

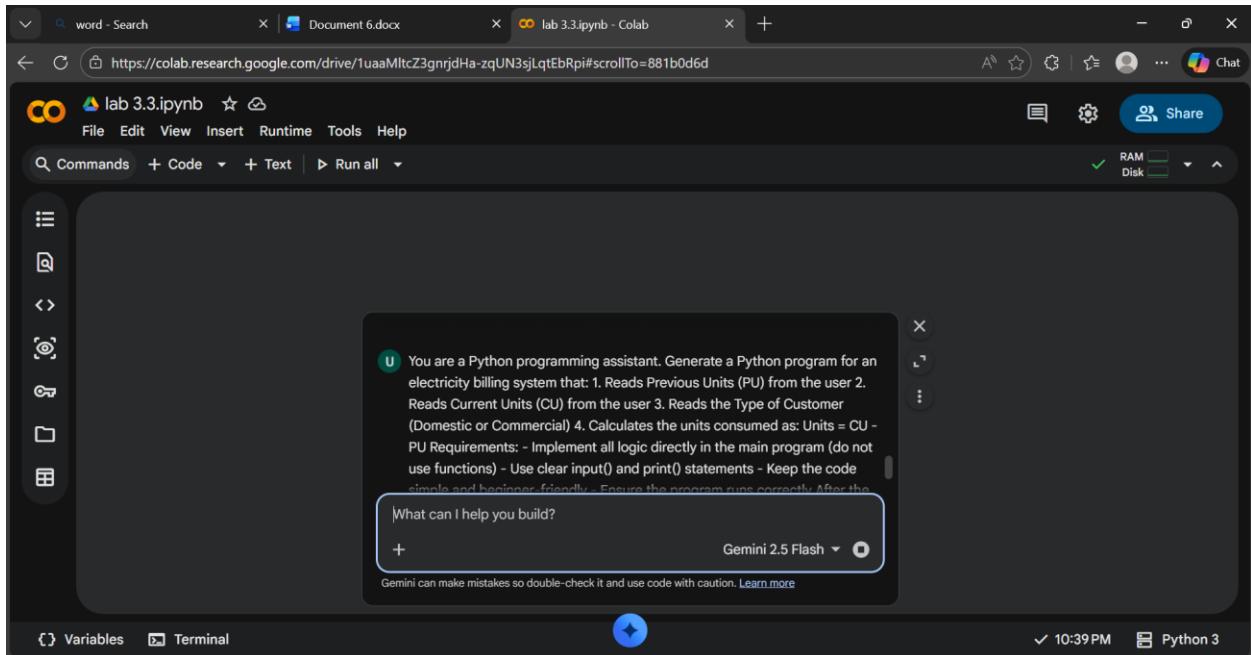
Assignment 3.3 AI ASSISTED CODING

Htno:2303a51305

Btno:05

Task 1: AI-Generated Logic for Reading Consumer Details .

Prompt:



Code:

```
# 1. Reads Previous Units (PU) from the user
previous_units = int(input("Enter Previous Units (PU): "))

# 2. Reads Current Units (CU) from the user
current_units = int(input("Enter Current Units (CU): "))

# 3. Reads the Type of Customer (Domestic or Commercial)
customer_type = input("Enter Customer Type (Domestic/Commercial): ")

# 4. Calculates the units consumed as: Units = CU - PU
units_consumed = current_units - previous_units

# Display the calculated values clearly
print("\n--- Electricity Bill Summary ---")
print(f"Previous Units: {previous_units}")
print(f"Current Units: {current_units}")
print(f"Customer Type: {customer_type}")
print(f"Units Consumed: {units_consumed}")
print("-----")
```

Output:

```
Enter Previous Units (PU): 150
Enter Current Units (CU): 200
Enter Customer Type (Domestic/Commercial): domestic

--- Electricity Bill Summary ---
Previous Units: 150
Current Units: 200
Customer Type: domestic
Units Consumed: 50
-----
```

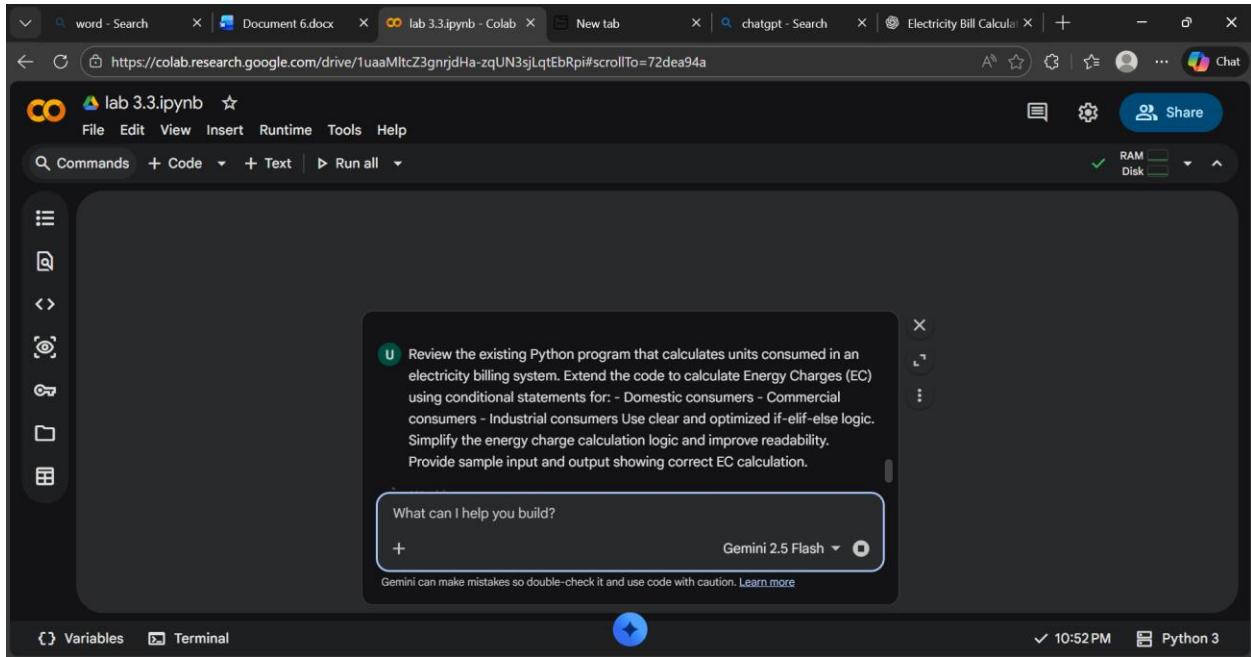
Explanation: The program reads previous units, current units, and customer type from the user.

It calculates **units consumed** by subtracting previous units from current units ($200 - 150 = 50$).

Finally, it displays a clear electricity bill summary with all entered details and the calculated consumption.

Task 2: Energy Charges Calculation Based on Units Consumed.

Prompt:



Code:

The image shows two screenshots of the Google Colab interface, each displaying a code cell for calculating electricity bills based on customer type (Domestic, Commercial, or Industrial).

Code Cell 1 (Top):

```
# 1. Reads Previous Units (PU) from the user
previous_units = int(input("Enter Previous Units (PU): "))

# 2. Reads Current Units (CU) from the user
current_units = int(input("Enter Current Units (CU): "))

# 3. Reads the Type of Customer (Domestic or Commercial or Industrial)
customer_type = input("Enter Customer Type (Domestic/Commercial/Industrial): ").strip().lower()

# 4. Calculates the units consumed as: Units = CU - PU
units_consumed = current_units - previous_units

# Define tariff rates (example rates)
domestic_rate_per_unit = 5.0
commercial_rate_per_unit = 8.0
industrial_rate_per_unit = 10.0

energy_charges = 0.0

# Calculate Energy Charges (EC) using conditional statements
if customer_type == 'domestic':
    energy_charges = units_consumed * domestic_rate_per_unit
```

Code Cell 2 (Bottom):

```
if customer_type == 'domestic':
    energy_charges = units_consumed * domestic_rate_per_unit
    rate_applied = domestic_rate_per_unit
elif customer_type == 'commercial':
    energy_charges = units_consumed * commercial_rate_per_unit
    rate_applied = commercial_rate_per_unit
elif customer_type == 'industrial':
    energy_charges = units_consumed * industrial_rate_per_unit
    rate_applied = industrial_rate_per_unit
else:
    print("Invalid customer type entered. Energy charges cannot be calculated.")
    rate_applied = 0.0 # Assign a default or indicate no rate

# Display the calculated values clearly
print("\n--- Electricity Bill Summary ---")
print(f"Previous Units: {previous_units}")
print(f"Current Units: {current_units}")
print(f"Customer Type: {customer_type.capitalize()}")
print(f"Units Consumed: {units_consumed}")

if energy_charges > 0 or customer_type in ['domestic', 'commercial', 'industrial']:
    print(f"Rate Applied (per unit): {rate_applied:.2f}")
```

Output:

The screenshot shows a Google Colab interface with a dark theme. The code cell contains the following Python script:

```
Enter Previous Units (PU): 100
...
Enter Current Units (CU): 250
Enter Customer Type (Domestic/Commercial/Industrial): domestic

--- Electricity Bill Summary ---
Previous Units: 100
Current Units: 250
Customer Type: Domestic
Units Consumed: 150
Rate Applied (per unit): 5.00
Energy Charges (EC): 750.00
```

The output of the script is displayed in the cell below the code.

Explanation: The extended program calculates **Energy Charges** based on units consumed and customer type using conditional statements.

Optimized **if-elif-else** logic improves readability and makes the billing rules easy to understand.

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

Prompt:

The screenshot shows a Google Colab interface with a dark theme. A modal window titled "U" contains the following prompt for ChatGPT:

Generate a Python program for an electricity billing system using modular design. Use user-defined functions to: 1. Calculate Energy Charges (EC) based on units consumed and customer type 2. Calculate Fixed Charges (FC) based on customer type Each function should return the calculated value. Include meaningful comments for clarity. Provide test cases with sample input and output to demonstrate correct EC and FC calculations.

The AI response starts with "Working..." and includes a text input field for "What can I help you build?".

Code:

The screenshot shows a Google Colab notebook titled "lab 3.3.ipynb". The code cell contains Python code defining tariff rates and fixed charges for domestic, commercial, and industrial customers. It includes functions to calculate energy charges based on consumption and customer type, and fixed charges based on customer type.

```
# --- Define Tariff Rates and Fixed Charges ---
# Rates per unit for different customer types
DOMESTIC_RATE_PER_UNIT = 5.0
COMMERCIAL_RATE_PER_UNIT = 8.0
INDUSTRIAL_RATE_PER_UNIT = 10.0

# Fixed charges for different customer types
DOMESTIC_FIXED_CHARGE = 50.0
COMMERCIAL_FIXED_CHARGE = 150.0
INDUSTRIAL_FIXED_CHARGE = 300.0

# --- Function to Calculate Energy Charges (EC) ---
def calculate_energy_charges(units_consumed, customer_type):
    """
    Calculates Energy Charges based on units consumed and customer type.
    Returns the calculated energy charges.
    """
    energy_charge = 0.0
    rate_applied = 0.0

    if customer_type == 'domestic':
        energy_charge = units_consumed * DOMESTIC_RATE_PER_UNIT
        rate_applied = DOMESTIC_RATE_PER_UNIT

    elif customer_type == 'commercial':
        energy_charge = units_consumed * COMMERCIAL_RATE_PER_UNIT
        rate_applied = COMMERCIAL_RATE_PER_UNIT

    elif customer_type == 'industrial':
        energy_charge = units_consumed * INDUSTRIAL_RATE_PER_UNIT
        rate_applied = INDUSTRIAL_RATE_PER_UNIT

    else:
        print(f"Warning: Unknown customer type '{customer_type}'. Energy charges set to 0.")

    return energy_charge, rate_applied

# --- Function to Calculate Fixed Charges (FC) ---
def calculate_fixed_charges(customer_type):
    """
    Calculates Fixed Charges based on customer type.
    Returns the calculated fixed charges.
    """
    fixed_charge = 0.0
```

The screenshot shows a Google Colab notebook titled "lab 3.3.ipynb". The code cell contains Python code for calculating energy and fixed charges based on customer type. It uses an if-elif-else structure to determine the correct rate and charge for each customer type, with a default case for unknown types.

```
if customer_type == 'domestic':
    energy_charge = units_consumed * DOMESTIC_RATE_PER_UNIT
    rate_applied = DOMESTIC_RATE_PER_UNIT
elif customer_type == 'commercial':
    energy_charge = units_consumed * COMMERCIAL_RATE_PER_UNIT
    rate_applied = COMMERCIAL_RATE_PER_UNIT
elif customer_type == 'industrial':
    energy_charge = units_consumed * INDUSTRIAL_RATE_PER_UNIT
    rate_applied = INDUSTRIAL_RATE_PER_UNIT
else:
    print(f"Warning: Unknown customer type '{customer_type}'. Energy charges set to 0.")

return energy_charge, rate_applied

# --- Function to Calculate Fixed Charges (FC) ---
def calculate_fixed_charges(customer_type):
    """
    Calculates Fixed Charges based on customer type.
    Returns the calculated fixed charges.
    """
    fixed_charge = 0.0
```

The screenshot shows a Google Colab notebook titled "lab 3.3.ipynb". The code in cell [11] defines a function to calculate fixed charges based on customer type (domestic, commercial, or industrial) and then prints a main program logic to calculate total bill.

```
fixed_charge = 0
if customer_type == 'domestic':
    fixed_charge = DOMESTIC_FIXED_CHARGE
elif customer_type == 'commercial':
    fixed_charge = COMMERCIAL_FIXED_CHARGE
elif customer_type == 'industrial':
    fixed_charge = INDUSTRIAL_FIXED_CHARGE
else:
    # Warning already handled in energy charges, or can be added here too
    pass
return fixed_charge

# --- Main Program Logic ---
print("--- Electricity Billing System ---")

# 1. Read Previous Units (PU) from the user
previous_units = int(input("Enter Previous Units (PU): "))

# 2. Read Current Units (CU) from the user
current_units = int(input("Enter Current Units (CU): "))

# 3. Read the Type of Customer (Domestic, Commercial, or Industrial)
customer_type = input("Enter Customer Type (Domestic/Commercial/Industrial): ").strip().lower()
```

The screenshot shows the continuation of the Google Colab notebook "lab 3.3.ipynb". The code in cell [11] calculates total bill by adding energy charges to fixed charges and then displays a bill summary with various details.

```
# Calculate Fixed Charges using the function
fixed_charges = calculate_fixed_charges(customer_type)

# Calculate Total Bill
total_bill = energy_charges + fixed_charges

# --- Display Bill Summary ---
print("\n--- Bill Summary ---")
print(f"Previous Units: {previous_units}")
print(f"Current Units: {current_units}")
print(f"Customer Type: {customer_type.capitalize()}")
print(f"Units Consumed: {units_consumed}")

if rate_applied > 0:
    print(f"Rate Applied (per unit): {rate_applied:.2f}")

print(f"Energy Charges (EC): {energy_charges:.2f}")
print(f"Fixed Charges (FC): {fixed_charges:.2f}")
print(f"Total Bill: {total_bill:.2f}")
print("-----")
```

Output:

```
--- Electricity Billing System ---  
Enter Previous Units (PU): 100  
Enter Current Units (CU): 250  
Enter Customer Type (Domestic/Commercial/Industrial): domestic  
  
--- Bill Summary ---  
Previous Units: 100  
Current Units: 250  
Customer Type: Domestic  
Units Consumed: 150  
Rate Applied (per unit): 5.00  
Energy Charges (EC): 750.00  
Fixed Charges (FC): 50.00  
Total Bill: 800.00
```

Explanation: This program uses **functions** to make the billing logic reusable for multiple consumers.

Separating Energy Charges and Fixed Charges into functions improves clarity, modularity, and easy maintenance of the code.

Task 4: Calculation of Additional Charges.

Prompt:

Extend the existing electricity billing Python program to calculate additional charges. Add logic to compute - Fixed Charges (FC) - Customer Charges (CC) - Electricity Duty (ED) as a percentage of Energy Charges (EC). Ensure the calculations are accurate and intermediate values are clearly printed. Improve billing accuracy and structure the output neatly. Provide sample input and output showing all individual charges.

Code:

The image shows two screenshots of a Google Colab notebook titled "lab 3.3.ipynb".

Code Cell 1:

```
-2# --- Define Tariff Rates and Fixed Charges ---
+## --- Define Tariff Rates and Fixed Charges ---
# Rates per unit for different customer types
DOMESTIC_RATE_PER_UNIT = 5.0
COMMERCIAL_RATE_PER_UNIT = 8.0
INDUSTRIAL_RATE_PER_UNIT = 10.0

# Fixed charges for different customer types
DOMESTIC_FIXED_CHARGE = 50.0
COMMERCIAL_FIXED_CHARGE = 150.0
INDUSTRIAL_FIXED_CHARGE = 300.0

# Customer Charges (assuming a flat rate for simplicity for all types)
CUSTOMER_CHARGE = 20.0

# Electricity Duty (as a percentage of Energy Charges)
ELECTRICITY_DUTY_PERCENTAGE = 0.05 # 5%
```

Code Cell 2:

```
>Returns the calculated energy charges and the rate applied.
"""
energy_charge = 0.0
rate_applied = 0.0

if customer_type == 'domestic':
    energy_charge = units_consumed * DOMESTIC_RATE_PER_UNIT
    rate_applied = DOMESTIC_RATE_PER_UNIT
elif customer_type == 'commercial':
    energy_charge = units_consumed * COMMERCIAL_RATE_PER_UNIT
    rate_applied = COMMERCIAL_RATE_PER_UNIT
elif customer_type == 'industrial':
    energy_charge = units_consumed * INDUSTRIAL_RATE_PER_UNIT
    rate_applied = INDUSTRIAL_RATE_PER_UNIT
else:
    print(f"Warning: Unknown customer type '{customer_type}'. Energy charges set to 0.")

return energy_charge, rate_applied

-# --- Function to Calculate Fixed Charges (FC) ---
+## --- Function to Calculate Fixed Charges (FC) ---
def calculate_fixed_charges(customer_type):
    """
```

The image shows two screenshots of a Google Colab notebook titled "lab 3.3.ipynb".

Screenshot 1: The first cell contains Python code for calculating fixed charges based on customer type. It defines a function `calculate_fixed_charge` which returns a fixed charge based on the customer type ('domestic', 'commercial', or 'industrial'). It also includes a function `calculate_customer_charges` which calculates customer charges using a flat rate for all types.

```
Calculates Fixed Charges based on customer type.  
Returns the calculated fixed charges.  
"""  
fixed_charge = 0.0  
if customer_type == 'domestic':  
    fixed_charge = DOMESTIC_FIXED_CHARGE  
elif customer_type == 'commercial':  
    fixed_charge = COMMERCIAL_FIXED_CHARGE  
elif customer_type == 'industrial':  
    fixed_charge = INDUSTRIAL_FIXED_CHARGE  
return fixed_charge  
  
--- Function to Calculate Customer Charges (CC) ---  
def calculate_customer_charges(customer_type):  
    """  
    Calculates Customer Charges.  
    For simplicity, this example uses a flat rate for all customer types.  
    """  
    # This function could be expanded to have type-dependent customer charges if needed.  
    return CUSTOMER_CHARGE
```

Screenshot 2: The second cell contains Python code for calculating electricity duty. It defines a function `calculate_electricity_duty` which calculates electricity duty as a percentage of energy charges. The main program logic reads previous units, current units, and customer type from the user, then prints a message and calculates the total bill.

```
--- Function to Calculate Electricity Duty (ED) ---  
def calculate_electricity_duty(energy_charges):  
    """  
    Calculates Electricity Duty as a percentage of Energy Charges.  
    """  
    return energy_charges * ELECTRICITY_DUTY_PERCENTAGE  
  
--- Main Program Logic ---  
print("--- Electricity Billing System (Extended) ---")  
  
# 1. Read Previous Units (PU) from the user  
previous_units = int(input("Enter Previous Units (PU): "))  
  
# 2. Read Current Units (CU) from the user  
current_units = int(input("Enter Current Units (CU): "))  
  
# 3. Read the Type of Customer (Domestic, Commercial, or Industrial)  
customer_type = input("Enter Customer Type (Domestic/Commercial/Industrial): ").strip().lower()  
  
# Calculate Units Consumed
```

A screenshot tool window is visible in the bottom right corner, indicating a screenshot has been copied to the clipboard.

The screenshot shows a Google Colab notebook titled "lab 3.3.ipynb". The code calculates a total bill by summing energy charges, fixed charges, customer charges, and electricity duty. It then prints a detailed bill summary with previous units, current units, customer type, and units consumed. It also prints individual charges (Energy, Fixed, Customer) and the total bill. The code is annotated with comments and docstrings.

```
# Calculate Total Bill
total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty

# --- Display Detailed Bill Summary ---
print("n--- Detailed Bill Summary ---")
print("n--- Detailed Bill Summary ---")
print("Previous Units: {previous_units}")
print("Current Units: {current_units}")
print("Customer Type: {customer_type.capitalize()}")
print("Units Consumed: {units_consumed}")
print("Units Consumed: {units_consumed} (Actual difference)")

if rate_applied > 0:
    print("Rate Applied (per unit): {rate_applied:.2f}")

print("Energy Charges (EC): {energy_charges:.2f}")
print("Fixed Charges (FC): {fixed_charges:.2f}")
print("Customer Charges (CC): {customer_charges:.2f}")
print("Electricity Duty (ED) ({ELECTRICITY_DUTY_PERCENTAGE*100:.0f}% of EC): {electricity_duty:.2f}")
print("Total Bill: {total_bill:.2f}")
print("-----")
```

Output:

The screenshot shows the execution output of the Python code. It prompts for previous and current units, and asks for customer type. It then displays a detailed bill summary with all components and the final total bill of 857.50.

```
--- Electricity Billing System (Extended) ---
Enter Previous Units (PU): 100
Enter Current Units (CU): 250
Enter Customer Type (Domestic/Commercial/Industrial): domestic

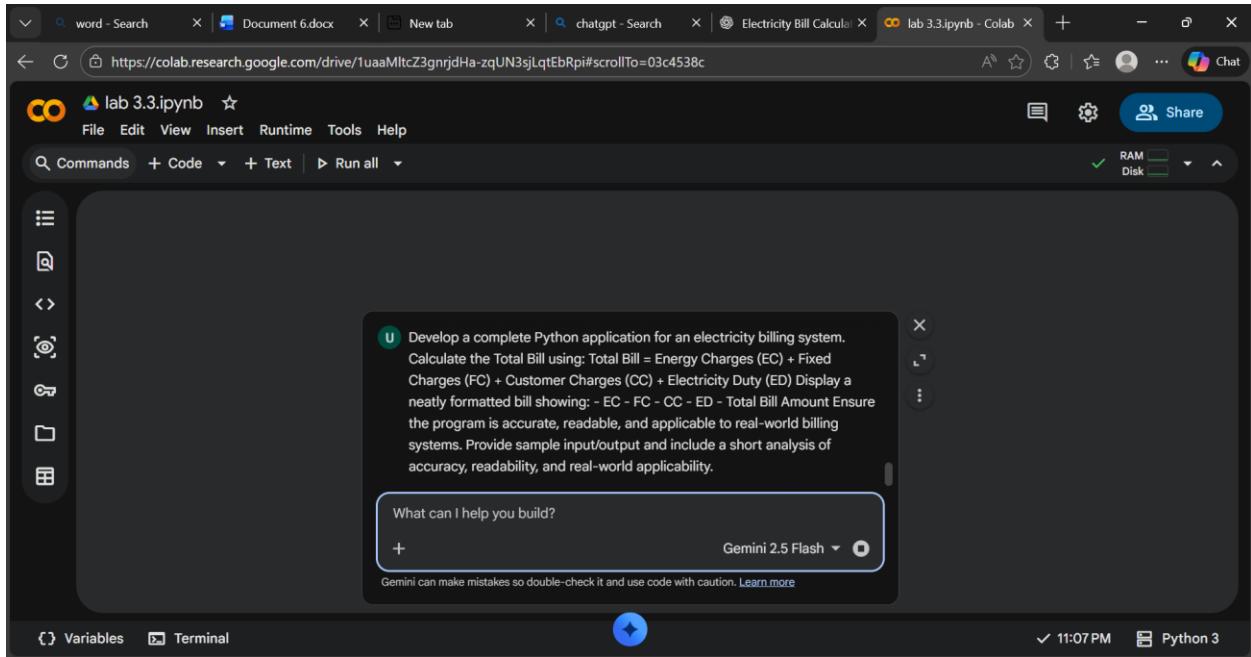
--- Detailed Bill Summary ---
Previous Units: 100
Current Units: 250
Customer Type: Domestic
Units Consumed: 150
Rate Applied (per unit): 5.00
Energy Charges (EC): 750.00
Fixed Charges (FC): 50.00
Customer Charges (CC): 20.00
Electricity Duty (ED) (5% of EC): 37.50
Total Bill: 857.50
-----
```

Explanation: The program is enhanced to include **additional billing components** like fixed charges, customer charges, and electricity duty.

Electricity Duty is calculated as a **percentage of Energy Charges**, and all values are displayed clearly for verification.

Task 5: Final Bill Generation and Output Analysis.

Prompt:



Code:

The screenshot shows a Google Colab notebook titled "lab 3.3.ipynb". The code cell [15] contains the following Python script:

```
import pandas as pd

# 1. Define global constants for domestic, commercial, and industrial rates per unit
DOMESTIC_RATE_PER_UNIT = 5.0
COMMERCIAL_RATE_PER_UNIT = 8.0
INDUSTRIAL_RATE_PER_UNIT = 10.0

# 2. Define global constants for domestic, commercial, and industrial fixed charges
DOMESTIC_FIXED_CHARGE = 50.0
COMMERCIAL_FIXED_CHARGE = 150.0
INDUSTRIAL_FIXED_CHARGE = 300.0

# 3. Define a global constant for the customer charge
CUSTOMER_CHARGE = 20.0

# 4. Define a global constant for the electricity duty percentage
ELECTRICITY_DUTY_PERCENTAGE = 0.05

def calculate_energy_charges(units_consumed, customer_type):
    """
    Calculates energy charges based on units consumed and customer type.
    Returns energy charges and the rate applied per unit.
    """
    if customer_type == 'domestic':
        rate_applied = COMMERCIAL_RATE_PER_UNIT
    elif customer_type == 'commercial':
        rate_applied = INDUSTRIAL_RATE_PER_UNIT
    else:
        raise ValueError("Invalid customer type")
    energy_charges = units_consumed * rate_applied
    return energy_charges, rate_applied
```

The screenshot shows the same Google Colab notebook "lab 3.3.ipynb". The code cell [15] contains the same Python script as the previous screenshot. A Snipping Tool window is overlaid on the right side of the screen, showing a screenshot of the Colab interface. The status bar at the bottom right indicates "Screenshot copied to clipboard" and "Automatically saved".

```
def calculate_energy_charges(units_consumed, customer_type):
    """
    Calculates energy charges based on units consumed and customer type.
    Returns energy charges and the rate applied per unit.
    """
    if customer_type == 'domestic':
        rate_applied = COMMERCIAL_RATE_PER_UNIT
    elif customer_type == 'commercial':
        rate_applied = INDUSTRIAL_RATE_PER_UNIT
    else:
        raise ValueError("Invalid customer type")
    energy_charges = units_consumed * rate_applied
    return energy_charges, rate_applied
```

The screenshot shows a Google Colab notebook titled "lab 3.3.ipynb". The code defines a function `calculate_electricity_duty` which calculates the total electricity charge by summing energy charges, fixed charges, and customer charges, then applying a percentage duty. It also prints a success message and a welcome message for the electricity bill calculator.

```
[15]  Os
def calculate_electricity_duty(energy_charges, fixed_charges, customer_charges):
    """
    Calculates electricity duty based on total charges.
    Returns the electricity duty amount.
    """
    total_charges_before_duty = energy_charges + fixed_charges + customer_charges
    electricity_duty = total_charges_before_duty * ELECTRICITY_DUTY_PERCENTAGE
    return electricity_duty

print("Constants and functions defined successfully.")

... Constants and functions defined successfully.

[16]  12s
print("\n--- Electricity Bill Calculator ---")

# 1. Read inputs from the user
try:
    previous_units = int(input("Enter Previous Units: "))
    current_units = int(input("Enter Current Units: "))
    customer_type_input = input("Enter Customer Type (domestic, commercial, industrial): ").lower()

```

The screenshot shows a Google Colab notebook titled "lab 3.3.ipynb". The code handles user input for current and previous units, calculates actual units consumed, and handles cases where current units are less than previous units. It validates customer type (domestic, commercial, or industrial) and calls functions to calculate energy and fixed charges.

```
[18]  Os
try:
    # Calculate actual_units_consumed
    actual_units_consumed = current_units - previous_units

    # Handle cases where Current Units are less than Previous Units
    if current_units < previous_units:
        print("Warning: Current Units are less than Previous Units. Billing units will be set to 0.")
        units_for_billing = 0
    else:
        units_for_billing = actual_units_consumed

    # Validate customer type
    if customer_type_input not in ['domestic', 'commercial', 'industrial']:
        raise ValueError("Invalid customer type provided.")

    # Call calculate_energy_charges
    energy_charges, rate_applied = calculate_energy_charges(units_for_billing, customer_type_input)

    # Call calculate_fixed_charges
    fixed_charges = calculate_fixed_charges(customer_type_input)

```

The image shows two screenshots of the Google Colab interface, both titled "lab 3.3.ipynb".

Screenshot 1: This screenshot shows a single code cell containing Python code to calculate an electricity bill. The code defines functions for fixed charges, customer charges, and electricity duty, then calculates the total bill amount and prints a summary.

```
# Call calculate_fixed_charges
fixed_charges = calculate_fixed_charges(customer_type_input)

# Call calculate_customer_charges
customer_charges = calculate_customer_charges()

# Call calculate_electricity_duty
electricity_duty = calculate_electricity_duty(energy_charges, fixed_charges, customer_charges)

# Calculate total_bill_amount
total_bill_amount = energy_charges + fixed_charges + customer_charges + electricity_duty

# Display Bill Summary
print("\n--- Electricity Bill Summary ---")
print(f"Previous Units: {previous_units} units")
print(f"Current Units: {current_units} units")
print(f"Customer Type: {customer_type_input.capitalize()}")
print(f"Actual Units Consumed: {actual_units_consumed} units")
print(f"Units Billed: {units_for_billing} units")
print(f"Rate Applied: ${rate_applied:.2f} per unit")
print("\n--- Charges Breakdown ---")
print(f"Energy Charges: ${energy_charges:.2f}")
```

Screenshot 2: This screenshot shows a code cell with several print statements and sample runs. It prints the customer charges, electricity duty, and final total bill amount. It then handles exceptions for ValueErrors and other errors. Finally, it runs three sample simulations for domestic, commercial, and industrial customers.

```
print(f"Customer Charges: ${customer_charges:.2f}")
print(f"Electricity Duty: ${electricity_duty:.2f}")
print("-----")
print(f"Final Total Bill Amount: ${total_bill_amount:.2f}")
print("-----")

except ValueError as e:
    print(f"Error: {e}")
except Exception as e:
    print(f>An unexpected error occurred: {e}")

# Sample Run 1: Domestic Customer
run_billing_simulation(100, 250, 'domestic')

# Sample Run 2: Commercial Customer
run_billing_simulation(200, 450, 'commercial')

# Sample Run 3: Industrial Customer
run_billing_simulation(500, 1000, 'industrial')

print("\nAll sample simulations completed.")
```

Output:

The screenshot shows a Google Colab interface with a single notebook tab titled "lab 3.3.ipynb". The code cell contains Python code for simulating electricity bills. The output displays the simulation results for a domestic customer, including bill summary, charges breakdown, and total bill amount.

```
run([18], {"name": "Os"}, {"name": "Industrial"}),  
print("\nAll sample simulations completed.")  
...  
--- Simulating Bill for Domestic Customer ---  
Inputs: Previous Units=100, Current Units=250, Customer Type=domestic  
--- Electricity Bill Summary ---  
Previous Units: 100 units  
Current Units: 250 units  
Customer Type: Domestic  
Actual Units Consumed: 150 units  
Units Billed: 150 units  
Rate Applied: ₹5.00 per unit  
--- Charges Breakdown ---  
Energy Charges: ₹750.00  
Fixed Charges: ₹50.00  
Customer Charges: ₹20.00  
Electricity Duty: ₹41.00  
-----  
Final Total Bill Amount: ₹861.00
```

Explanation: The final program accurately computes all billing components and presents them in a clear, structured format.

Its readable logic and modular design make it suitable for real-world electricity billing applications and easy future enhancements.