid customer type. Please enter 'Domestic', 'Commercial',  
or 'Industrial'.")  
    exit()  
total_bill = energy_charges + fixed_charges + electricity_duty  
# Display the bill details  
print(f"Units Consumed: {units_consumed}")  
print(f"Energy Charges: ${energy_charges:.2f}")
```

```
print(f"Fixed Charges: ${fixed_charges:.2f}")  
print(f"Electricity Duty (@{duty_percentage}%)  
${electricity_duty:.2f}")
```

```
print(f"Total Bill Amount: ${total_bill:.2f}")
```

OUTPUT:

```

day3.py > ...
175 # Print each charge separately with clear labels.
176 # Ensure billing accuracy and verify intermediate results.
177 # Improve output formatting for better clarity.
178 # Function to calculate Electricity Duty
179 def calculate_electricity_duty(energy_charges, duty_percentage):
180     return (energy_charges * duty_percentage) / 100
181
182 # Main program
183 previous_units = float(input("Enter Previous Units (PU): "))
184 current_units = float(input("Enter Current Units (CU): "))
185 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip().lower()
186 units_consumed = current_units - previous_units
187 energy_charges = calculate_energy_charges(units_consumed, customer_type)
188 fixed_charges = calculate_fixed_charges(customer_type)
189 duty_percentage = 5.0 # Example duty percentage
190 electricity_duty = calculate_electricity_duty(energy_charges, duty_percentage)
191 if energy_charge is None or fixed_charges is None:
192     print("Invalid customer type. Please enter 'Domestic', 'Commercial', or 'Industrial'.")
193     exit()
194 total_bill = energy_charge + fixed_charges + electricity_duty
195 # Display the bill details
196 print("Units Consumed: {units_consumed}")
197 print("Energy Charges: ${energy_charges:.2f}")
198 print("Fixed Charges: ${fixed_charges:.2f}")
199 print("Electricity Duty (@duty_percentage): ${electricity_duty:.2f}")
200

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Enter Previous Units (PU): 45
Enter Current Units (CU): 84
Enter Type of Customer (Domestic/Commercial/Industrial): INDUSTRIAL
Units Consumed: 39.0
Energy Charges: \$150.00
Fixed Charges: \$150.00
Electricity Duty (@5.00): \$4.88
Total Bill Amount: \$252.88
Ps C:\Users\sriva\OneDrive\Desktop\AI Assisted>

EXPLANATION:

This task extends billing by adding additional charges like fixed charges, customer charges, and electricity duty.

Electricity duty is calculated as a percentage of energy charges.

Printing individual charges helps verify calculation accuracy.

This step makes the bill more realistic and detailed.

❖ TASK-5:

PROMPT:

Create a complete electricity billing system in Python.

Calculate Energy Charges (EC), Fixed Charges (FC), Customer Charges (CC), and Electricity Duty (ED).

Find the total bill using all these values.

Display the final result in a neat format that looks like a real electricity bill.

CODE:

```

def calculate_customer_charges(customer_type):
    if customer_type == "domestic":
        return 20.0

```

```

        elif customer_type == "commercial":
            return 50.0
        elif customer_type == "industrial":
            return 80.0
        else:
            return None
    # Main program
    previous_units = float(input("Enter Previous Units (PU): "))
    current_units = float(input("Enter Current Units (CU): "))
    customer_type = input("Enter Type of Customer\n(Domestic/Commercial/Industrial): ").strip().lower()
    units_consumed = current_units - previous_units
    energy_charges = calculate_energy_charges(units_consumed,
                                                customer_type)
    fixed_charges = calculate_fixed_charges(customer_type)
    customer_charges = calculate_customer_charges(customer_type)
    duty_percentage = 5.0 # Example duty percentage
    electricity_duty = calculate_electricity_duty(energy_charges,
                                                   duty_percentage)
    if (energy_charges is None or fixed_charges is None or
        customer_charges is None):
        print("Invalid customer type. Please enter 'Domestic', 'Commercial',
or 'Industrial'.")
        exit()
    total_bill = (energy_charges + fixed_charges +
                  customer_charges + electricity_duty)
    # Display the bill details
    print("----- Electricity Bill ---")
    print(f"Units Consumed: {units_consumed}")
    print(f"Energy Charges (EC): ${energy_charges:.2f}")
    print(f"Fixed Charges (FC): ${fixed_charges:.2f}")
    print(f"Customer Charges (CC): ${customer_charges:.2f}")
    print(f"Electricity Duty (ED) @{duty_percentage}%:\n${electricity_duty:.2f}")
    print(f"Total Bill Amount: ${total_bill:.2f}")

```

OUTPUT:

The screenshot shows a terminal window with Python code for calculating electricity bills. The code defines a function `calculate_customer_charges` that takes a customer type as input and returns the total bill amount. It prompts the user for previous units, current units, and customer type, then calculates energy charges, fixed charges, and electricity duty. The total bill is the sum of these components. The output is displayed in a neat, bill-like format.

```
File Edit Selection View Go Run Terminal Help ← → Q, AI Assisted
day1.py day2.py day3.py x
day3.py > ...
219 def calculate_customer_charges(customer_type):
220     elif customer_type == "Industrial":
221         return 80.0
222     else:
223         return None
224
225     # Main program
226     previous_units = float(input("Enter Previous Units (PU): "))
227     current_units = float(input("Enter Current Units (CU): "))
228     customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip().lower()
229     units_consumed = current_units - previous_units
230     energy_charges = calculate_energy_charges(units_consumed, customer_type)
231     fixed_charges = calculate_fixed_charges(customer_type)
232     customer_charges = calculate_customer_charges(customer_type)
233     electricity_duty = calculate_electricity_duty(energy_charges, duty_percentage)
234
235     if (energy_charges is None) or (fixed_charges is None):
236         print("Invalid customer type. Please enter 'Domestic', 'Commercial', or 'Industrial'.")
237         exit()
238     total_bill = (energy_charges + fixed_charges +
239                  customer_charges + electricity_duty)
240
241     # Display the bill details
242     print("----- Electricity Bill -----")
243     print(f"Units Consumed: {units_consumed}")
244     print(f"Energy Charges (EC): ${energy_charges:.2f}")
245     print(f"Fixed Charges (FC): ${fixed_charges:.2f}")
246
247     print(f"Total Bill Amount: ${total_bill:.2f}")
248
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Enter Type of Customer (Domestic/Commercial/Industrial): commercial
----- Electricity Bill -----
Units Consumed: 15.0
Energy Charge (EC): $30.00
Fixed Charges (FC): $100.00
Customer Charges (CC): $50.00
Electricity Duty (ED): $5.00: $1.50
Total Bill Amount: $181.50
PS C:\Users\Urvi\OneDrive\Desktop\AI Assisted>
20°C Sunny
202, Col 1 Spaces: 4 UTF-8 CRLF Python 3.13.7 (venv) ENG IN 21+
```

EXPLANATION:

Task 5 generates the final electricity bill by combining all charge components.

The total bill amount is calculated by adding EC, FC, CC, and ED.

The output is displayed in a neat, bill-like format for clarity.

This task demonstrates a complete, real-world electricity billing application