Assignment-3.3

2303A51304

Batch-05

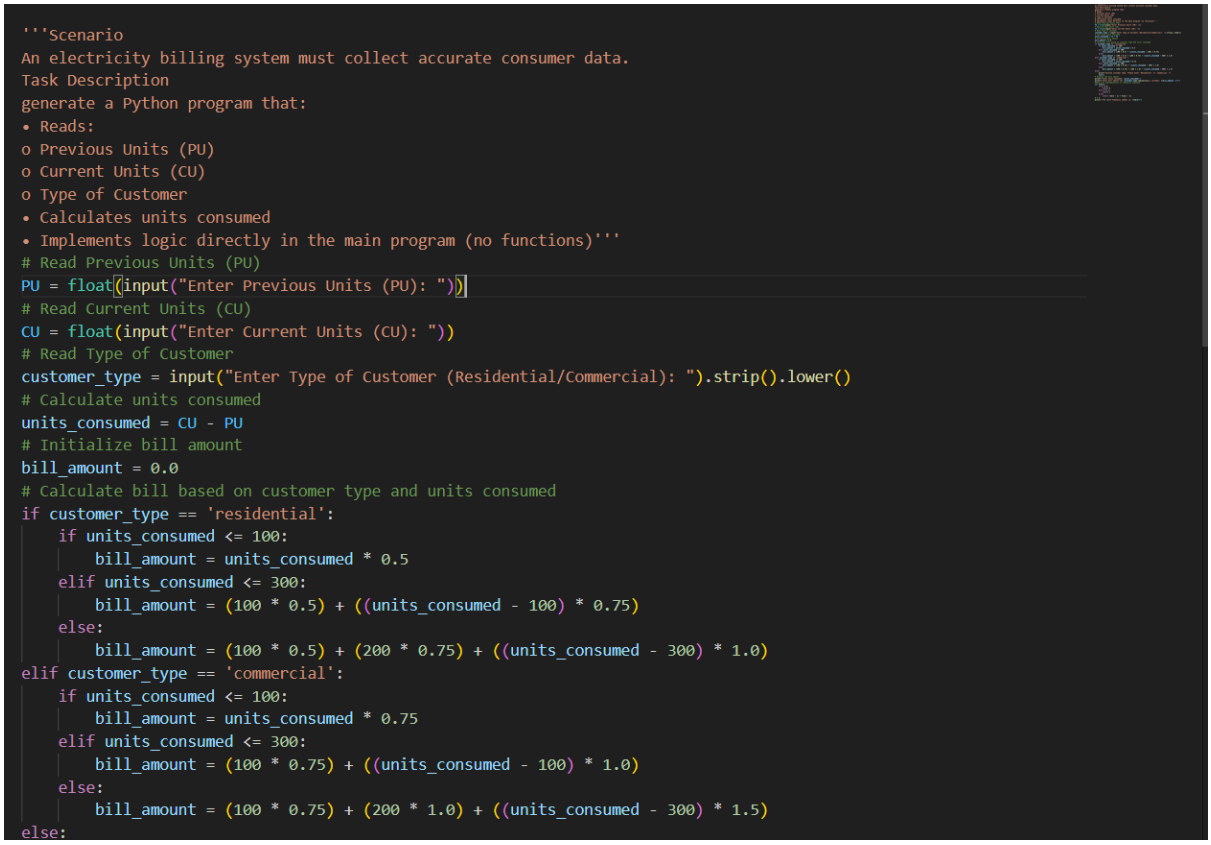
Task01:

Prompt:

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)

Code:



```
'''Scenario
An electricity billing system must collect accurate consumer data.
Task Description
generate a Python program that:
• Reads:
o Previous Units (PU)
o Current Units (CU)
o Type of Customer
• Calculates units consumed
• Implements logic directly in the main program (no functions)'''
# Read Previous Units (PU)
PU = float(input("Enter Previous Units (PU): "))
# Read Current Units (CU)
CU = float(input("Enter Current Units (CU): "))
# Read Type of Customer
customer_type = input("Enter Type of Customer (Residential/Commercial): ").strip().lower()
# Calculate units consumed
units_consumed = CU - PU
# Initialize bill amount
bill_amount = 0.0
# Calculate bill based on customer type and units consumed
if customer_type == 'residential':
    if units_consumed <= 100:
        bill_amount = units_consumed * 0.5
    elif units_consumed <= 300:
        bill_amount = (100 * 0.5) + ((units_consumed - 100) * 0.75)
    else:
        bill_amount = (100 * 0.5) + (200 * 0.75) + ((units_consumed - 300) * 1.0)
elif customer_type == 'commercial':
    if units_consumed <= 100:
        bill_amount = units_consumed * 0.75
    elif units_consumed <= 300:
        bill_amount = (100 * 0.75) + ((units_consumed - 100) * 1.0)
    else:
        bill_amount = (100 * 0.75) + (200 * 1.0) + ((units_consumed - 300) * 1.5)
else:
```

```
        bill_amount = units_consumed * 0.75
    elif units_consumed <= 300:
        bill_amount = (100 * 0.75) + ((units_consumed - 100) * 1.0)
    else:
        bill_amount = (100 * 0.75) + (200 * 1.0) + ((units_consumed - 300) * 1.5)
else:
    print("Invalid customer type. Please enter 'Residential' or 'Commercial'.")
    exit()
# Display the bill amount
print(f"Total units consumed: {units_consumed}")
print(f"Total bill amount for {customer_type.capitalize()} customer: ${bill_amount:.2f}")
#Recursive implementation of fibanocci sequence
def fib(n):
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n - 1) + fib(n - 2)
n = 3
print(f"The {n}th Fibonacci number is: {fib(n)}")
```

Output:

```
he 3th Fibonacci number is: 2
S C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Programs/Python/Python313/python.exe c
/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Asssign_3.3.py
nter Previous Units (PU): 150
nter Current Units (CU): 200
nter Type of Customer (Residential/Commercial): Residential
otal units consumed: 50.0
otal bill amount for Residential customer: $25.00
he 3th Fibonacci number is: 2
S C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

Explanation:

The program reads previous units, current units, and customer type to collect electricity consumption data. It calculates units consumed by subtracting previous units from current units.

Based on customer type (residential or commercial), different slab-wise billing rates are applied.

The total bill amount is computed and displayed along with units consumed.

A recursive Fibonacci function is included to demonstrate recursion by calculating the 3rd Fibonacci number.

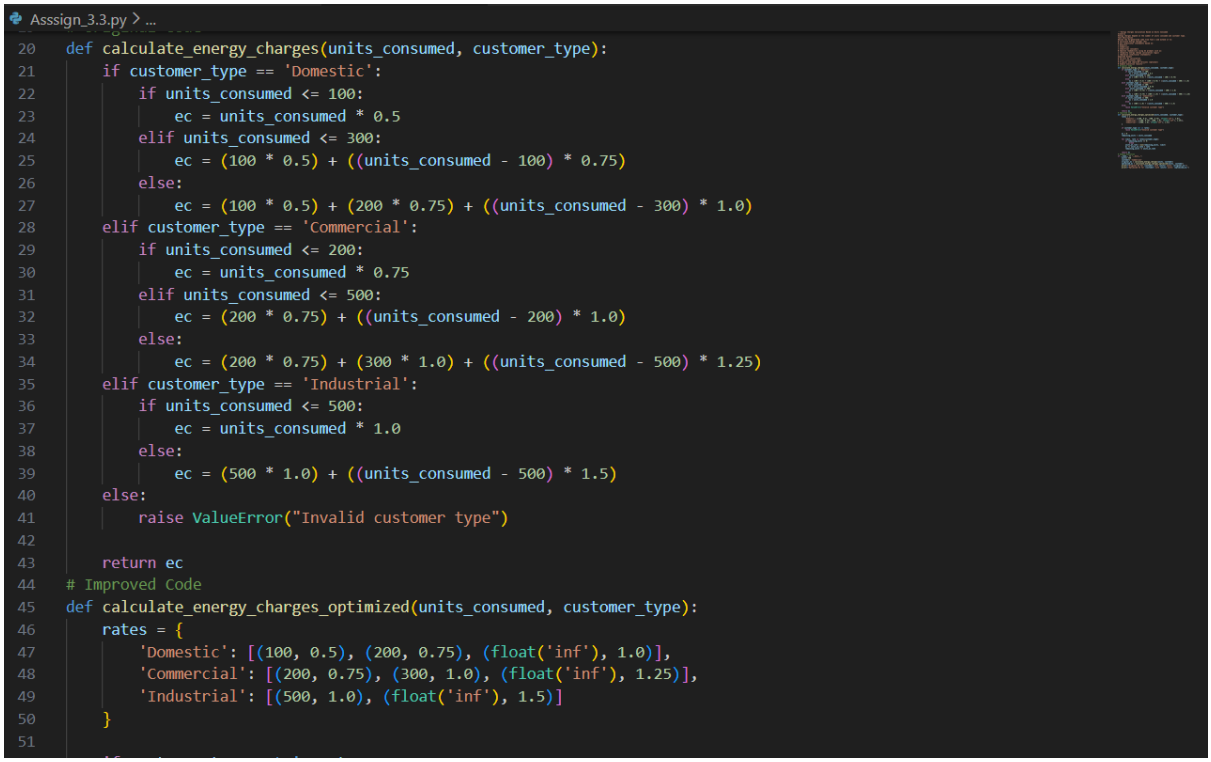
Task02:

Prompt: Write a python code

Calculate Energy Charges (EC) and Use conditional statements based on: Domestic, Commercial, Industrial consumers

Improve readability using AI prompts such as: Simplify energy charge calculation logic, Optimize conditional statements

Code:



Asssign_3.3.py > ...

```python
def calculate_energy_charges(units_consumed, customer_type):
    if customer_type == 'Domestic':
        if units_consumed <= 100:
            ec = units_consumed * 0.5
        elif units_consumed <= 300:
            ec = (100 * 0.5) + ((units_consumed - 100) * 0.75)
        else:
            ec = (100 * 0.5) + (200 * 0.75) + ((units_consumed - 300) * 1.0)
    elif customer_type == 'Commercial':
        if units_consumed <= 200:
            ec = units_consumed * 0.75
        elif units_consumed <= 500:
            ec = (200 * 0.75) + ((units_consumed - 200) * 1.0)
        else:
            ec = (200 * 0.75) + (300 * 1.0) + ((units_consumed - 500) * 1.25)
    elif customer_type == 'Industrial':
        if units_consumed <= 500:
            ec = units_consumed * 1.0
        else:
            ec = (500 * 1.0) + ((units_consumed - 500) * 1.5)
    else:
        raise ValueError("Invalid customer type")

    return ec
# Improved Code
def calculate_energy_charges_optimized(units_consumed, customer_type):
    rates = {
        'Domestic': [(100, 0.5), (200, 0.75), (float('inf'), 1.0)],
        'Commercial': [(200, 0.75), (300, 1.0), (float('inf'), 1.25)],
        'Industrial': [(500, 1.0), (float('inf'), 1.5)]
    }

```

```python
Asssign_3.3.py > ...
    return ec
# Improved Code
def calculate_energy_charges_optimized(units_consumed, customer_type):
    rates = {
        'Domestic': [(100, 0.5), (200, 0.75), (float('inf'), 1.0)],
        'Commercial': [(200, 0.75), (300, 1.0), (float('inf'), 1.25)],
        'Industrial': [(500, 1.0), (float('inf'), 1.5)]
    }

    if customer_type not in rates:
        raise ValueError("Invalid customer type")

    ec = 0
    remaining_units = units_consumed

    for limit, rate in rates[customer_type]:
        if remaining_units <= 0:
            break
        units_at_rate = min(remaining_units, limit)
        ec += units_at_rate * rate
        remaining_units -= units_at_rate

    return ec
# Sample Execution
if __name__ == "__main__":
    units= 450
    customer= 'Commercial'
    original_ec = calculate_energy_charges(units, customer)
    optimized_ec = calculate_energy_charges_optimized(units, customer)
    print(f"Original EC for {customer} with {units} units: {original_ec}")
    print(f"Optimized EC for {customer} with {units} units: {optimized_ec}")
```

Output:

```
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>

:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Asssign_3.3.py
Original EC for Commercial with 450 units: 400.0
Optimized EC for Commercial with 450 units: 400.0
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

Explanation:

The original code uses nested if–elif conditions, making it longer and harder to maintain.

The optimized code stores slab limits and rates in a dictionary, improving readability and structure.

Adding or modifying tariff slabs is easier in the optimized version without changing logic flow. The optimized approach reduces code repetition by using a loop for calculations. Both versions produce the same correct energy charge, but the optimized code is cleaner and scalable.

Task03:

Prompt: Python program that:

Uses user-defined functions to Calculate Energy Charges, Calculate Fixed Charges, Returns calculated values

Code:

```python
# Function to calculate Energy Charges
def calculate_energy_charges(units_consumed, rate_per_unit):
    energy_charges = units_consumed * rate_per_unit
    return energy_charges
# Function to calculate Fixed Charges
def calculate_fixed_charges(fixed_charge_rate):
    return fixed_charge_rate
# Test cases to validate the functions
if __name__ == "__main__":
    # Test case 1
    units = 150
    rate = 0.12
    fixed_charge = 50.0
    ec = calculate_energy_charges(units, rate)
    fc = calculate_fixed_charges(fixed_charge)
    print(f"Test Case 1 - Energy Charges: ${ec:.2f}, Fixed Charges: ${fc:.2f}")

    # Test case 2
    units = 200
    rate = 0.15
    fixed_charge = 75.0
    ec = calculate_energy_charges(units, rate)
    fc = calculate_fixed_charges(fixed_charge)
    print(f"Test Case 2 - Energy Charges: ${ec:.2f}, Fixed Charges: ${fc:.2f}")
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Programs/Python/Python313/py
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Asssign_3.3.py
Test Case 1 - Energy Charges: $18.00, Fixed Charges: $50.00
Test Case 2 - Energy Charges: $30.00, Fixed Charges: $75.00
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

Explanation:

This program demonstrates reusable billing logic using user-defined functions.
The calculate_energy_charges() function computes energy cost by multiplying units consumed with the rate per unit and returns the value.
The calculate_fixed_charges() function returns a fixed charge amount applicable to the

consumer.In the main block, multiple test cases are executed to validate the functions with different inputs.This approach improves code reusability, clarity, and makes billing calculations easy to maintain.

Task04:

Prompt: Write a python code to calculate the FC ,CC,ED, add electricity duty calculation

Code:

```python
def calculate_electricity_bill(units_consumed):
    # Constants
    RATE_PER_UNIT = 5.0  # Cost per unit of electricity
    FIXED_CHARGES = 50.0  # Fixed charges
    CUSTOMER_CHARGES = 20.0  # Customer charges
    ELECTRICITY_DUTY_RATE = 0.05  # 5% Electricity Duty

    # Calculate Energy Charges (EC)
    energy_charges = units_consumed * RATE_PER_UNIT

    # Calculate Electricity Duty (ED)
    electricity_duty = energy_charges * ELECTRICITY_DUTY_RATE

    # Total Bill Calculation
    total_bill = energy_charges + FIXED_CHARGES + CUSTOMER_CHARGES + electricity_duty

    # Print individual charge values
    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): ${electricity_duty:.2f}")
    print(f"Total Bill: ${total_bill:.2f}")
    return total_bill
# Example usage
units = 150  # Example units consumed
calculate_electricity_bill(units)
```

Output:

```
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding> & C:/Users/pooji/AppData/Local/Programs/Python/Python313/python.exe c
:/Users/pooji/OneDrive/Desktop/AI_Assistance_coding/Asssign_3.3.py
Energy Charges (EC): $750.00
Fixed Charges (FC): $50.00
Customer Charges (CC): $20.00
Electricity Duty (ED): $37.50
Total Bill: $857.50
PS C:\Users\pooji\OneDrive\Desktop\AI_Assistance_coding>
```

Explanation: This program calculates a complete electricity bill by including multiple charges. It first computes Energy Charges (EC) based on units consumed and a fixed per-unit rate.
Then it calculates Electricity Duty (ED) as a percentage of the energy charges.
Fixed Charges (FC) and Customer Charges (CC) are added as constant values.
Finally, all charges are summed and displayed clearly to produce the total bill.

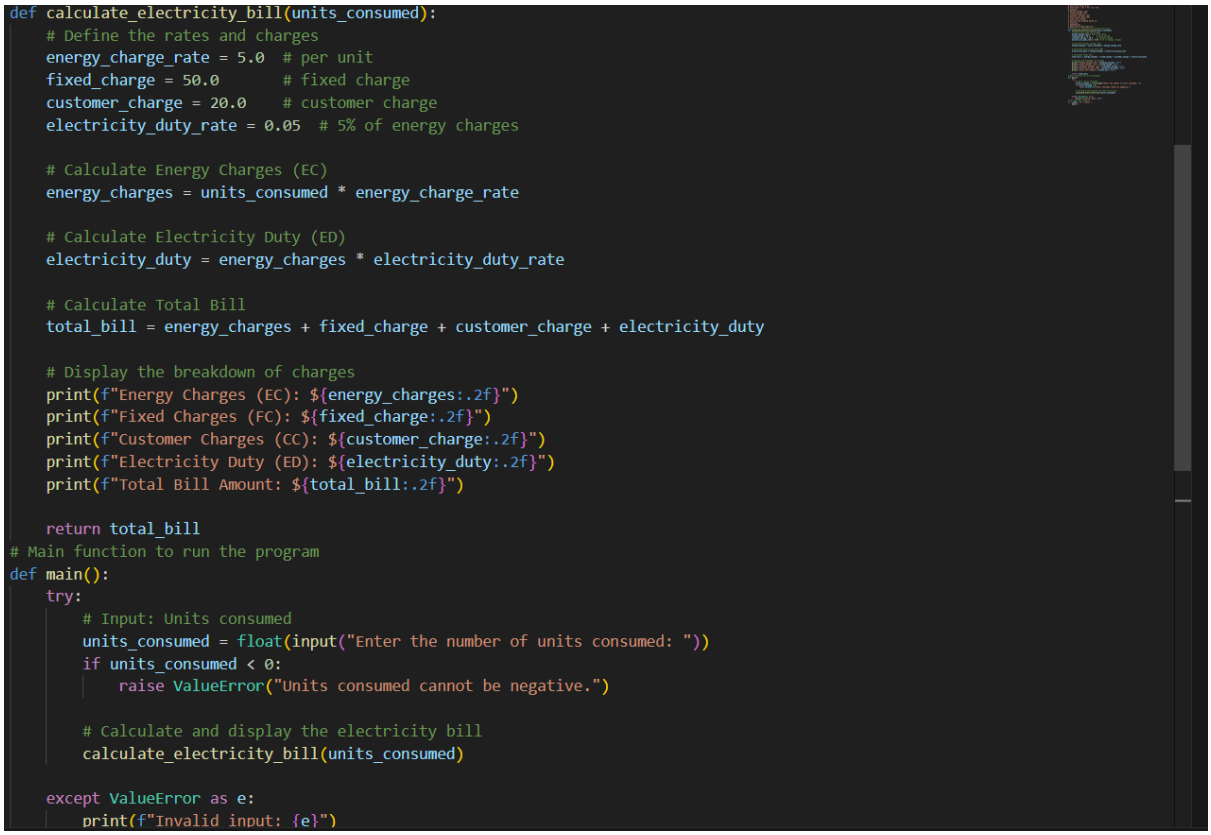
Task05:

Prompt: 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

Code:



```
def calculate_electricity_bill(units_consumed):
    # Define the rates and charges
    energy_charge_rate = 5.0  # per unit
    fixed_charge = 50.0       # fixed charge
    customer_charge = 20.0    # customer charge
    electricity_duty_rate = 0.05  # 5% of energy charges

    # Calculate Energy Charges (EC)
    energy_charges = units_consumed * energy_charge_rate

    # Calculate Electricity Duty (ED)
    electricity_duty = energy_charges * electricity_duty_rate

    # Calculate Total Bill
    total_bill = energy_charges + fixed_charge + customer_charge + electricity_duty

    # Display the breakdown of charges
    print(f"Energy Charges (EC): ${energy_charges:.2f}")
    print(f"Fixed Charges (FC): ${fixed_charge:.2f}")
    print(f"Customer Charges (CC): ${customer_charge:.2f}")
    print(f"Electricity Duty (ED): ${electricity_duty:.2f}")
    print(f"Total Bill Amount: ${total_bill:.2f}")

    return total_bill
# Main function to run the program
def main():
    try:
        # Input: Units consumed
        units_consumed = float(input("Enter the number of units consumed: "))
        if units_consumed < 0:
            raise ValueError("Units consumed cannot be negative.")

        # Calculate and display the electricity bill
        calculate_electricity_bill(units_consumed)

    except ValueError as e:
        print(f"Invalid input: {e}")
```

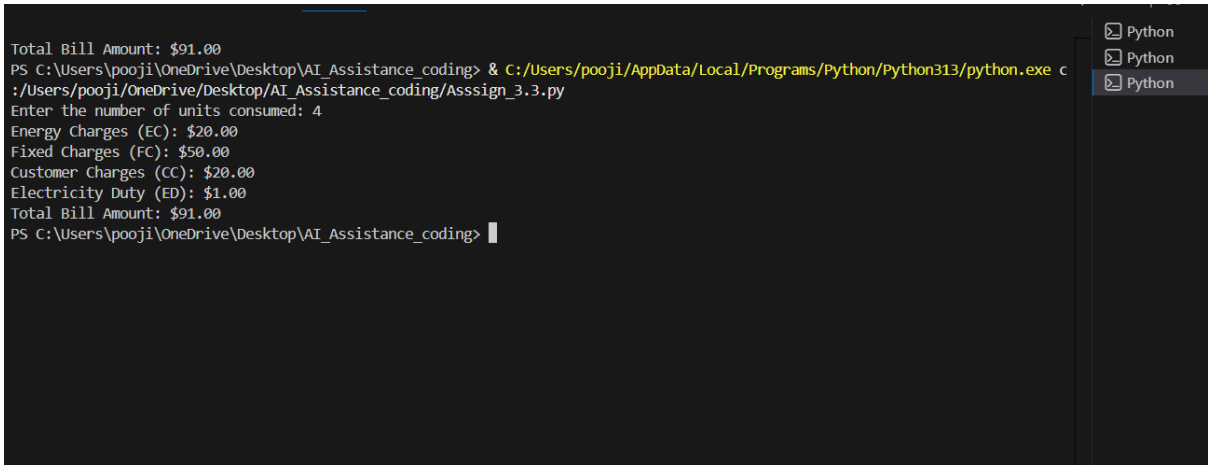
```python
def calculate_electricity_bill(units_consumed):
    print(f"Electricity Duty (ED): ${electricity_duty:.2f}")
    print(f"Total Bill Amount: ${total_bill:.2f}")

    return total_bill
# Main function to run the program
def main():
    try:
        # Input: Units consumed
        units_consumed = float(input("Enter the number of units consumed: "))
        if units_consumed < 0:
            raise ValueError("Units consumed cannot be negative.")

        # Calculate and display the electricity bill
        calculate_electricity_bill(units_consumed)

    except ValueError as e:
        print(f"Invalid input: {e}")
# Run the main function
if __name__ == "__main__":
    main()
```

Output:



Explanation:

This program accurately calculates the total electricity bill by summing Energy Charges, Fixed Charges, Customer Charges, and Electricity Duty.
Clear variable names and step-by-step calculations make the code easy to read and understand.
Input validation ensures realistic values by preventing negative unit entries.
The detailed output displays each charge separately, improving transparency for users.