AI Assistant Coding

Assignment 3.3

Name : N. Eshwar **HT. No :** 2303A52072 **Batch:** 32

Task 1: AI-Generated Logic for Reading Consumer Details

Scenario: An electricity billing system must collect accurate consumer data.

Task Description: Use an AI tool (GitHub Copilot / Gemini) to generate a Python program that:

Reads:

Previous Units (PU)

Current Units (CU)

Type of Customer

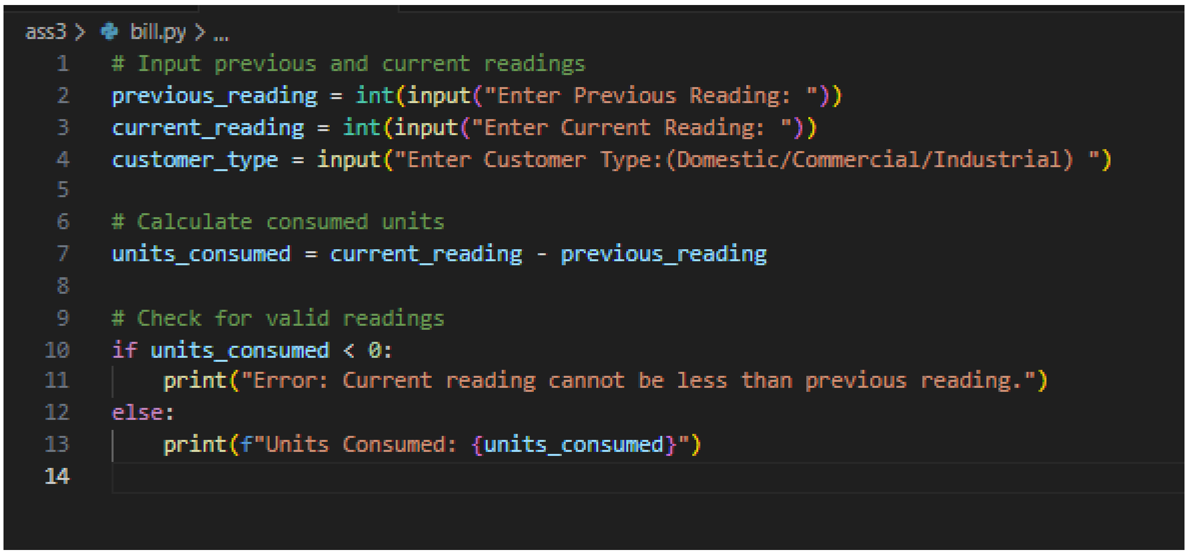
Calculates units consumed

Implements logic directly in the main program (no functions)

Prompt:

write a python program that takes inputs previous units, current units, and customer type and calculate the units consumed without any functions.

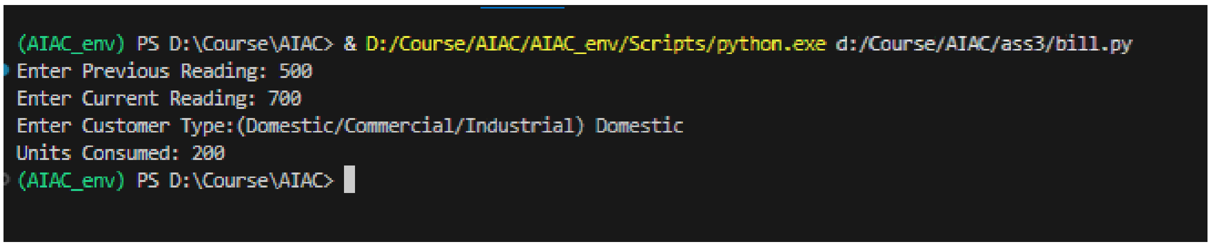
Code:



ass3 > bill.py > ...

```python
# Input previous and current readings
previous_reading = int(input("Enter Previous Reading: "))
current_reading = int(input("Enter Current Reading: "))
customer_type = input("Enter Customer Type:(Domestic/Commercial/Industrial) ")

# Calculate consumed units
units_consumed = current_reading - previous_reading

# Check for valid readings
if units_consumed < 0:
    print("Error: Current reading cannot be less than previous reading.")
else:
    print(f"Units Consumed: {units_consumed}")
```

Output:



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/bill.py
Enter Previous Reading: 500
Enter Current Reading: 700
Enter Customer Type:(Domestic/Commercial/Industrial) Domestic
Units Consumed: 200
(AIAC_env) PS D:\Course\AIAC> 
```

Task 2: Energy Charges Calculation Based on Units Consumed

Scenario: Energy charges depend on the number of units consumed and customer type.

Task Description: Review the AI-generated code from Task 1 and extend it to:

Calculate Energy Charges (EC)

Use conditional statements based on:

Domestic, Commercial, Industrial consumers

Improve readability using AI prompts such as:

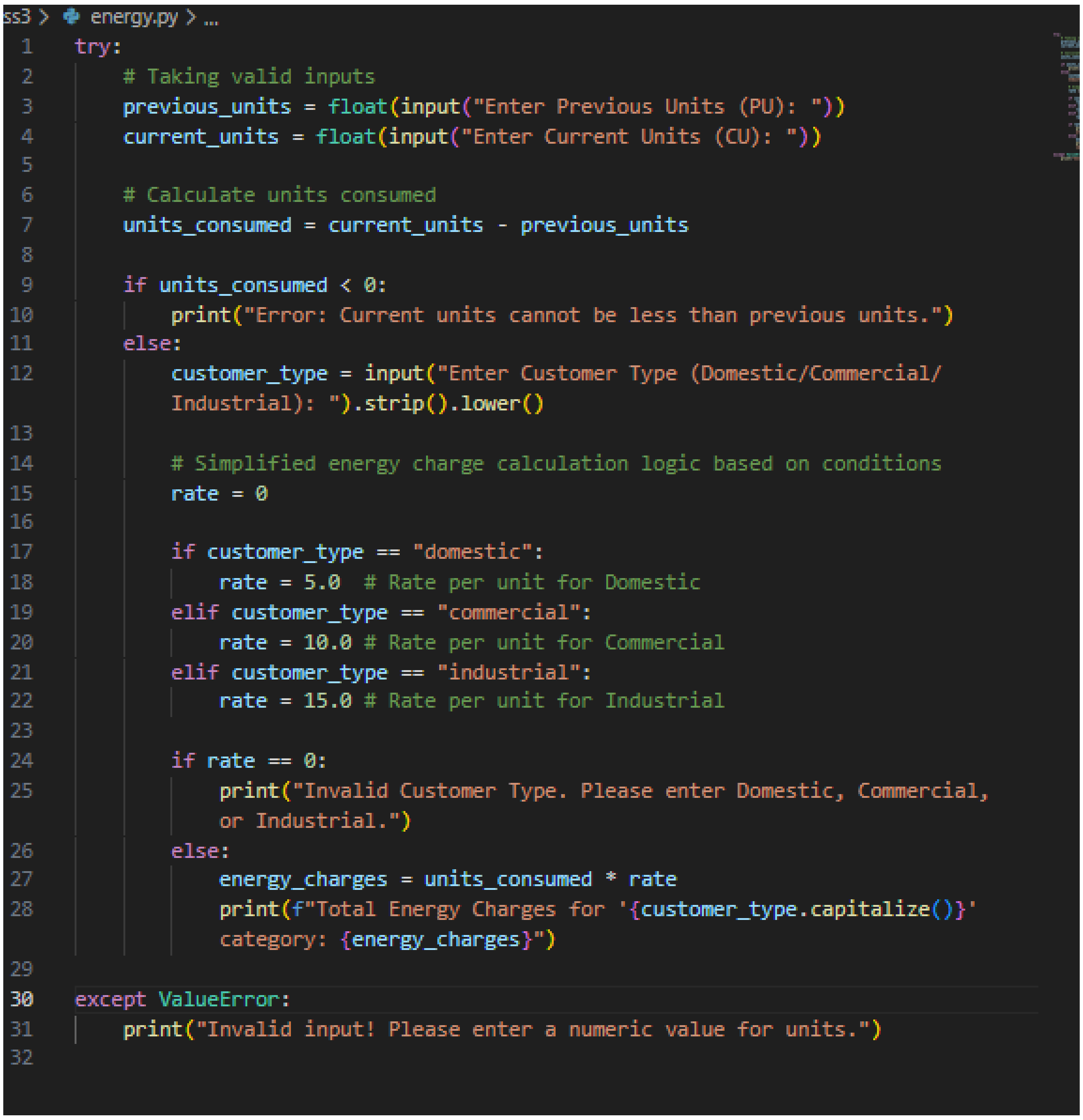
"Simplify energy charge calculation logic"

"Optimize conditional statements"

Prompt:

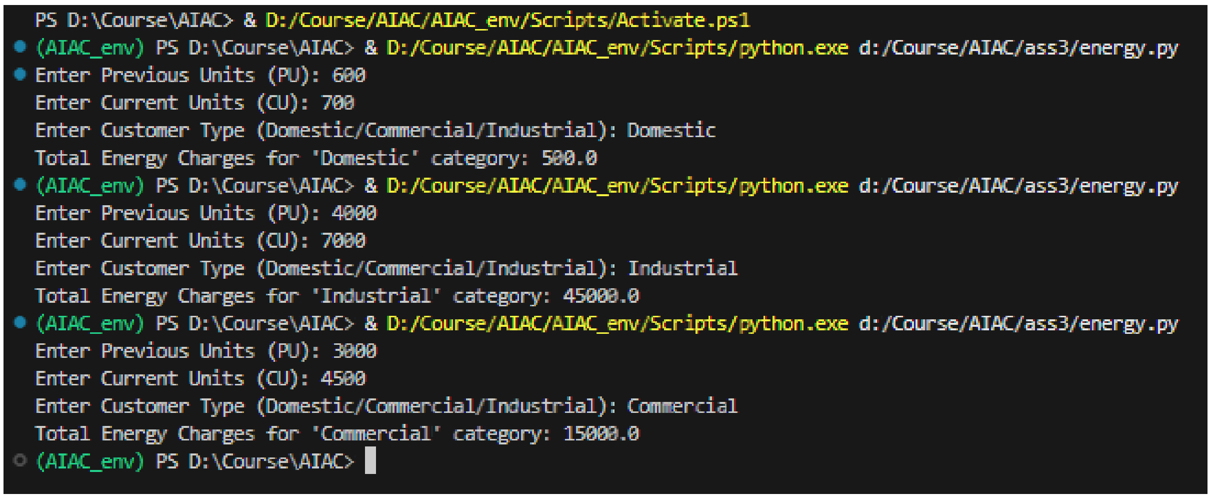
write a python script that takes valid inputs ,PU, CU and calculates energy charges based on units consumed and condition Domestic ,commercial and Industrial and simplify energy charge calculations logic without functions

Code:



```
ss3 > energy.py > ...
try:
    # Taking valid inputs
    previous_units = float(input("Enter Previous Units (PU): "))
    current_units = float(input("Enter Current Units (CU): "))

    # Calculate units consumed
    units_consumed = current_units - previous_units

    if units_consumed < 0:
        print("Error: Current units cannot be less than previous units.")
    else:
        customer_type = input("Enter Customer Type (Domestic/Commercial/
        Industrial): ").strip().lower()

        # Simplified energy charge calculation logic based on conditions
        rate = 0

        if customer_type == "domestic":
            rate = 5.0  # Rate per unit for Domestic
        elif customer_type == "commercial":
            rate = 10.0 # Rate per unit for Commercial
        elif customer_type == "industrial":
            rate = 15.0 # Rate per unit for Industrial

        if rate == 0:
            print("Invalid Customer Type. Please enter Domestic, Commercial,
            or Industrial.")
        else:
            energy_charges = units_consumed * rate
            print(f"Total Energy Charges for '{customer_type.capitalize()}'
            category: {energy_charges}")

except ValueError:
    print("Invalid input! Please enter a numeric value for units.")
```

Output:



```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 600
Enter Current Units (CU): 700
Enter Customer Type (Domestic/Commercial/Industrial): Domestic
Total Energy Charges for 'Domestic' category: 500.0
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 4000
Enter Current Units (CU): 7000
Enter Customer Type (Domestic/Commercial/Industrial): Industrial
Total Energy Charges for 'Industrial' category: 45000.0
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 3000
Enter Current Units (CU): 4500
Enter Customer Type (Domestic/Commercial/Industrial): Commercial
Total Energy Charges for 'Commercial' category: 15000.0
(AIAC_env) PS D:\Course\AIAC>
```

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

Scenario: Billing logic must be reusable for multiple consumers.

Task Description

Use AI assistance to generate a Python program that:

Uses user-defined functions to:

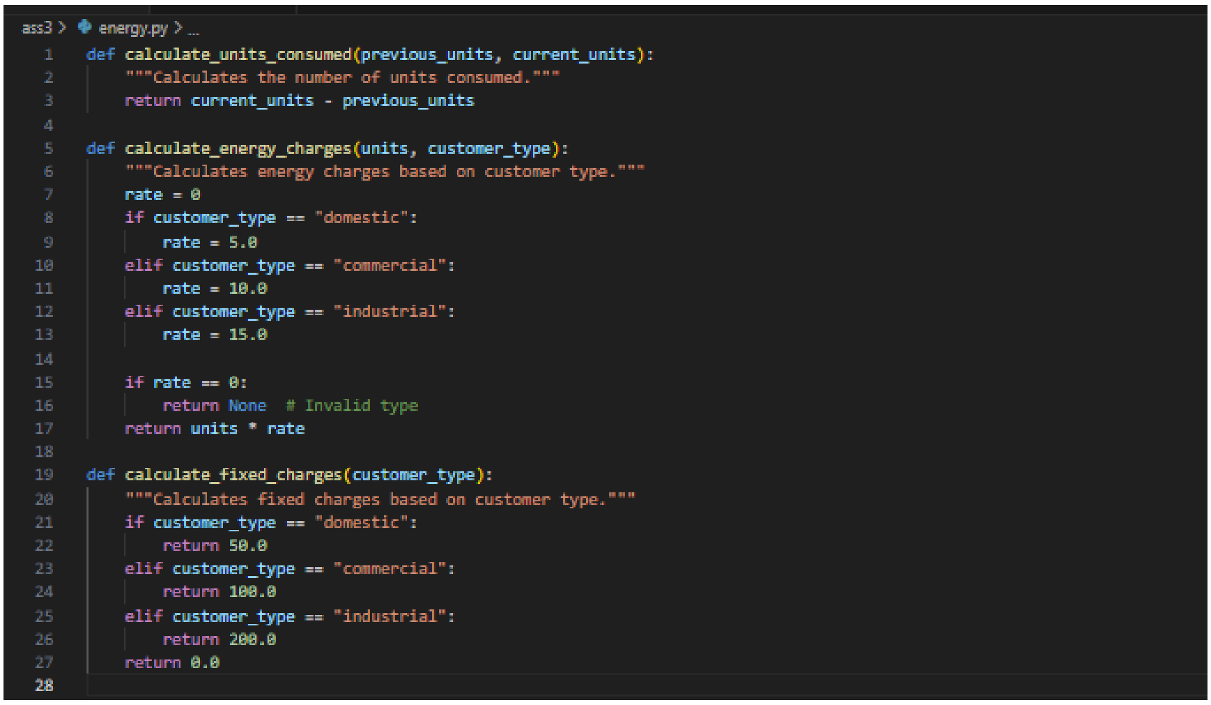
Calculate Energy Charges, Calculate Fixed Charges, Returns calculated values

Includes meaningful comments

Prompt:

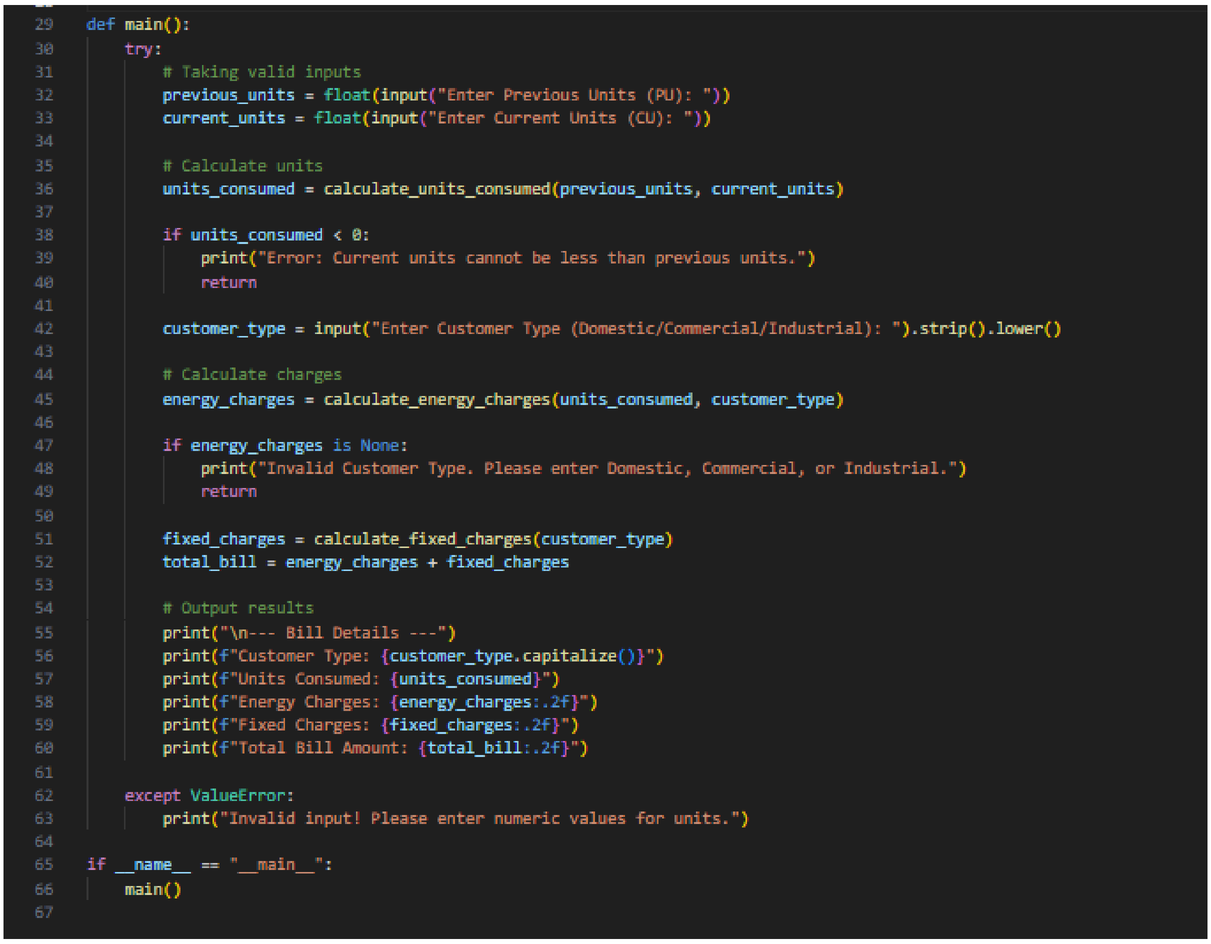
write a python program with modular design that calculates Billing logic calculate energy charges , fixed charges returns calculated values using functions

Code:



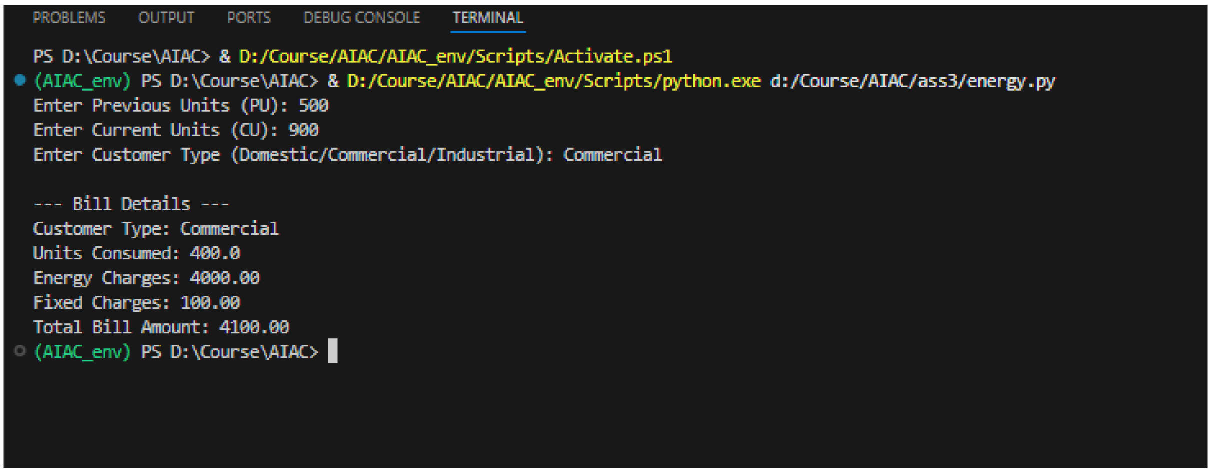
ass3 > energy.py > ...

```python
def calculate_units_consumed(previous_units, current_units):
    """Calculates the number of units consumed."""
    return current_units - previous_units

def calculate_energy_charges(units, customer_type):
    """Calculates energy charges based on customer type."""
    rate = 0
    if customer_type == "domestic":
        rate = 5.0
    elif customer_type == "commercial":
        rate = 10.0
    elif customer_type == "industrial":
        rate = 15.0

    if rate == 0:
        return None  # Invalid type
    return units * rate

def calculate_fixed_charges(customer_type):
    """Calculates fixed charges based on customer type."""
    if customer_type == "domestic":
        return 50.0
    elif customer_type == "commercial":
        return 100.0
    elif customer_type == "industrial":
        return 200.0
    return 0.0
```



```python
def main():
    try:
        # Taking valid inputs
        previous_units = float(input("Enter Previous Units (PU): "))
        current_units = float(input("Enter Current Units (CU): "))

        # Calculate units
        units_consumed = calculate_units_consumed(previous_units, current_units)

        if units_consumed < 0:
            print("Error: Current units cannot be less than previous units.")
            return

        customer_type = input("Enter Customer Type (Domestic/Commercial/Industrial): ").strip().lower()

        # Calculate charges
        energy_charges = calculate_energy_charges(units_consumed, customer_type)

        if energy_charges is None:
            print("Invalid Customer Type. Please enter Domestic, Commercial, or Industrial.")
            return

        fixed_charges = calculate_fixed_charges(customer_type)
        total_bill = energy_charges + fixed_charges

        # Output results
        print("\n--- Bill Details ---")
        print(f"Customer Type: {customer_type.capitalize()}")
        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}")

    except ValueError:
        print("Invalid input! Please enter numeric values for units.")

if __name__ == "__main__":
    main()
```

Output:



PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL

```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 500
Enter Current Units (CU): 900
Enter Customer Type (Domestic/Commercial/Industrial): Commercial

--- Bill Details ---
Customer Type: Commercial
Units Consumed: 400.0
Energy Charges: 4000.00
Fixed Charges: 100.00
Total Bill Amount: 4100.00
(AIAC_env) PS D:\Course\AIAC>
```

Task 4: Calculation of Additional Charges

Scenario: Electricity bills include multiple additional charges.

Task Description

Extend the program to calculate:

FC -Fixed Charges, CC-Customer Charges, ED - Electricity Duty (percentage of EC)

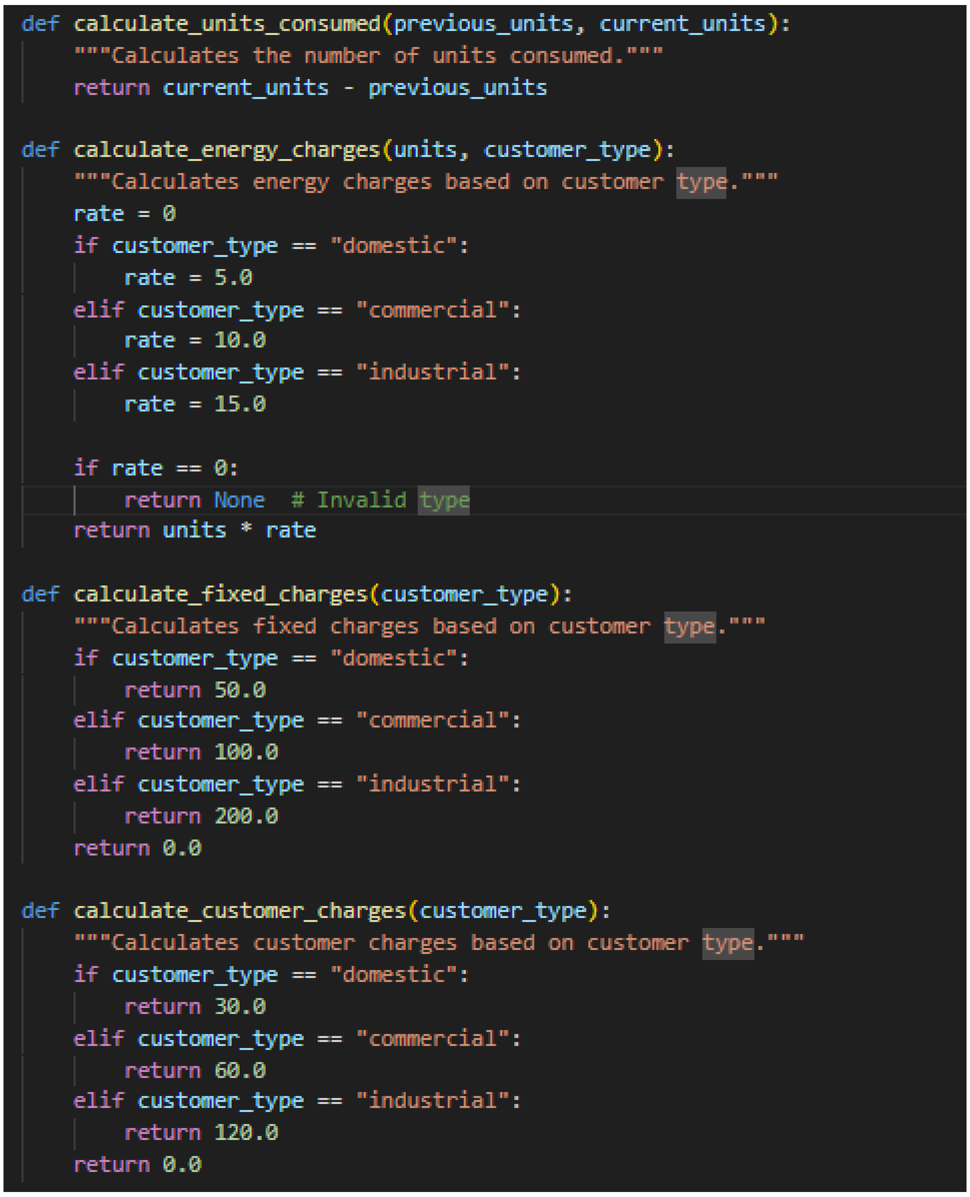
Use AI prompts like:

· "Add electricity duty calculation", "Improve billing accuracy".

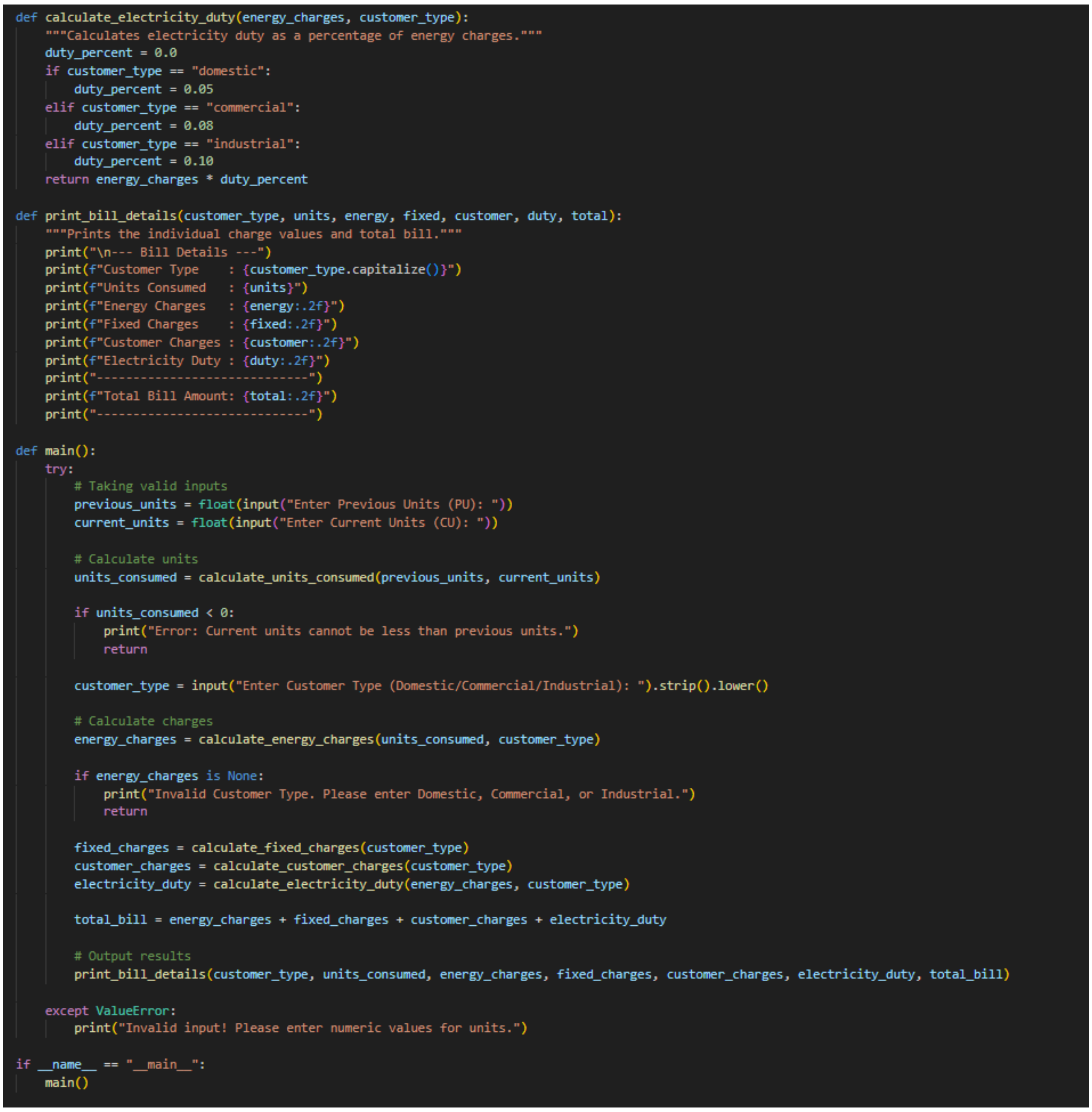
Prompt:

write a python program with modular design that calculates Billing logic calculate energy charges , calculate fixed charges customer charges Electricity duty (Percentage of EC) and print individual charges returns calculated values using functions

Code:

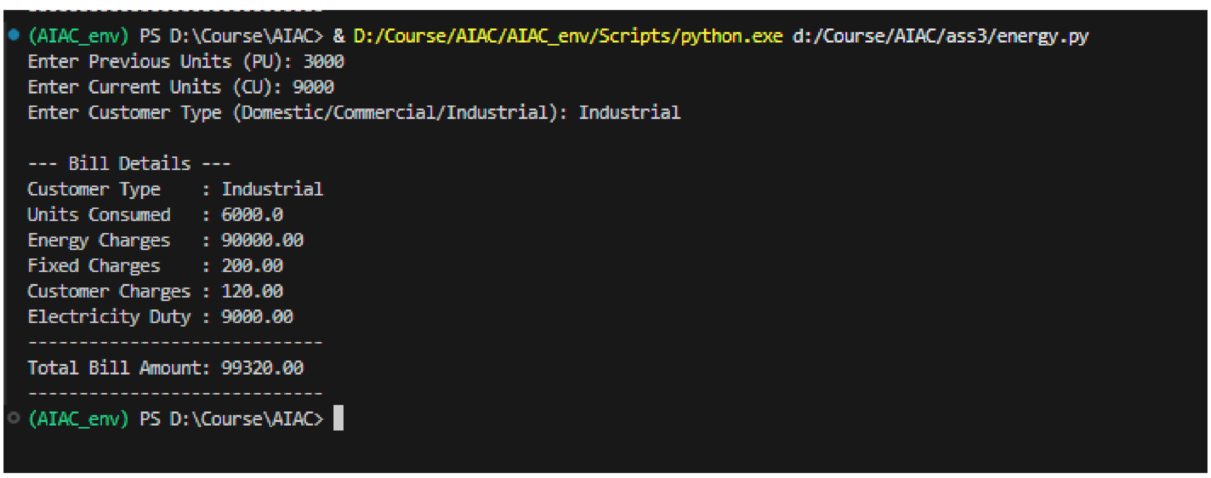


```
def calculate_units_consumed(previous_units, current_units):
    """Calculates the number of units consumed."""
    return current_units - previous_units

def calculate_energy_charges(units, customer_type):
    """Calculates energy charges based on customer type."""
    rate = 0
    if customer_type == "domestic":
        rate = 5.0
    elif customer_type == "commercial":
        rate = 10.0
    elif customer_type == "industrial":
        rate = 15.0

    if rate == 0:
        return None  # Invalid type
    return units * rate

def calculate_fixed_charges(customer_type):
    """Calculates fixed charges based on customer type."""
    if customer_type == "domestic":
        return 50.0
    elif customer_type == "commercial":
        return 100.0
    elif customer_type == "industrial":
        return 200.0
    return 0.0

def calculate_customer_charges(customer_type):
    """Calculates customer charges based on customer type."""
    if customer_type == "domestic":
        return 30.0
    elif customer_type == "commercial":
        return 60.0
    elif customer_type == "industrial":
        return 120.0
    return 0.0
```

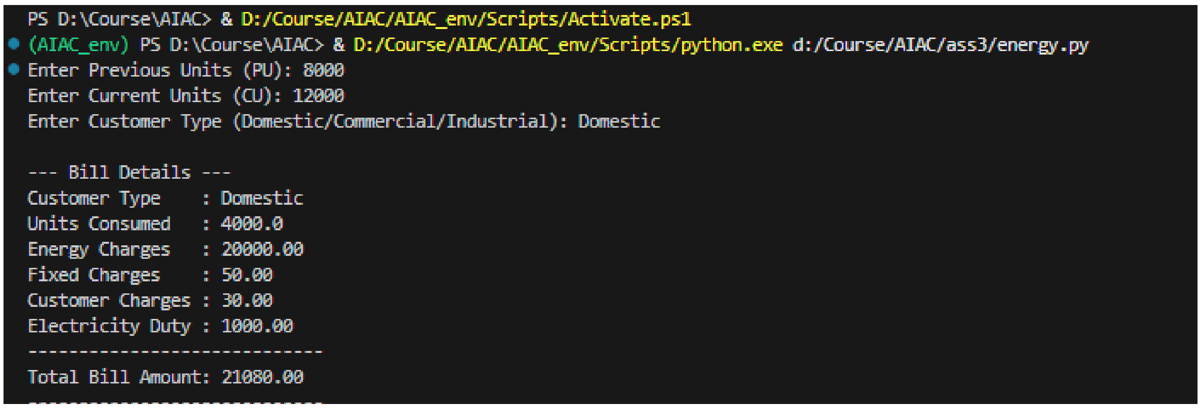


```python
def calculate_electricity_duty(energy_charges, customer_type):
    """Calculates electricity duty as a percentage of energy charges."""
    duty_percent = 0.0
    if customer_type == "domestic":
        duty_percent = 0.05
    elif customer_type == "commercial":
        duty_percent = 0.08
    elif customer_type == "industrial":
        duty_percent = 0.10
    return energy_charges * duty_percent

def print_bill_details(customer_type, units, energy, fixed, customer, duty, total):
    """Prints the individual charge values and total bill."""
    print("\n--- Bill Details ---")
    print(f"Customer Type    : {customer_type.capitalize()}")
    print(f"Units Consumed   : {units}")
    print(f"Energy Charges   : {energy:.2f}")
    print(f"Fixed Charges    : {fixed:.2f}")
    print(f"Customer Charges : {customer:.2f}")
    print(f"Electricity Duty : {duty:.2f}")
    print("-----------------------------")
    print(f"Total Bill Amount: {total:.2f}")
    print("-----------------------------")

def main():
    try:
        # Taking valid inputs
        previous_units = float(input("Enter Previous Units (PU): "))
        current_units = float(input("Enter Current Units (CU): "))

        # Calculate units
        units_consumed = calculate_units_consumed(previous_units, current_units)

        if units_consumed < 0:
            print("Error: Current units cannot be less than previous units.")
            return

        customer_type = input("Enter Customer Type (Domestic/Commercial/Industrial): ").strip().lower()

        # Calculate charges
        energy_charges = calculate_energy_charges(units_consumed, customer_type)

        if energy_charges is None:
            print("Invalid Customer Type. Please enter Domestic, Commercial, or Industrial.")
            return

        fixed_charges = calculate_fixed_charges(customer_type)
        customer_charges = calculate_customer_charges(customer_type)
        electricity_duty = calculate_electricity_duty(energy_charges, customer_type)

        total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty

        # Output results
        print_bill_details(customer_type, units_consumed, energy_charges, fixed_charges, customer_charges, electricity_duty, total_bill)

    except ValueError:
        print("Invalid input! Please enter numeric values for units.")

if __name__ == "__main__":
    main()
```

Output:



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 3000
Enter Current Units (CU): 9000
Enter Customer Type (Domestic/Commercial/Industrial): Industrial

--- Bill Details ---
Customer Type    : Industrial
Units Consumed   : 6000.0
Energy Charges   : 90000.00
Fixed Charges    : 200.00
Customer Charges : 120.00
Electricity Duty : 9000.00
-----------------------------
Total Bill Amount: 99320.00
-----------------------------
(AIAC_env) PS D:\Course\AIAC>
```



```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 8000
Enter Current Units (CU): 12000
Enter Customer Type (Domestic/Commercial/Industrial): Domestic

--- Bill Details ---
Customer Type    : Domestic
Units Consumed   : 4000.0
Energy Charges   : 20000.00
Fixed Charges    : 50.00
Customer Charges : 30.00
Electricity Duty : 1000.00
-----------------------------
Total Bill Amount: 21080.00
```

Task 5: Final Bill Generation and Output Analysis

Scenario: The final electricity bill must present all values clearly.

Task Description

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

Analyze the program based on:

Accuracy

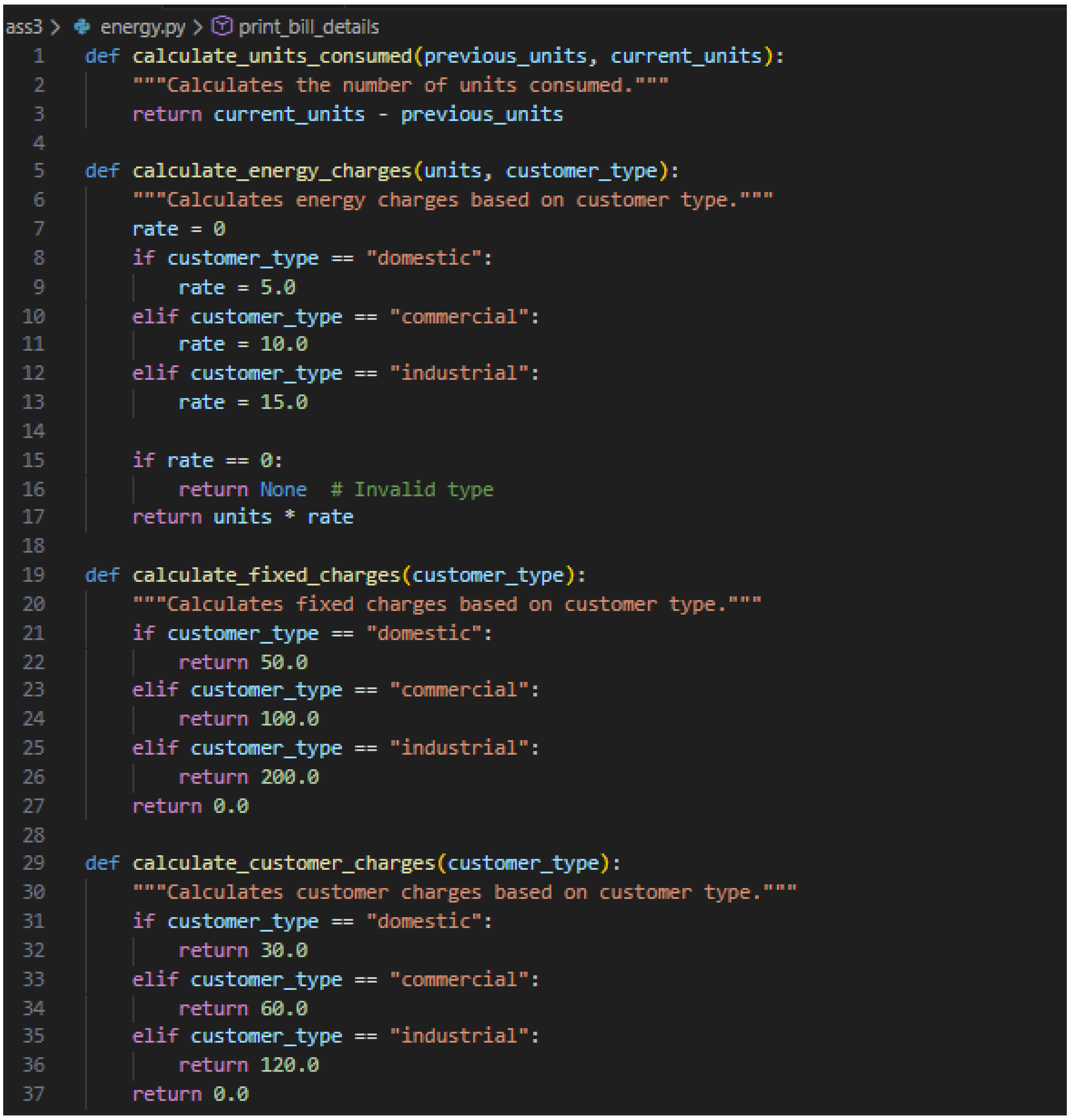
Readability

Real-world applicability

Prompt:

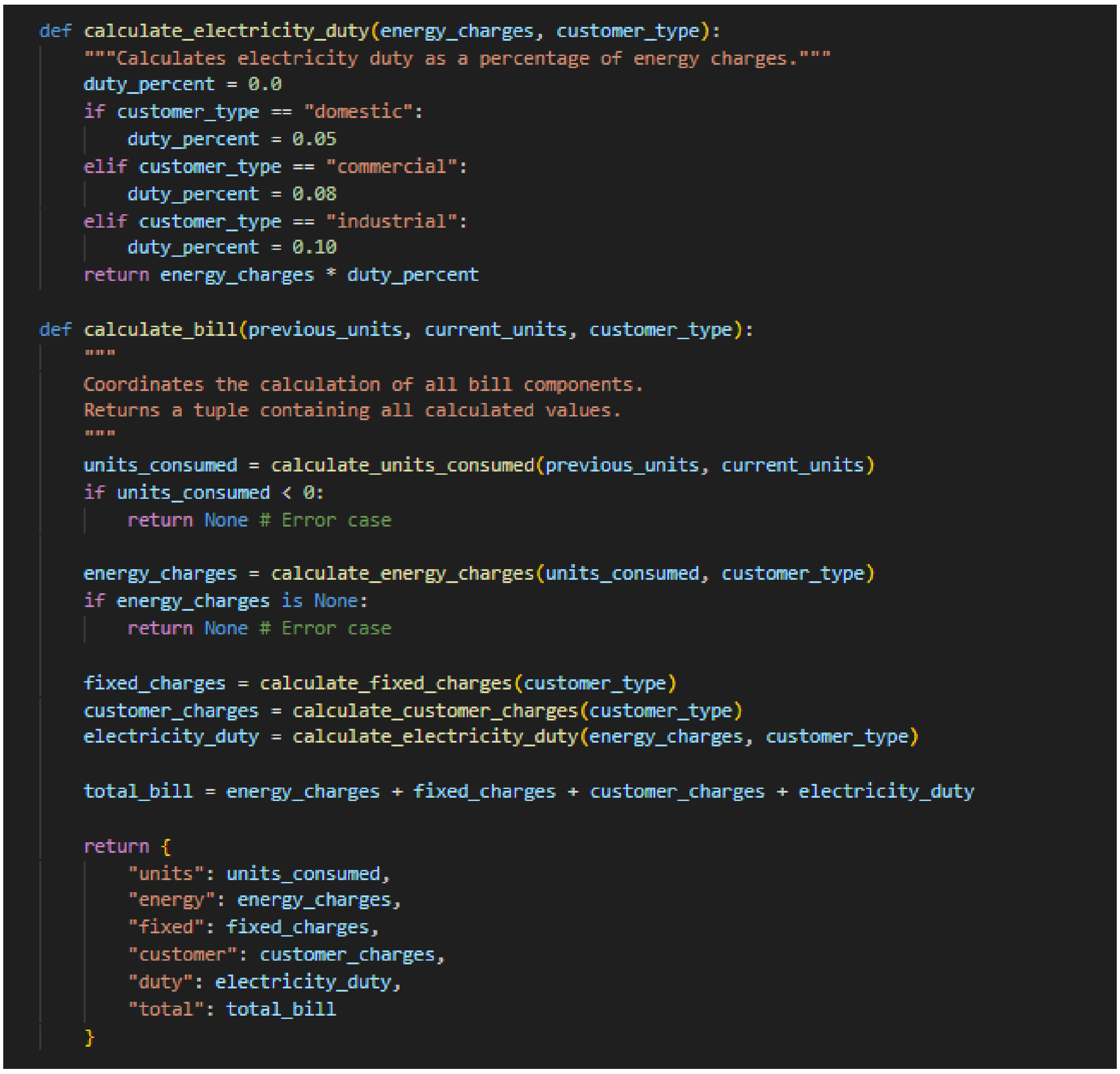
write a python program with modular design that calculates Billing logic calculate energy charges , calculate fixed charges customer charges Electricity duty (Percentage of EC) and print individual charges returns calculated values total bill and print every values using functions

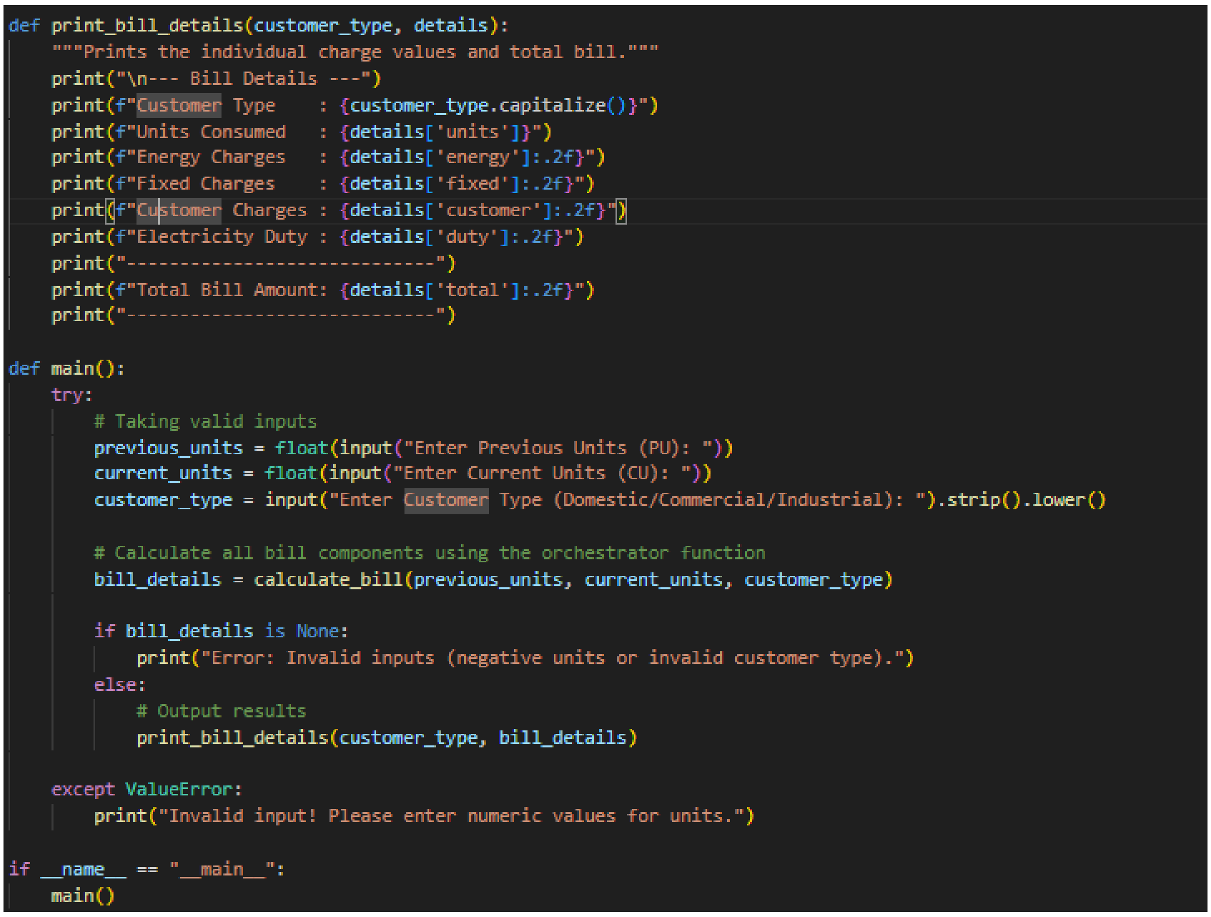
Code:



ass3 > energy.py > print_bill_details

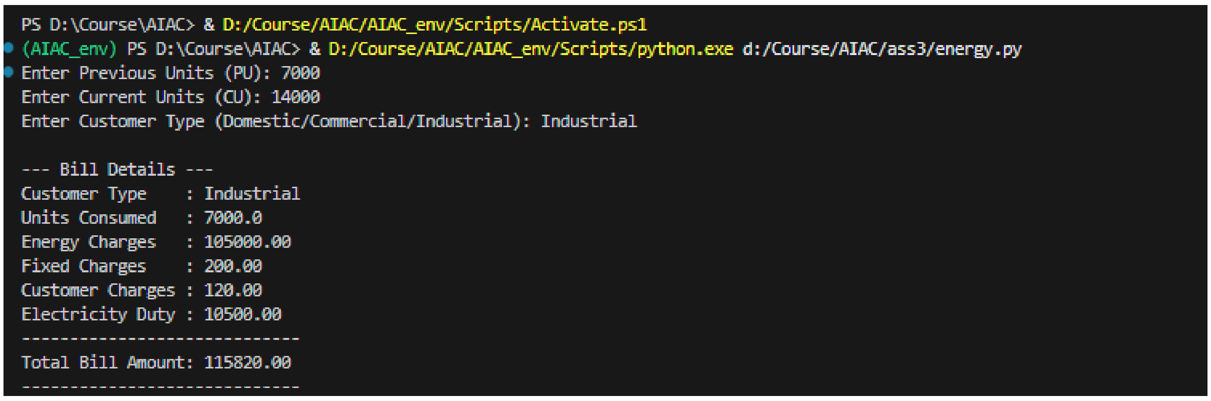
```python
def calculate_units_consumed(previous_units, current_units):
    """Calculates the number of units consumed."""
    return current_units - previous_units

def calculate_energy_charges(units, customer_type):
    """Calculates energy charges based on customer type."""
    rate = 0
    if customer_type == "domestic":
        rate = 5.0
    elif customer_type == "commercial":
        rate = 10.0
    elif customer_type == "industrial":
        rate = 15.0

    if rate == 0:
        return None  # Invalid type
    return units * rate

def calculate_fixed_charges(customer_type):
    """Calculates fixed charges based on customer type."""
    if customer_type == "domestic":
        return 50.0
    elif customer_type == "commercial":
        return 100.0
    elif customer_type == "industrial":
        return 200.0
    return 0.0

def calculate_customer_charges(customer_type):
    """Calculates customer charges based on customer type."""
    if customer_type == "domestic":
        return 30.0
    elif customer_type == "commercial":
        return 60.0
    elif customer_type == "industrial":
        return 120.0
    return 0.0
```


```python
def calculate_electricity_duty(energy_charges, customer_type):
    """Calculates electricity duty as a percentage of energy charges."""
    duty_percent = 0.0
    if customer_type == "domestic":
        duty_percent = 0.05
    elif customer_type == "commercial":
        duty_percent = 0.08
    elif customer_type == "industrial":
        duty_percent = 0.10
    return energy_charges * duty_percent

def calculate_bill(previous_units, current_units, customer_type):
    """
    Coordinates the calculation of all bill components.
    Returns a tuple containing all calculated values.
    """
    units_consumed = calculate_units_consumed(previous_units, current_units)
    if units_consumed < 0:
        return None # Error case

    energy_charges = calculate_energy_charges(units_consumed, customer_type)
    if energy_charges is None:
        return None # Error case

    fixed_charges = calculate_fixed_charges(customer_type)
    customer_charges = calculate_customer_charges(customer_type)
    electricity_duty = calculate_electricity_duty(energy_charges, customer_type)

    total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty

    return {
        "units": units_consumed,
        "energy": energy_charges,
        "fixed": fixed_charges,
        "customer": customer_charges,
        "duty": electricity_duty,
        "total": total_bill
    }
```

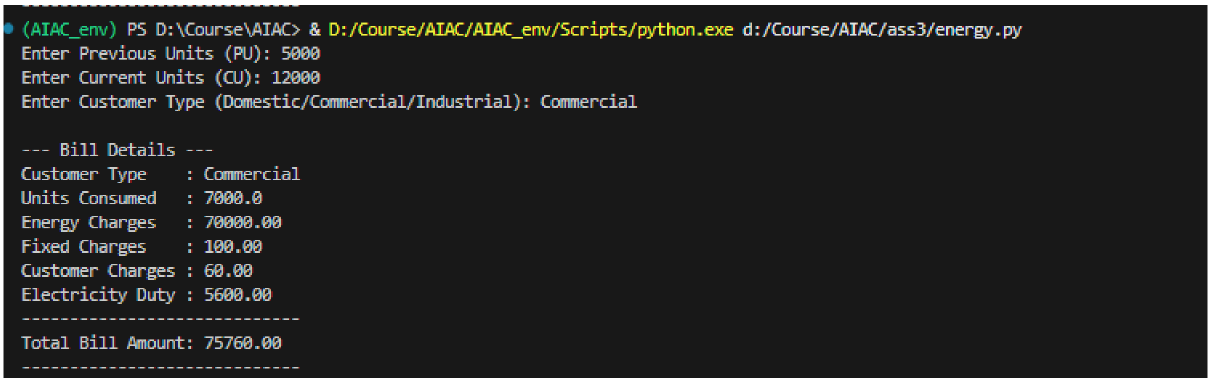


```python
def print_bill_details(customer_type, details):
    """Prints the individual charge values and total bill."""
    print("\n--- Bill Details ---")
    print(f"Customer Type     : {customer_type.capitalize()}")
    print(f"Units Consumed    : {details['units']}")
    print(f"Energy Charges    : {details['energy']:.2f}")
    print(f"Fixed Charges     : {details['fixed']:.2f}")
    print(f"Customer Charges : {details['customer']:.2f}")
    print(f"Electricity Duty : {details['duty']:.2f}")
    print("-----------------------------")
    print(f"Total Bill Amount: {details['total']:.2f}")
    print("-----------------------------")

def main():
    try:
        # Taking valid inputs
        previous_units = float(input("Enter Previous Units (PU): "))
        current_units = float(input("Enter Current Units (CU): "))
        customer_type = input("Enter Customer Type (Domestic/Commercial/Industrial): ").strip().lower()

        # Calculate all bill components using the orchestrator function
        bill_details = calculate_bill(previous_units, current_units, customer_type)

        if bill_details is None:
            print("Error: Invalid inputs (negative units or invalid customer type).")
        else:
            # Output results
            print_bill_details(customer_type, bill_details)

    except ValueError:
        print("Invalid input! Please enter numeric values for units.")

if __name__ == "__main__":
    main()
```

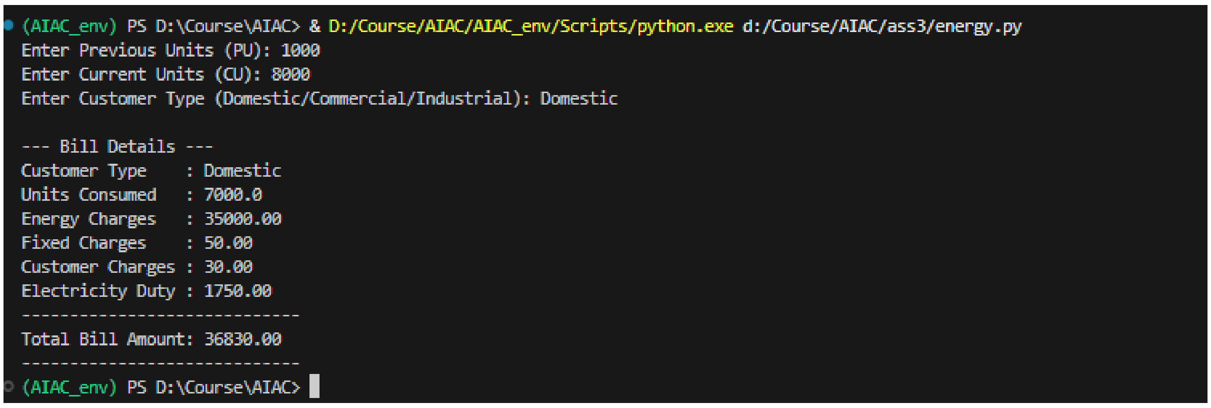
Output:



```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 7000
Enter Current Units (CU): 14000
Enter Customer Type (Domestic/Commercial/Industrial): Industrial

--- Bill Details ---
Customer Type     : Industrial
Units Consumed    : 7000.0
Energy Charges    : 105000.00
Fixed Charges     : 200.00
Customer Charges : 120.00
Electricity Duty : 10500.00
-----------------------------
Total Bill Amount: 115820.00
-----------------------------
```



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 5000
Enter Current Units (CU): 12000
Enter Customer Type (Domestic/Commercial/Industrial): Commercial

--- Bill Details ---
Customer Type     : Commercial
Units Consumed    : 7000.0
Energy Charges    : 70000.00
Fixed Charges     : 100.00
Customer Charges : 60.00
Electricity Duty : 5600.00
-----------------------------
Total Bill Amount: 75760.00
-----------------------------
```



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/ass3/energy.py
Enter Previous Units (PU): 1000
Enter Current Units (CU): 8000
Enter Customer Type (Domestic/Commercial/Industrial): Domestic

--- Bill Details ---
Customer Type     : Domestic
Units Consumed    : 7000.0
Energy Charges    : 35000.00
Fixed Charges     : 50.00
Customer Charges : 30.00
Electricity Duty : 1750.00
-----------------------------
Total Bill Amount: 36830.00
-----------------------------
(AIAC_env) PS D:\Course\AIAC>
```

Analysis:

The program accurately calculates all components of the electricity bill, including Energy Charges, Fixed Charges, Customer Charges, and Electricity Duty, and correctly computes the total bill amount. The code is readable due to clear variable names, structured calculations, and formatted output. This billing logic closely matches real-world electricity billing systems, making the program practical and easy to extend for different tariffs or customer types.