

ASSIGNMENT-3.3

2303A51343

BATCH-10

TASK-1:

PROMPT:

Generate a simple Python program that reads Previous Units (PU), Current Units (CU), and Type of Customer from the user.

Calculate the units consumed as CU – PU.

Implement all logic directly in the main program without using functions.

Display the units consumed clearly with proper print statements.

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

```

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:

Write a Python program for an electricity billing system that reads the previous meter reading, current meter reading, and the type of customer, and then calculates the total units consumed. Extend the program to calculate the Energy Charges (EC) based on the category of customer, namely domestic, commercial, and industrial. The program should use meaningful variable names, display output using clean and clear print statements, and properly handle customer type input to ensure correct bill

CODE:

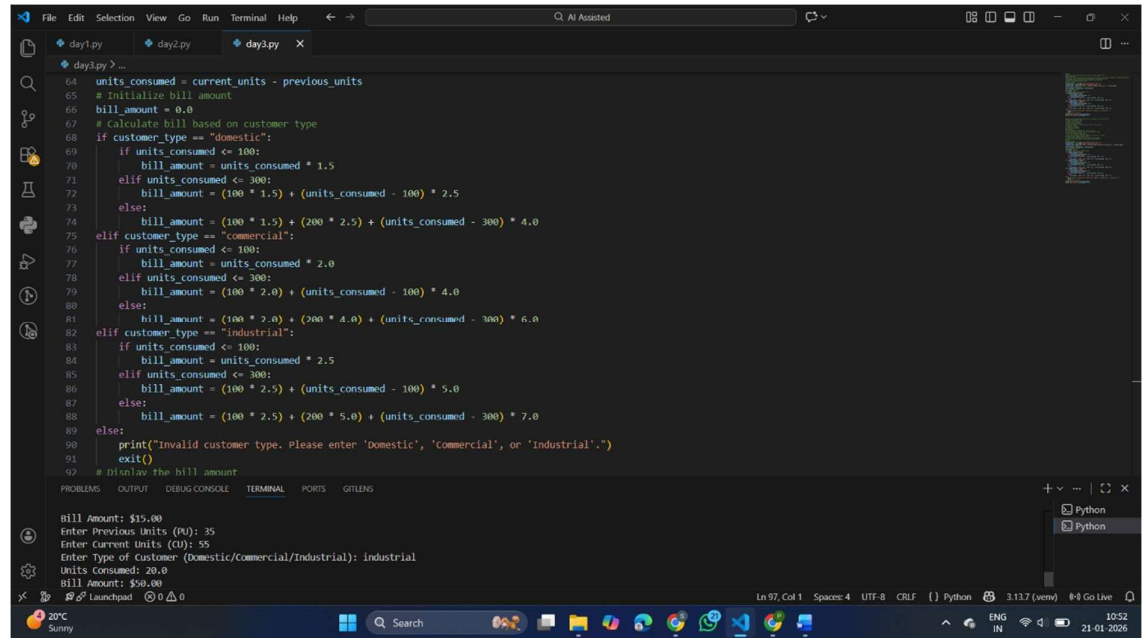
```
previous_units = float(input("Enter Previous Units (PU): "))
```

```

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:



```
64 units_consumed = current_units - previous_units
65 # Initialize bill amount
66 bill_amount = 0.0
67 # calculate bill based on customer type
68 if customer_type == "domestic":
69     if units_consumed <= 100:
70         bill_amount = units_consumed * 1.5
71     elif units_consumed <= 300:
72         bill_amount = (100 * 1.5) + (units_consumed - 100) * 2.5
73     else:
74         bill_amount = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
75 elif customer_type == "commercial":
76     if units_consumed <= 100:
77         bill_amount = units_consumed * 2.0
78     elif units_consumed <= 300:
79         bill_amount = (100 * 2.0) + (units_consumed - 100) * 4.0
80     else:
81         bill_amount = (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
82 elif customer_type == "industrial":
83     if units_consumed <= 100:
84         bill_amount = units_consumed * 2.5
85     elif units_consumed <= 300:
86         bill_amount = (100 * 2.5) + (units_consumed - 100) * 5.0
87     else:
88         bill_amount = (100 * 2.5) + (200 * 5.0) + (units_consumed - 300) * 7.0
89 else:
90     print("Invalid customer type, Please enter 'Domestic', 'Commercial', or 'Industrial'.")
91     exit()
92 # Display the bill amount
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
Bill Amount: $15.00
Enter Previous Units (PU): 35
Enter Current Units (CU): 55
Enter Type of Customer (Domestic/Commercial/Industrial): Industrial
Units Consumed: 20.0
Bill Amount: $50.00
```

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:

Generate a Python program for an electricity billing system using a modular design.

Requirements:

Create a function to calculate Energy Charges (EC):

- Accept units consumed and customer type as parameters.
- Use conditional logic for Domestic, Commercial, and Industrial customers.
- Return the calculated Energy Charges.

- Create a function to calculate Fixed Charges (FC):

- Accept customer type as a parameter.
- Return fixed charges based on customer category.

- In the main program:

- Read Previous Units (PU), Current Units (CU), and Type of Customer.
- Calculate units consumed.
- Call the Energy Charges and Fixed Charges functions.
- Display all calculated values clearly.

Function to calculate Energy Charges

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
(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 VS Code editor with a Python file named `day3.py`. The code defines a function `calculate_fixed_charges` and a main program that takes user input for previous units, current units, and customer type. It then calculates energy charges, fixed charges, and the total bill amount. The terminal output shows the program running with the following inputs: Previous Units (PU): 52, Current Units (CU): 62, and Customer Type: Industrial. The output displays: Units Consumed: 10, Energy Charges: \$25.00, Fixed Charges: \$150.00, and Total Bill Amount: \$175.00.

```

def calculate_fixed_charges(customer_type):
    if 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_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}")

```

```

Enter Previous Units (PU): 52
Enter Current Units (CU): 62
Enter Type of Customer (Domestic/Commercial/Industrial): Industrial
Units Consumed: 10.0
Energy Charges: $25.00
Fixed Charges: $150.00
Total Bill Amount: $175.00
PS C:\Users\sriva\OneDrive\Desktop\AI Assisted>

```

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 existing Python electricity billing program to include additional charges.

Calculate and display:

- FC - Fixed Charges
- CC - Customer Charges
- ED - Electricity Duty (as a percentage of Energy Charges) Print each charge separately with clear labels.

Ensure billing accuracy and verify intermediate results. Improve output formatting for better clarity.

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:

```

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_charges is None or fixed_charges is None:
192     print("Invalid customer type. Please enter 'Domestic', 'Commercial', or 'Industrial'.")
193     exit()
194 total_bill = energy_charges + fixed_charges + electricity_duty
195 # Display the bill details
196 print(f"Units Consumed: {units_consumed}")
197 print(f"Energy Charges: ${energy_charges:.2f}")
198
199 print(f"Fixed Charges: ${fixed_charges:.2f}")
200 print(f"Electricity Duty (@{duty_percentage}%): ${electricity_duty:.2f}")

```

Enter Previous Units (PU): 45
 Enter Current Units (CU): 84
 Enter Type of Customer (Domestic/Commercial/Industrial): INDUSTRIAL
 Units Consumed: 39.0
 Energy Charges: \$97.50
 Fixed Charges: \$150.00
 Electricity Duty (@5.0%): \$4.88
 Total Bill Amount: \$252.38

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 the final Python electricity billing program. #Requirements:

Calculate Total Bill using:

Total Bill = EC + FC + CC + ED #Display

clearly:

#Energy Charges (EC) #Fixed

Charges (FC) #Customer

Charges (CC) #Electricity Duty

(ED) #Total Bill Amount

#Format the output neatly like a real electricity bill.

CODE:

```
def calculate_customer_charges(customer_type):
```

```
    if customer_type == "domestic":
```

```
        return 20.0
```

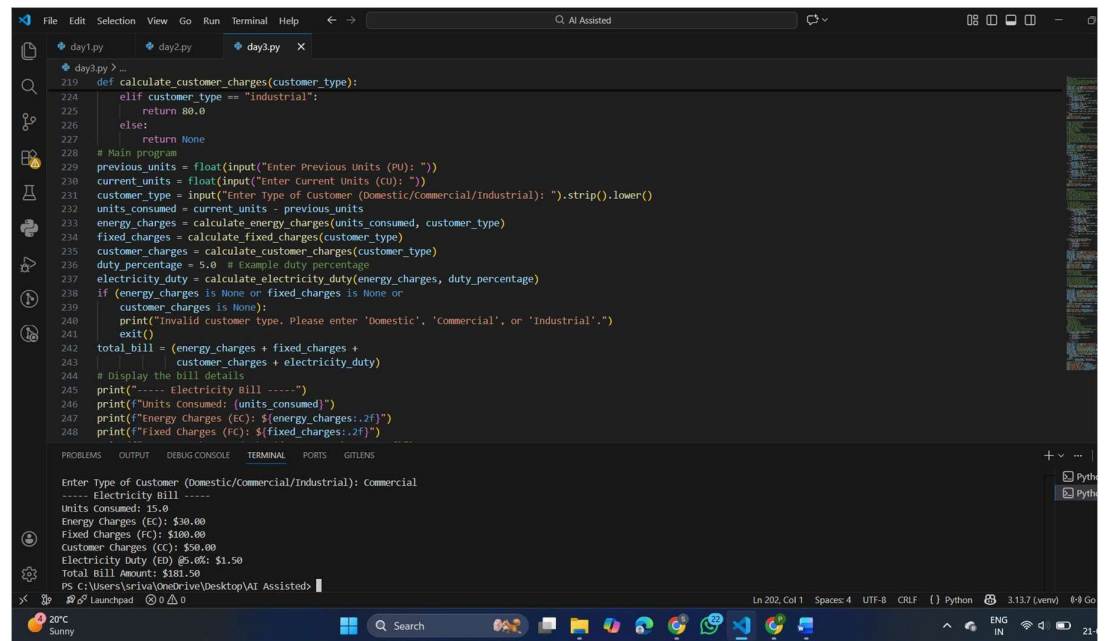


```

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

```

OUTPUT:



```
219 def calculate_customer_charges(customer_type):
220     if customer_type == "Industrial":
221         return 80.0
222     else:
223         return None
224 # Main program
225 previous_units = float(input("Enter Previous Units (PU): "))
226 current_units = float(input("Enter Current Units (CU): "))
227 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip().lower()
228 units_consumed = current_units - previous_units
229 energy_charges = calculate_energy_charges(units_consumed, customer_type)
230 fixed_charges = calculate_fixed_charges(customer_type)
231 customer_charges = calculate_customer_charges(customer_type)
232 duty_percentage = 5.0 # Example duty percentage
233 electricity_duty = calculate_electricity_duty(energy_charges, duty_percentage)
234 if (energy_charges is None or fixed_charges is None or
235     customer_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 # Display the bill details
241 print("----- Electricity Bill -----")
242 print(f"Units Consumed: {units_consumed}")
243 print(f"Energy Charges (EC): ${energy_charges:2f}")
244 print(f"Fixed Charges (FC): ${fixed_charges:2f}")
245 print(f"Customer Charges (CC): ${customer_charges:2f}")
246 print(f"Electricity Duty (ED) @5.0%: ${electricity_duty:2f}")
247 print(f"Total Bill Amount: ${total_bill:2f}")
```

Enter Type of Customer (Domestic/Commercial/Industrial): Commercial
----- Electricity Bill -----
Units Consumed: 15.0
Energy Charges (EC): \$30.00
Fixed Charges (FC): \$100.00
Customer Charges (CC): \$50.00
Electricity Duty (ED) @5.0%: \$1.50
Total Bill Amount: \$181.50
PS C:\Users\sriya\OneDrive\Desktop\AI Assisted>

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