

ASSIGNMENT-3.3

2303A51337

BATCH-10

➤ TASK-1:

PROMPT:

PYTHON code for reading consumer details and generate a python program it should read previous units and current units and type of customer (domestic or commercial) and calculate the units consumed program without function

Reading consumer details and calculating units consumed

CODE:

```
previous_units = int(input("Enter previous meter reading: "))

current_units = int(input("Enter current meter reading: "))

customer_type = input("Enter customer type (domestic/commercial): ").strip().lower()

units_consumed = current_units - previous_units

if units_consumed < 0:

    print("Error: Current units cannot be less than previous units.")

else:

    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) * 3.5

else:

    bill_amount = (100 * 2.0) + (200 * 3.5) + (units_consumed - 300) * 5.0

else:

    print("Error: Invalid customer type.")

    bill_amount = None
```

if bill_amount is not None:

```
print(f"Units Consumed: {units_consumed}")
```

OUTPUT:

```
File Edit Selection View Go Run Terminal Help < → Q AI Assisted
OPEN EDITORS
AI ASSISTED
OUTLINE
TIMELINE
lab3.py x lab4.py
#ITASK-1 PYTHON code for reading consumer details and generate a python program it should read previous units and current units and type of customer
previous_units = int(input("Enter previous meter reading: "))
current_units = int(input("Enter current meter reading: "))
customer_type = input("Enter customer type (domestic/commercial): ").strip().lower()
units_consumed = current_units - previous_units
if units_consumed < 0:
    print("Error: current units cannot be less than previous units.")
else:
    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) * 3.5
        else:
            bill_amount = (100 * 2.0) + (200 * 3.5) + (units_consumed - 300) * 5.0
    else:
        print("Error: Invalid customer type.")
        bill_amount = None
    if bill_amount is not None:
        print(f"Units Consumed: {units_consumed}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
+ · · · ×
Enter previous meter reading: 95
Enter current meter reading: 104
Enter customer type (domestic/commercial): domestic
Units Consumed: 9
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted & C:/Users/sriva/appData/local/Programs/Python/Python313/python.exe "C:/Users/sriva/OneDrive/Desktop/lab3.py"
Ln 31, Col 1 Spaces: 4 UTT-8 CRLF () Python 3.13.7 ↵ Go Live
```

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

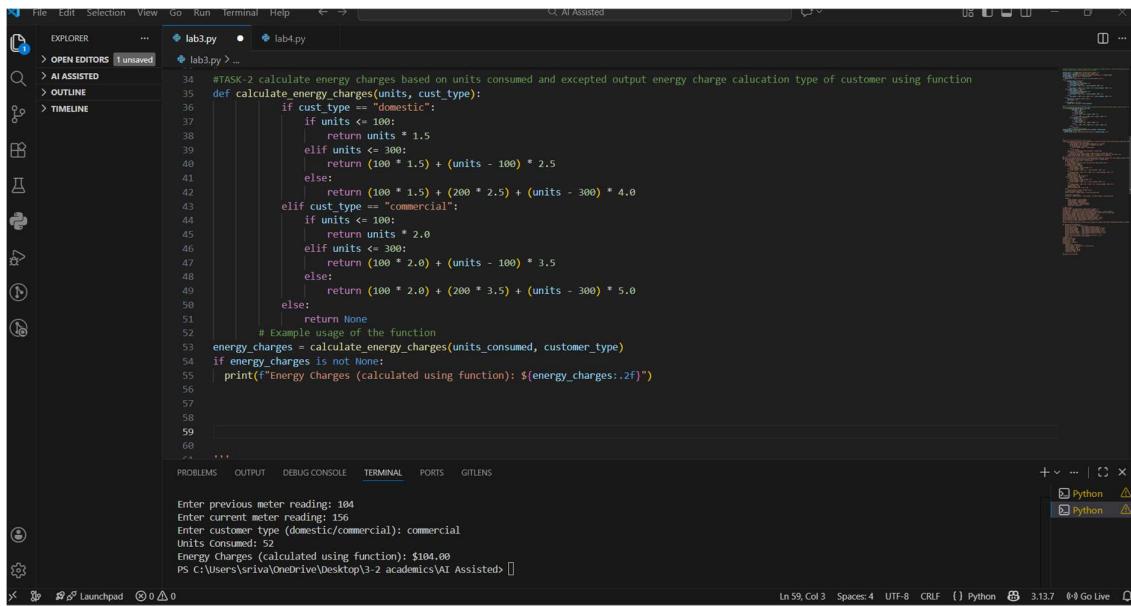
PROMPT:

calculate energy charges based on units consumed and excepted output energy charge calucation type of customer using function.

CODE:

```
def calculate_energy_charges(units, cust_type):  
    if cust_type == "domestic":  
        if units <= 100:  
            return units * 1.5  
        elif units <= 300:  
            return (100 * 1.5) + (units - 100) * 2.5  
        else:  
            return (100 * 1.5) + (200 * 2.5) + (units - 300) * 4.0  
    elif cust_type == "commercial":  
        if units <= 100:  
            return units * 2.0  
        elif units <= 300:  
            return (100 * 2.0) + (units - 100) * 3.5  
        else:  
            return (100 * 2.0) + (200 * 3.5) + (units - 300) * 5.0  
    else:  
        return None  
  
# Example usage of the function  
  
energy_charges = calculate_energy_charges(units_consumed, customer_type)  
if energy_charges is not None:  
    print(f"Energy Charges (calculated using function): ${energy_charges:.2f}")
```

OUTPUT:



```
#TASK-2 calculate energy charges based on units consumed and excepted output energy charge calucation type of customer using function
def calculate_energy_charges(units, cust_type):
    if cust_type == "domestic":
        if units <= 100:
            return units * 1.5
        elif units <= 300:
            return (100 * 1.5) + (units - 100) * 2.5
        else:
            return (100 * 1.5) + (200 * 2.5) + (units - 300) * 4.0
    elif cust_type == "commercial":
        if units <= 100:
            return units * 2.0
        elif units <= 300:
            return (100 * 2.0) + (units - 100) * 3.5
        else:
            return (100 * 2.0) + (200 * 3.5) + (units - 300) * 5.0
    else:
        return None
# Example usage of the function
energy_charges = calculate_energy_charges(units_consumed, customer_type)
if energy_charges is not None:
    print(f"Energy Charges (calculated using function): ${energy_charges:.2f}")

Enter previous meter reading: 104
Enter current meter reading: 156
Enter customer type (domestic/commercial): commercial
Units consumed: 52
Energy Charges (calculated using function): $104.00
```

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:

calculate billing logic must be reusable for multiple consumers and calculate energy charges and fixed charges return calculated values

CODE:

```
def calculate_bill(units, cust_type):

    energy_charges = calculate_energy_charges(units, cust_type)

    fixed_charges = 50 if cust_type == "domestic" else 100

    if energy_charges is not None:

        return energy_charges + fixed_charges

    else:

        return None
```

```

total_bill = calculate_bill(units_consumed, customer_type)

if total_bill is not None:

    print(f"Total Bill Amount (using reusable function): ${total_bill:.2f}")

print(f"Energy Charges: ${bill_amount - (50 if customer_type == 'domestic' else 100):.2f}")

print(f"Fixed Charges: ${50 if customer_type == 'domestic' else 100:.2f}")

```

OUTPUT:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows two files: `lab3.py` and `lab4.py`.
- Code Editor:** Displays the Python code for Task 3.
- Terminal:** Shows the execution of the code and its output. The user enters meter reading (123), customer type (DOMESTIC), and units consumed (10). The program calculates energy charges (\$28.50), total bill amount (\$78.50), and fixed charges (\$50.00).
- Status Bar:** Shows the file path (C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted\lab3.py), line number (Ln 83, Col 1), and other terminal settings.

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:

Generate electricity bill including multiple additional charges like fixed charges, customer charges, percentage of electricity duty, duty calculation by improving accuracy

CODE:

```
def calculate_electricity_bill(previous_units, current_units, customer_type):  
    units_consumed = current_units - previous_units  
  
    if units_consumed < 0:  
        raise ValueError("Current units cannot be less than previous units.")  
  
    if customer_type == "domestic":  
        if units_consumed <= 100:  
            energy_charges = units_consumed * 1.5  
  
        elif units_consumed <= 300:  
            energy_charges = (100 * 1.5) + (units_consumed - 100) * 2.5  
  
        else:  
            energy_charges = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0  
  
        fixed_charges = 50  
  
        customer_charges = 20  
  
        electricity_duty_rate = 0.05  
  
    elif customer_type == "commercial":  
        if units_consumed <= 100:  
            energy_charges = units_consumed * 2.0  
  
        elif units_consumed <= 300:  
            energy_charges = (100 * 2.0) + (units_consumed - 100) * 3.5  
  
        else:  
            energy_charges = (100 * 2.0) + (200 * 3.5) + (units_consumed - 300) * 5.0  
  
        fixed_charges = 100  
  
        customer_charges = 50  
  
        electricity_duty_rate = 0.10  
  
    else:  
        raise ValueError("Customer type must be domestic or commercial.")  
  
    electricity_duty = energy_charges * electricity_duty_rate
```

```

total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty

return units_consumed, energy_charges, fixed_charges, customer_charges,
electricity_duty, total_bill

def get_int(msg):

    while True:

        value = input(msg)

        if value.strip() == "":
            print("Input cannot be empty. Enter a number.")

        elif not value.isdigit():

            print("Enter only digits.")

        else:

            return int(value)

try:

    previous_units = get_int("Enter previous meter reading: ")

    current_units = get_int("Enter current meter reading: ")

    customer_type = input("Enter customer type (domestic/commercial): ").strip().lower()

    u, e, f, c, d, t = calculate_electricity_bill(previous_units, current_units, customer_type)

    print("\n--- Electricity Bill ---")

    print("Units Consumed:", u)

    print("Energy Charges:", round(e, 2))

    print("Fixed Charges:", f)

    print("Customer Charges:", c)

    print("Electricity Duty:", round(d, 2))

    print("Total Bill Amount:", round(t, 2))

except ValueError as err:

    print("Error:", err)

```

OUTPUT:

```
File Edit Selection View Go Run Terminal Help AI Assisted
OPEN EDITORS 1 unsaved
AI ASSISTED
OUTLINE
TIMELINE
lab3.py lab4.py calculate_electricity_bill
72 def calculate_electricity_bill(previous_units, current_units, customer_type):
73     units_consumed = current_units - previous_units
74     if units_consumed < 0:
75         raise ValueError("Current units cannot be less than previous units.")
76     if customer_type == "domestic":
77         if units_consumed <= 100:
78             energy_charges = units_consumed * 1.5
79         elif units_consumed <= 300:
80             energy_charges = (100 * 1.5) + (units_consumed - 100) * 2.5
81         else:
82             energy_charges = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
83         fixed_charges = 50
84         customer_charges = 20
85         electricity_duty_rate = 0.05
86     elif customer_type == "commercial":
87         if units_consumed <= 100:
88             energy_charge = units_consumed * 2.0
89         elif units_consumed <= 300:
90             energy_charge = (100 * 2.0) + (units_consumed - 100) * 3.5
91         else:
92             energy_charge = (100 * 2.0) + (200 * 3.5) + (units_consumed - 300) * 5.0
93         fixed_charges = 100
94         customer_charges = 50
95         electricity_duty_rate = 0.10
96     else:
97         raise ValueError("Customer type must be domestic or commercial.")
98     electricity_duty = energy_charges * electricity_duty_rate
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Enter previous meter reading: 123
Enter current meter reading: 159
Enter customer type (domestic/commercial): domestic

--- Electricity Bill ---
Units Consumed: 36
Energy Charges: 54.0
Fixed Charges: 50

Lev 75 Col 19 Source A - UTF-8 CR LF Python 3.12.7 Feb 2024

```
File Edit Selection View Go Run Terminal Help AI Assisted
OPEN EDITORS 1 unsaved
AI ASSISTED
OUTLINE
TIMELINE
lab3.py lab4.py calculate_electricity_bill
72 def calculate_electricity_bill(previous_units, current_units, customer_type):
99     total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty
100    return units_consumed, energy_charges, fixed_charges, customer_charges, electricity_duty, total_bill
101
102 def get_int(msg):
103     while True:
104         value = input(msg)
105         if value.strip() == "":
106             print("Input cannot be empty. Enter a number.")
107         elif not value.isdigit():
108             print("Enter only digits.")
109         else:
110             return int(value)
111
112 try:
113     previous_units = get_int("Enter previous meter reading: ")
114     current_units = get_int("Enter current meter reading: ")
115     customer_type = input("Enter customer type (domestic/commercial): ").strip().lower()
116     u, e, f, c, d, t = calculate_electricity_bill(previous_units, current_units, customer_type)
117     print("\n--- Electricity Bill ---")
118     print("Units Consumed:", u)
119     print("Energy charges:", round(e, 2))
120     print("Fixed Charges:", f)
121     print("Customer Charges:", c)
122     print("Electricity Duty:", round(d, 2))
123     print("Total Bill Amount:", round(t, 2))
124
125 except ValueError as err:
126     print("Error:", err)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

... Electricity Bill ...
Units Consumed: 36
Energy Charges: 54.0
Fixed Charges: 50
Customer Charges: 20
Electricity Duty: 2.7
Total Bill Amount: 126.7

PS C:\Users\riva\OneDrive\Desktop\3-2_academics\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:

Generate final bill of electricity including all charges with proper formatting and display in python .

CODE:

```
def display_bill(bill_details):

    print("\n----- Electricity Bill -----")

    print(f"Consumer ID : {bill_details['Consumer ID']}")

    print(f"Units Consumed : {bill_details['Units Consumed']} units")

    print(f"Energy Charges : ₹{bill_details['Energy Charges']:.2f}")

    print(f"Fixed Charges : ₹{bill_details['Fixed Charges']:.2f}")

    print(f"Customer Charges : ₹{bill_details['Customer Charges']:.2f}")

    print(f"Electricity Duty : ₹{bill_details['Electricity Duty']:.2f}")

    print("-----")

    print(f"Total Bill Amount : ₹{bill_details['Total Bill']:.2f}")

    print("-----\n")

# Example usage

consumer_id = "C12345"

previous_units = 500

current_units = 750

consumer_type = "domestic"

bill_details = {

    "Consumer ID": consumer_id,

    "Units Consumed": current_units - previous_units,

    "Energy Charges": 625.00,

    "Fixed Charges": 50.00,

    "Customer Charges": 20.00,

    "Electricity Duty": 31.25,
```

```

    "Total Bill": 726.25

}

display_bill(bill_details)

```

OUTPUT:

The screenshot shows a code editor interface with two files open: lab3.py and lab4.py. The lab3.py file contains a function definition for 'display_bill' which prints various bill details. The lab4.py file contains an example usage of this function with specific consumer data.

```

lab3.py
130 def display_bill(bill_details):
131     print("----- Electricity Bill -----")
132     print(f"Consumer ID : {bill_details['Consumer ID']}")
133     print(f"Units Consumed : {bill_details['Units consumed']} units")
134     print(f"Energy Charges : ₹{bill_details['Energy charges']:.2f}")
135     print(f"Fixed Charges : ₹{bill_details['Fixed charges']:.2f}")
136     print(f"Customer Charges : ₹{bill_details['Customer charges']:.2f}")
137     print(f"Electricity Duty : ₹{bill_details['Electricity duty']:.2f}")
138     print("-----")
139     print(f"Total Bill Amount : ₹{bill_details['Total Bill']:.2f}")
140     print("-----\n")
141
142 # Example usage
143 consumer_id = "C12345"
144 previous_units = 500
145 current_units = 750
146 consumer_type = "domestic"
147 bill_details = {
148     "Consumer ID": consumer_id,
149     "Units consumed": current_units - previous_units,
150     "Energy charges": 625.00,
151     "Fixed charges": 50.00,
152     "Customer charges": 20.00,
153     "Electricity Duty": 31.25,
154     "Total Bill": 726.25
155 }
156 display_bill(bill_details)
157

```

The output window shows the printed bill details:

```

Consumer ID : C12345
Units Consumed : 250 units
Energy Charges : ₹625.00
Fixed Charges : ₹50.00
Customer Charges : ₹20.00
Electricity Duty : ₹31.25
-----
Total Bill Amount : ₹726.25

```

EXPLANATION:

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