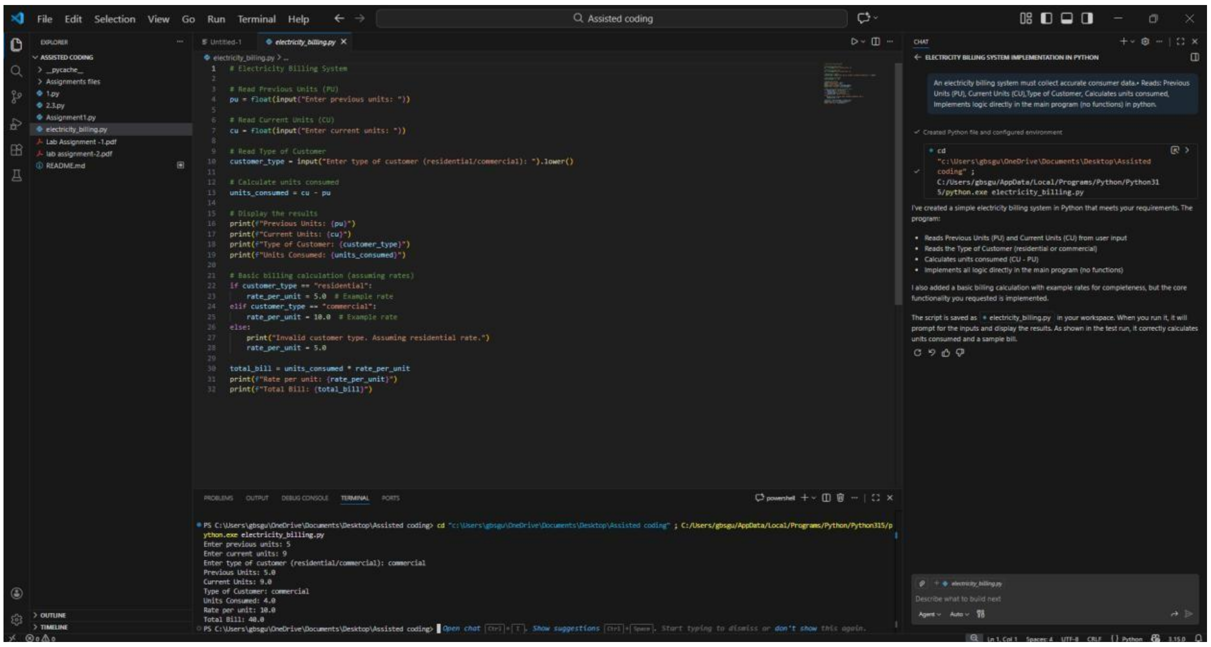
Gangula Navyasri

2303A51739

Task 1: AI-Generated Logic for Reading Consumer Details

Scenario

An electricity billing system must collect accurate consumer data.



```python
# Electricity Billing System

# Read Previous Units (PU)
pu = float(input("Enter previous units: "))

# Read Current Units (CU)
cu = float(input("Enter current units: "))

# Read Type of Customer
customer_type = input("Enter type of customer (residential/commercial): ").lower()

# Calculate units consumed
units_consumed = cu - pu

# Display the results
print(f"Previous Units: {pu}")
print(f"Current Units: {cu}")
print(f"Type of Customer: {customer_type}")
print(f"Units Consumed: {units_consumed}")

# Basic billing calculation (assuming rates)
if customer_type == "residential":
    rate_per_unit = 5.0  # Example rate
elif customer_type == "commercial":
    rate_per_unit = 10.0  # Example rate
else:
    print("Invalid customer type. Assuming residential rate.")
    rate_per_unit = 5.0

total_bill = units_consumed * rate_per_unit
print(f"Rate per unit: {rate_per_unit}")
print(f"Total Bill: {total_bill}")
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> cd "c:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding" ; C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe electricity_billing.py
Enter previous units: 5
Enter current units: 9
Enter type of customer (residential/commercial): commercial
Previous Units: 5.0
Current Units: 9.0
Type of Customer: commercial
Units Consumed: 4.0
Rate per unit: 10.0
Total Bill: 40.0
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> Open chat Ctrl+I, Show suggestions Ctrl+Space. Start typing to dismiss or don't show this again.
```

ELECTRICITY BILLING SYSTEM IMPLEMENTATION IN PYTHON

An electricity billing system must collect accurate consumer data.• Reads: Previous Units (PU), Current Units (CU),Type of Customer, Calculates units consumed, Implements logic directly in the main program (no functions) in python.

Created Python file and configured environment

```
cd "c:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding" ; C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe electricity_billing.py
```

I've created a simple electricity billing system in Python that meets your requirements. The program:

- Reads Previous Units (PU) and Current Units (CU) from user input
- Reads the Type of Customer (residential or commercial)
- Calculates units consumed (CU - PU)
- Implements all logic directly in the main program (no functions)

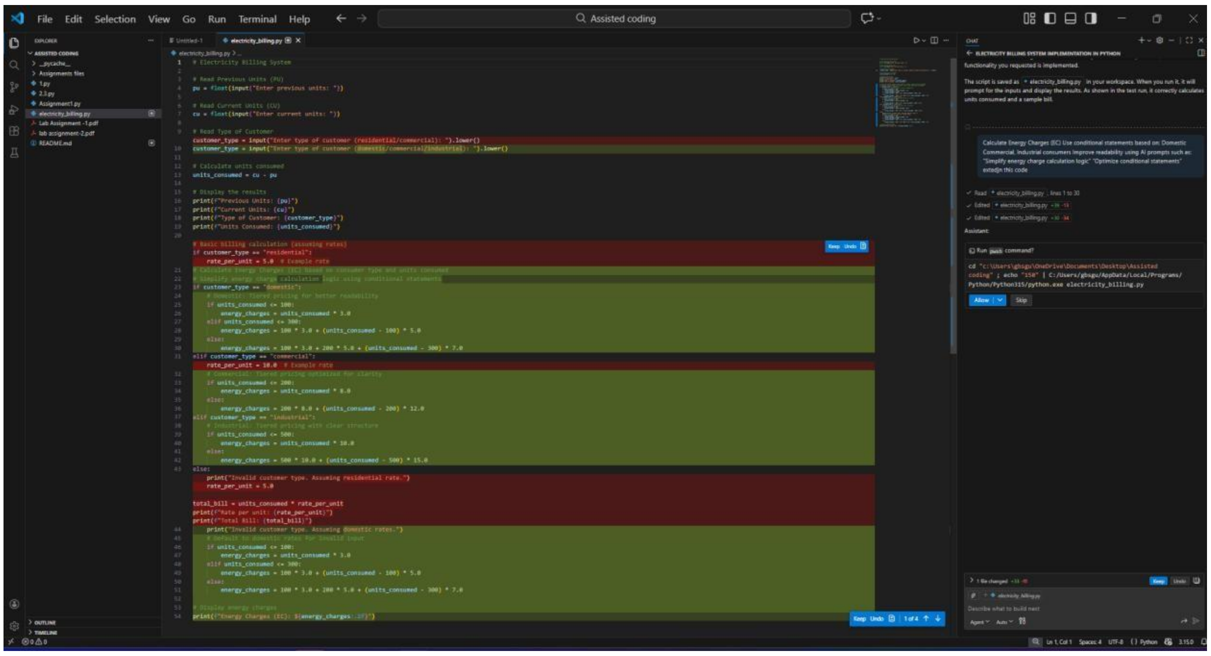
I also added a basic billing calculation with example rates for completeness, but the core functionality you requested is implemented.

The script is saved as electricity_billing.py in your workspace. When you run it, it will prompt for the inputs and display the results. As shown in the test run, it correctly calculates units consumed and a sample bill.

Task 2: Energy Charges Calculation Based on Units Consumed

Scenario

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



```python
# Electricity Billing System

# Read Previous Units (PU)
pu = float(input("Enter previous units: "))

# Read Current Units (CU)
cu = float(input("Enter current units: "))

# Read Type of Customer
customer_type = input("Enter type of customer (residential/commercial): ").lower()
customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()

# Calculate units consumed
units_consumed = cu - pu

# Display the results
print(f"Previous Units: {pu}")
print(f"Current Units: {cu}")
print(f"Type of Customer: {customer_type}")
print(f"Units Consumed: {units_consumed}")

# Basic billing calculation (assuming rates)
if customer_type == "residential":
    rate_per_unit = 5.0  # Example rate
# Calculate Energy Charges (EC) based on consumer type and units consumed
# Simplify energy charge calculation logic using conditional statements
if customer_type == "domestic":
    # Domestic: Tiered pricing for better readability
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif units_consumed <= 300:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
elif customer_type == "commercial":
    rate_per_unit = 10.0  # Example rate
    # Commercial: Tiered pricing optimized for clarity
    if units_consumed <= 200:
        energy_charges = units_consumed * 8.0
    else:
        energy_charges = 200 * 8.0 + (units_consumed - 200) * 12.0
elif customer_type == "industrial":
    # Industrial: Tiered pricing with clear structure
    if units_consumed <= 500:
        energy_charges = units_consumed * 10.0
    else:
        energy_charges = 500 * 10.0 + (units_consumed - 500) * 15.0
else:
    print("Invalid customer type. Assuming residential rate.")
    rate_per_unit = 5.0

total_bill = units_consumed * rate_per_unit
print(f"Rate per unit: {rate_per_unit}")
print(f"Total Bill: {total_bill}")
    print("Invalid customer type. Assuming domestic rates.")
    # Default to domestic rates for invalid input
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif units_consumed <= 300:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0

# Display energy charges
print(f"Energy Charges (EC): ${energy_charges:.2f}")
```

ELECTRICITY BILLING SYSTEM IMPLEMENTATION IN PYTHON

functionality you requested is implemented.

The script is saved as electricity_billing.py in your workspace. When you run it, it will prompt for the inputs and display the results. As shown in the test run, it correctly calculates units consumed and a sample bill.

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" extedjn this code

Read electricity_billing.py, lines 1 to 30

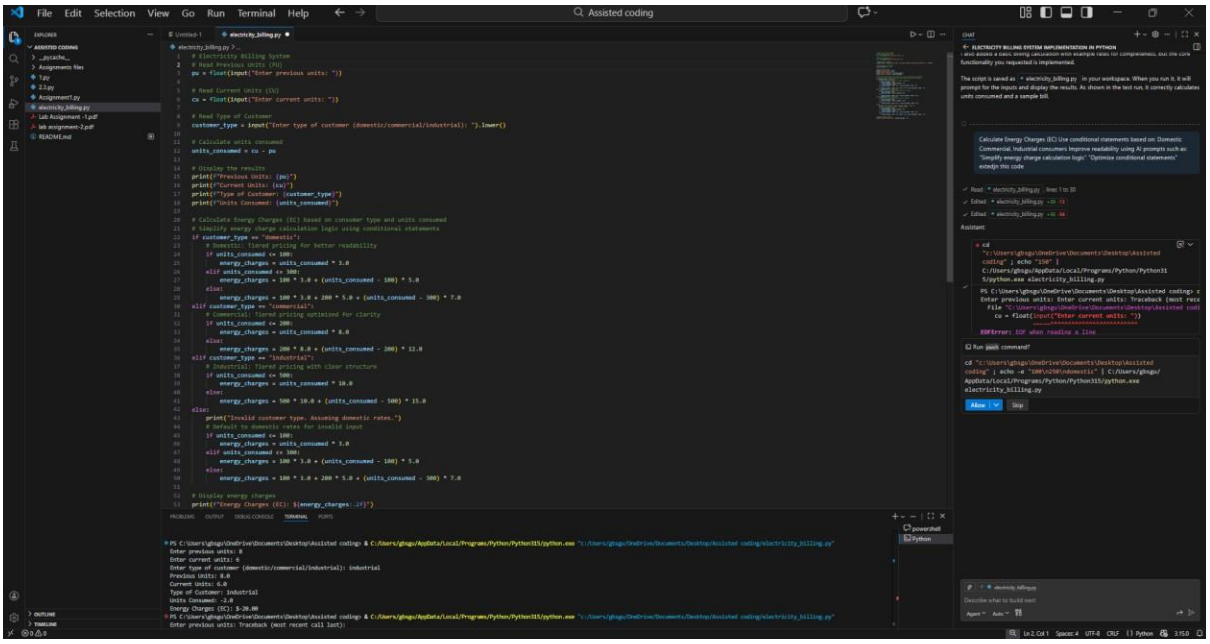
Edited electricity_billing.py +19 -13

Edited electricity_billing.py +30 -34

Assistant:

Run pwsh command?

```
cd "c:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding" ; echo "150" | C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe electricity_billing.py
```



```python
# Electricity Billing System
# Read Previous Units (PU)
pu = float(input("Enter previous units: "))

# Read Current Units (CU)
cu = float(input("Enter current units: "))

# Read Type of Customer
customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()

# Calculate units consumed
units_consumed = cu - pu

# Display the results
print(f"Previous Units: {pu}")
print(f"Current Units: {cu}")
print(f"Type of Customer: {customer_type}")
print(f"Units Consumed: {units_consumed}")

# Calculate Energy Charges (EC) based on consumer type and units consumed
# Simplify energy charge calculation logic using conditional statements
if customer_type == "domestic":
    # Domestic: Tiered pricing for better readability
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif units_consumed <= 300:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
elif customer_type == "commercial":
    # Commercial: Tiered pricing optimized for clarity
    if units_consumed <= 200:
        energy_charges = units_consumed * 8.0
    else:
        energy_charges = 200 * 8.0 + (units_consumed - 200) * 12.0
elif customer_type == "industrial":
    # Industrial: Tiered pricing with clear structure
    if units_consumed <= 500:
        energy_charges = units_consumed * 10.0
    else:
        energy_charges = 500 * 10.0 + (units_consumed - 500) * 15.0
else:
    print("Invalid customer type. Assuming domestic rates.")
    # Default to domestic rates for invalid input
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif units_consumed <= 300:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0

# Display energy charges
print(f"Energy Charges (EC): ${energy_charges:.2f}")
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
Enter previous units: 8
Enter current units: 6
Enter type of customer (domestic/commercial/industrial): industrial
Previous Units: 8.0
Current Units: 6.0
Type of Customer: industrial
Units Consumed: -2.0
Energy Charges (EC): $-20.00
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
Enter previous units: Traceback (most recent call last):
```

ELECTRICITY BILLING SYSTEM IMPLEMENTATION IN PYTHON

functionality you requested is implemented.

The script is saved as electricity_billing.py in your workspace. When you run it, it will prompt for the inputs and display the results. As shown in the test run, it correctly calculates units consumed and a sample bill.

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" extedjn this code

Read electricity_billing.py, lines 1 to 20

Edited electricity_billing.py

Edited electricity_billing.py

Assistant:

```
cd "c:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding" ; echo "150" | C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe electricity_billing.py
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> c
Enter previous units: Enter current units: Traceback (most rece
  File "C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted codi
    cu = float(input("Enter current units: "))
EOFError: EOF when reading a line
```

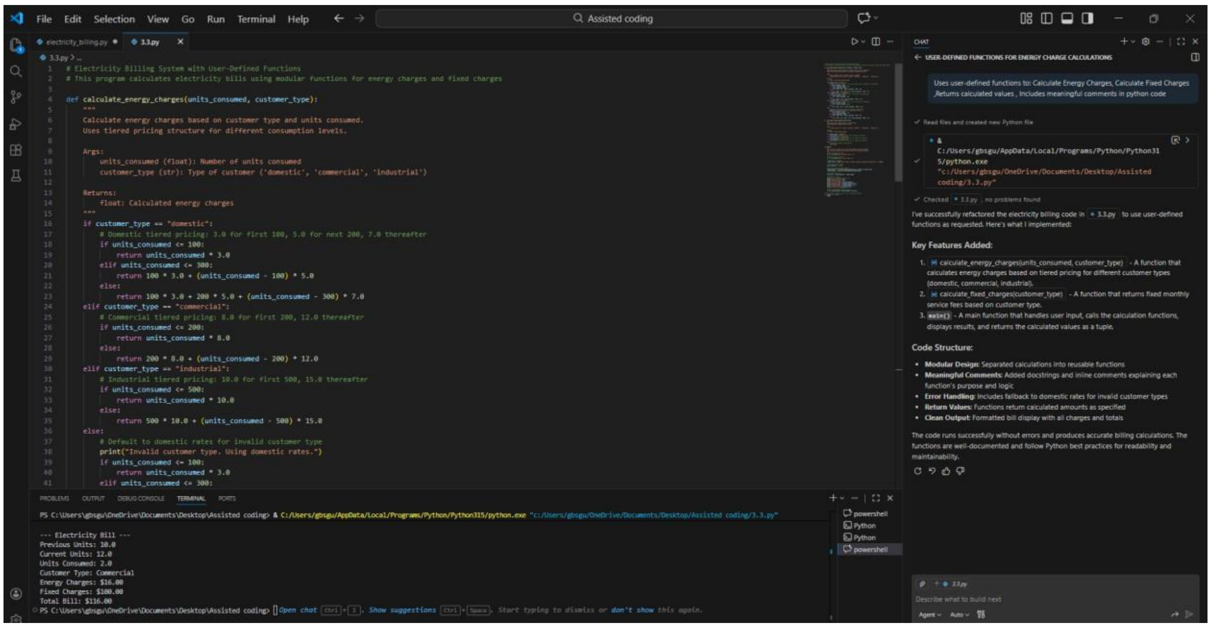
Run pwsh command?

```
cd "c:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding" ; echo -e "100\n250\ndomestic" | C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe electricity_billing.py
```

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

Scenario

Billing logic must be reusable for multiple consumers.



```python
# Electricity Billing System with User-Defined Functions
# This program calculates electricity bills using modular functions for energy charges and fixed charges

def calculate_energy_charges(units_consumed, customer_type):
    """
    Calculate energy charges based on customer type and units consumed.
    Uses tiered pricing structure for different consumption levels.

    Args:
        units_consumed (float): Number of units consumed
        customer_type (str): Type of customer ('domestic', 'commercial', 'industrial')

    Returns:
        float: Calculated energy charges
    """
    if customer_type == "domestic":
        # Domestic tiered pricing: 3.0 for first 100, 5.0 for next 200, 7.0 thereafter
        if units_consumed <= 100:
            return units_consumed * 3.0
        elif units_consumed <= 300:
            return 100 * 3.0 + (units_consumed - 100) * 5.0
        else:
            return 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
    elif customer_type == "commercial":
        # Commercial tiered pricing: 8.0 for first 200, 12.0 thereafter
        if units_consumed <= 200:
            return units_consumed * 8.0
        else:
            return 200 * 8.0 + (units_consumed - 200) * 12.0
    elif customer_type == "industrial":
        # Industrial tiered pricing: 10.0 for first 500, 15.0 thereafter
        if units_consumed <= 500:
            return units_consumed * 10.0
        else:
            return 500 * 10.0 + (units_consumed - 500) * 15.0
    else:
        # Default to domestic rates for invalid customer type
        print("Invalid customer type. Using domestic rates.")
        if units_consumed <= 100:
            return units_consumed * 3.0
        elif units_consumed <= 300:
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/3.3.py"

--- Electricity Bill ---
Previous Units: 10.0
Current Units: 12.0
Units Consumed: 2.0
Customer Type: Commercial
Energy Charges: $16.00
Fixed Charges: $100.00
Total Bill: $116.00
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> Open chat Ctrl + I. Show suggestions Ctrl + Space. Start typing to dismiss or don't show this again.
```

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

Uses user-defined functions to: Calculate Energy Charges, Calculate Fixed Charges ,Returns calculated values , Includes meaningful comments in python code

Read files and created new Python file

```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/3.3.py"
```

Checked 3.3.py, no problems found

I've successfully refactored the electricity billing code in 3.3.py to use user-defined functions as requested. Here's what I implemented:

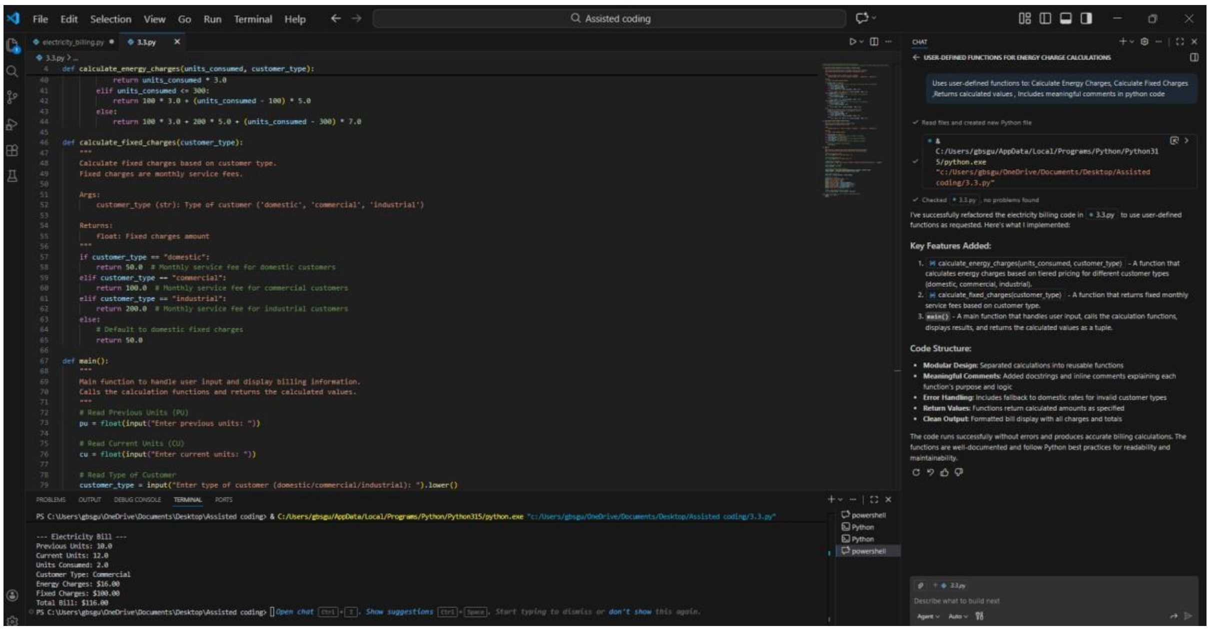
Key Features Added:

1. calculate_energy_charges(units_consumed, customer_type) - A function that calculates energy charges based on tiered pricing for different customer types (domestic, commercial, industrial).
2. calculate_fixed_charges(customer_type) - A function that returns fixed monthly service fees based on customer type.
3. `main()` - A main function that handles user input, calls the calculation functions, displays results, and returns the calculated values as a tuple.

Code Structure:

- **Modular Design:** Separated calculations into reusable functions
- **Meaningful Comments:** Added docstrings and inline comments explaining each function's purpose and logic
- **Error Handling:** Includes fallback to domestic rates for invalid customer types
- **Return Values:** Functions return calculated amounts as specified
- **Clean Output:** Formatted bill display with all charges and totals

The code runs successfully without errors and produces accurate billing calculations. The functions are well-documented and follow Python best practices for readability and maintainability.



```python
def calculate_energy_charges(units_consumed, customer_type):
            return units_consumed * 3.0
        elif units_consumed <= 300:
            return 100 * 3.0 + (units_consumed - 100) * 5.0
        else:
            return 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0

def calculate_fixed_charges(customer_type):
    """
    Calculate fixed charges based on customer type.
    Fixed charges are monthly service fees.

    Args:
        customer_type (str): Type of customer ('domestic', 'commercial', 'industrial')

    Returns:
        float: Fixed charges amount
    """
    if customer_type == "domestic":
        return 50.0  # Monthly service fee for domestic customers
    elif customer_type == "commercial":
        return 100.0  # Monthly service fee for commercial customers
    elif customer_type == "industrial":
        return 200.0  # Monthly service fee for industrial customers
    else:
        # Default to domestic fixed charges
        return 50.0

def main():
    """
    Main function to handle user input and display billing information.
    Calls the calculation functions and returns the calculated values.
    """
    # Read Previous Units (PU)
    pu = float(input("Enter previous units: "))

    # Read Current Units (CU)
    cu = float(input("Enter current units: "))

    # Read Type of Customer
    customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/3.3.py"

--- Electricity Bill ---
Previous Units: 10.0
Current Units: 12.0
Units Consumed: 2.0
Customer Type: Commercial
Energy Charges: $16.00
Fixed Charges: $100.00
Total Bill: $116.00
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> Open chat Ctrl+I, Show suggestions Ctrl+Space, Start typing to dismiss or don't show this again.
```

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

Uses user-defined functions to: Calculate Energy Charges, Calculate Fixed Charges ,Returns calculated values , Includes meaningful comments in python code

✓ Read files and created new Python file

```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/3.3.py"
```

✓ Checked 3.3.py, no problems found

I've successfully refactored the electricity billing code in 3.3.py to use user-defined functions as requested. Here's what I implemented:

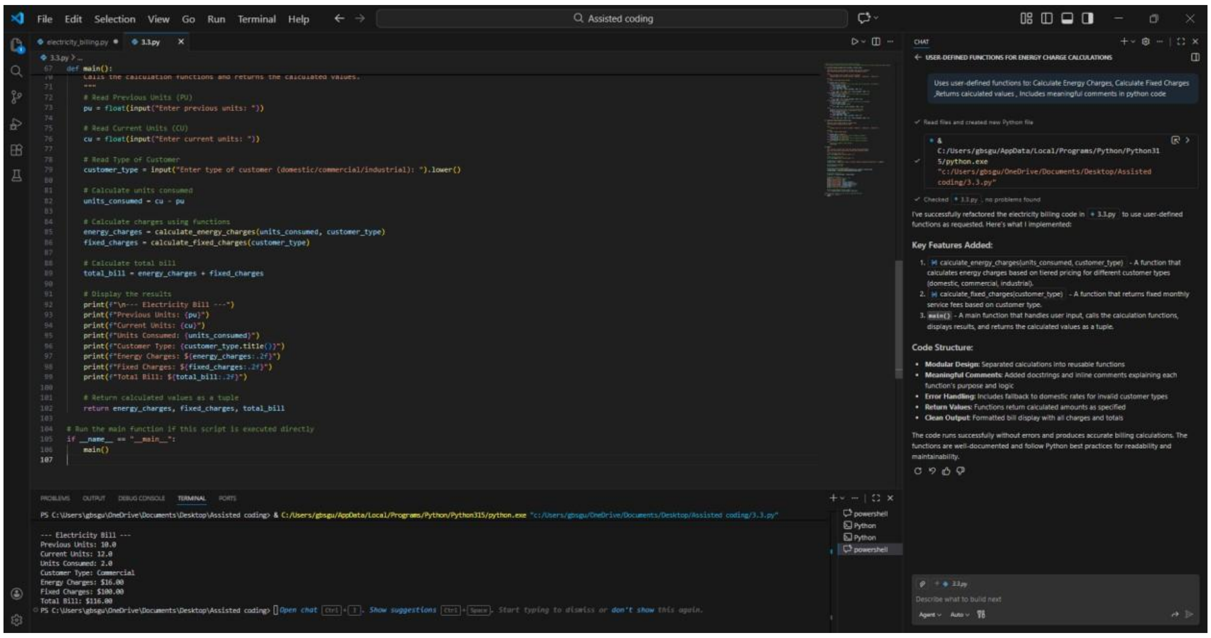
Key Features Added:

1. `calculate_energy_charges(units_consumed, customer_type)` - A function that calculates energy charges based on tiered pricing for different customer types (domestic, commercial, industrial).
2. `calculate_fixed_charges(customer_type)` - A function that returns fixed monthly service fees based on customer type.
3. `main()` - A main function that handles user input, calls the calculation functions, displays results, and returns the calculated values as a tuple.

Code Structure:

- **Modular Design:** Separated calculations into reusable functions
- **Meaningful Comments:** Added docstrings and inline comments explaining each function's purpose and logic
- **Error Handling:** Includes fallback to domestic rates for invalid customer types
- **Return Values:** Functions return calculated amounts as specified
- **Clean Output:** Formatted bill display with all charges and totals

The code runs successfully without errors and produces accurate billing calculations. The functions are well-documented and follow Python best practices for readability and maintainability.



```python
def main():
    Calls the calculation functions and returns the calculated values.
    """
    # Read Previous Units (PU)
    pu = float(input("Enter previous units: "))

    # Read Current Units (CU)
    cu = float(input("Enter current units: "))

    # Read Type of Customer
    customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()

    # Calculate units consumed
    units_consumed = cu - pu

    # Calculate charges using functions
    energy_charges = calculate_energy_charges(units_consumed, customer_type)
    fixed_charges = calculate_fixed_charges(customer_type)

    # Calculate total bill
    total_bill = energy_charges + fixed_charges

    # Display the results
    print(f"\n--- Electricity Bill ---")
    print(f"Previous Units: {pu}")
    print(f"Current Units: {cu}")
    print(f"Units Consumed: {units_consumed}")
    print(f"Customer Type: {customer_type.title()}")
    print(f"Energy Charges: ${energy_charges:.2f}")
    print(f"Fixed Charges: ${fixed_charges:.2f}")
    print(f"Total Bill: ${total_bill:.2f}")

    # Return calculated values as a tuple
    return energy_charges, fixed_charges, total_bill

# Run the main function if this script is executed directly
if __name__ == "__main__":
    main()
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/3.3.py"

--- Electricity Bill ---
Previous Units: 10.0
Current Units: 12.0
Units Consumed: 2.0
Customer Type: Commercial
Energy Charges: $16.00
Fixed Charges: $100.00
Total Bill: $116.00
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> Open chat Ctrl+I, Show suggestions Ctrl+Space, Start typing to dismiss or don't show this again.
```

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

Uses user-defined functions to: Calculate Energy Charges, Calculate Fixed Charges ,Returns calculated values , Includes meaningful comments in python code

✓ Read files and created new Python file

✓ Checked 3.3.py, no problems found

I've successfully refactored the electricity billing code in 3.3.py to use user-defined functions as requested. Here's what I implemented:

Key Features Added:

1. `calculate_energy_charges(units_consumed, customer_type)` - A function that calculates energy charges based on tiered pricing for different customer types (domestic, commercial, industrial).
2. `calculate_fixed_charges(customer_type)` - A function that returns fixed monthly service fees based on customer type.
3. `main()` - A main function that handles user input, calls the calculation functions, displays results, and returns the calculated values as a tuple.

Code Structure:

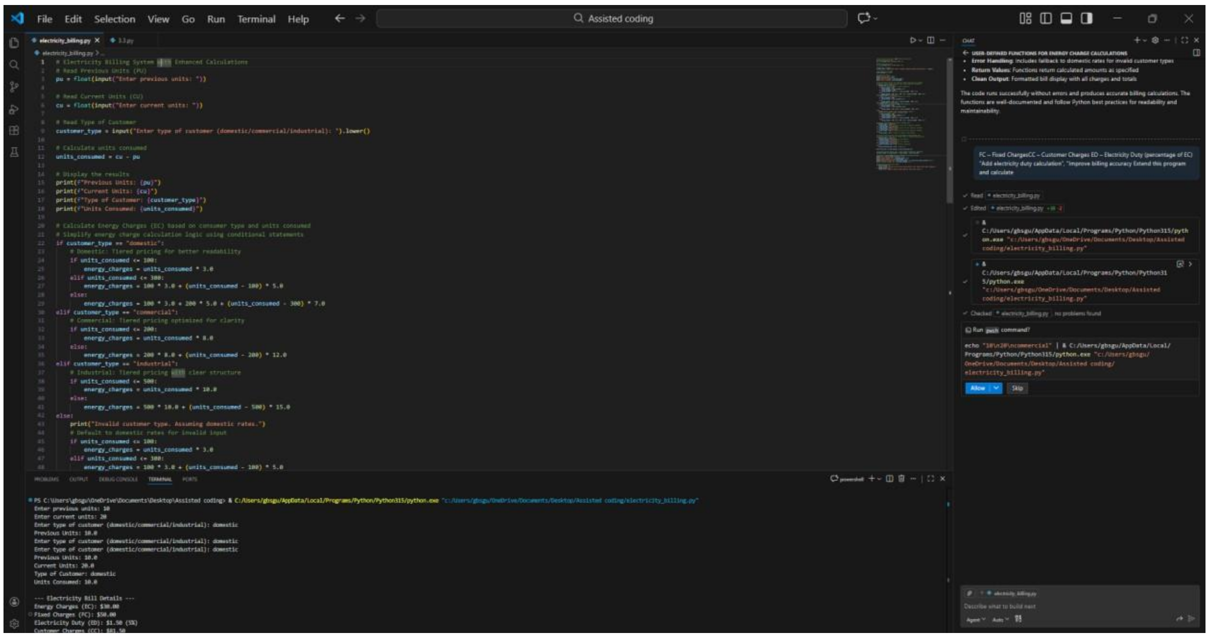
- **Modular Design:** Separated calculations into reusable functions
- **Meaningful Comments:** Added docstrings and inline comments explaining each function's purpose and logic
- **Error Handling:** Includes fallback to domestic rates for invalid customer types
- **Return Values:** Functions return calculated amounts as specified
- **Clean Output:** Formatted bill display with all charges and totals

The code runs successfully without errors and produces accurate billing calculations. The functions are well-documented and follow Python best practices for readability and maintainability.

Task 4: Calculation of Additional Charges

Scenario

Electricity bills include multiple additional charges.


File Edit Selection View Go Run Terminal Help
Assisted coding
electricity_billing.py
3.3.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
CHAT

```python
# Electricity Billing System with Enhanced Calculations
# Read Previous Units (PU)
pu = float(input("Enter previous units: "))

# Read Current Units (CU)
cu = float(input("Enter current units: "))

# Read Type of Customer
customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower()

# Calculate units consumed
units_consumed = cu - pu

# Display the results
print(f"Previous Units: {pu}")
print(f"Current Units: {cu}")
print(f"Type of Customer: {customer_type}")
print(f"Units Consumed: {units_consumed}")

# Calculate Energy Charges (EC) based on consumer type and units consumed
# Simplify energy charge calculation logic using conditional statements
if customer_type == "domestic":
    # Domestic: Tiered pricing for better readability
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif units_consumed <= 300:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
elif customer_type == "commercial":
    # Commercial: Tiered pricing optimized for clarity
    if units_consumed <= 200:
        energy_charges = units_consumed * 8.0
    else:
        energy_charges = 200 * 8.0 + (units_consumed - 200) * 12.0
elif customer_type == "industrial":
    # Industrial: Tiered pricing with clear structure
    if units_consumed <= 500:
        energy_charges = units_consumed * 10.0
    else:
        energy_charges = 500 * 10.0 + (units_consumed - 500) * 15.0
else:
    print("Invalid customer type. Assuming domestic rates.")
    # Default to domestic rates for invalid input
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif units_consumed <= 300:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
Enter previous units: 10
Enter current units: 20
Enter type of customer (domestic/commercial/industrial): domestic
Previous Units: 10.0
Enter type of customer (domestic/commercial/industrial): domestic
Enter type of customer (domestic/commercial/industrial): domestic
Previous Units: 10.0
Current Units: 20.0
Type of Customer: domestic
Units Consumed: 10.0

--- Electricity Bill Details ---
Energy Charges (EC): $30.00
Fixed Charges (FC): $50.00
Electricity Duty (ED): $1.50 (5%)
Customer Charges (CC): $81.50
```

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

- **Error Handling**: Includes fallback to domestic rates for invalid customer types
- **Return Values**: Functions return calculated amounts as specified
- **Clean Output**: Formatted bill display with all charges and totals

The code runs successfully without errors and produces accurate billing calculations. The functions are well-documented and follow Python best practices for readability and maintainability.

FC – Fixed ChargesCC – Customer Charges ED – Electricity Duty (percentage of EC) "Add electricity duty calculation", "Improve billing accuracy Extend this program and calculate

✓ Read electricity_billing.py

✓ Edited electricity_billing.py +18 -2

```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

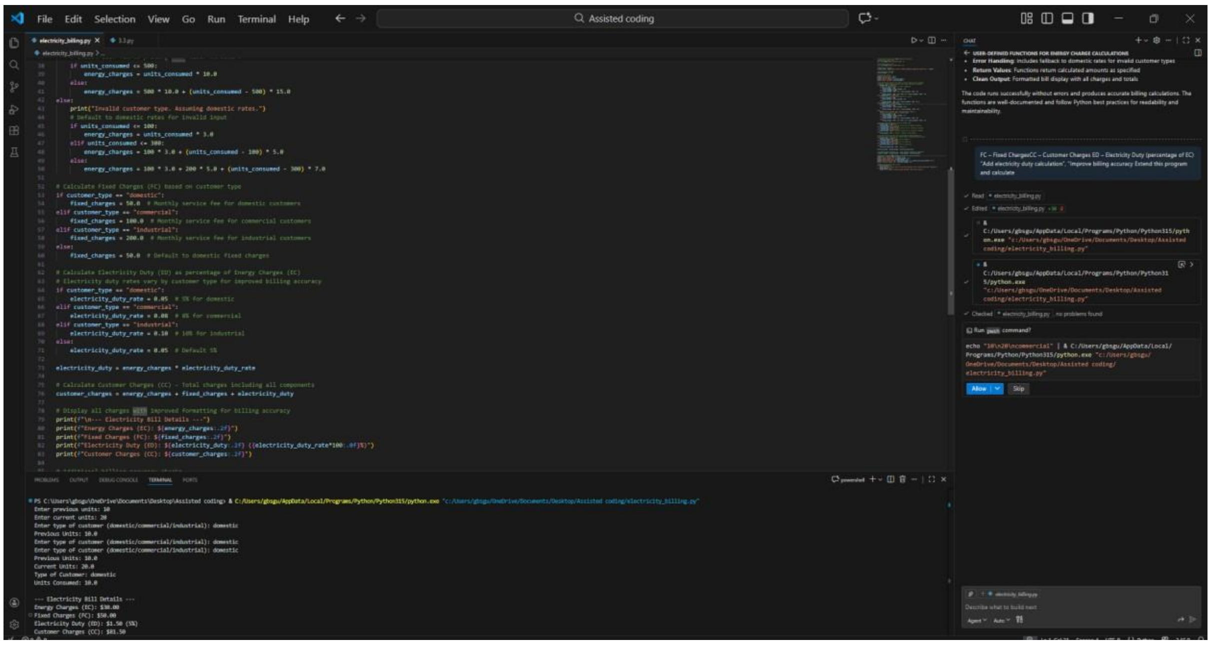
```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

✓ Checked electricity_billing.py, no problems found

Run pwsh command?

```
echo "10\n20\ncommercial" | & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

Allow
Skip
Describe what to build next
Agent
Auto


File Edit Selection View Go Run Terminal Help
Assisted coding
electricity_billing.py
3.3.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
CHAT

```python
    if units_consumed <= 500:
        energy_charges = units_consumed * 10.0
    else:
        energy_charges = 500 * 10.0 + (units_consumed - 500) * 15.0
else:
    print("Invalid customer type. Assuming domestic rates.")
    # Default to domestic rates for invalid input
    if units_consumed <= 100:
        energy_charges = units_consumed * 3.0
    elif units_consumed <= 300:
        energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
    else:
        energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0

# Calculate Fixed Charges (FC) based on customer type
if customer_type == "domestic":
    fixed_charges = 50.0  # Monthly service fee for domestic customers
elif customer_type == "commercial":
    fixed_charges = 100.0  # Monthly service fee for commercial customers
elif customer_type == "industrial":
    fixed_charges = 200.0  # Monthly service fee for industrial customers
else:
    fixed_charges = 50.0  # Default to domestic fixed charges

# Calculate Electricity Duty (ED) as percentage of Energy Charges (EC)
# Electricity duty rates vary by customer type for improved billing accuracy
if customer_type == "domestic":
    electricity_duty_rate = 0.05  # 5% for domestic
elif customer_type == "commercial":
    electricity_duty_rate = 0.08  # 8% for commercial
elif customer_type == "industrial":
    electricity_duty_rate = 0.10  # 10% for industrial
else:
    electricity_duty_rate = 0.05  # Default 5%

electricity_duty = energy_charges * electricity_duty_rate

# Calculate Customer Charges (CC) - Total charges including all components
customer_charges = energy_charges + fixed_charges + electricity_duty

# Display all charges with improved formatting for billing accuracy
print(f"\n--- Electricity Bill Details ---")
print(f"Energy Charges (EC): ${energy_charges:.2f}")
print(f"Fixed Charges (FC): ${fixed_charges:.2f}")
print(f"Electricity Duty (ED): ${electricity_duty:.2f} ({electricity_duty_rate*100:.0f}%)")
print(f"Customer Charges (CC): ${customer_charges:.2f}")
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
Enter previous units: 10
Enter current units: 20
Enter type of customer (domestic/commercial/industrial): domestic
Previous Units: 10.0
Enter type of customer (domestic/commercial/industrial): domestic
Enter type of customer (domestic/commercial/industrial): domestic
Previous Units: 10.0
Current Units: 20.0
Type of Customer: domestic
Units Consumed: 10.0

--- Electricity Bill Details ---
Energy Charges (EC): $30.00
Fixed Charges (FC): $50.00
Electricity Duty (ED): $1.50 (5%)
Customer Charges (CC): $81.50
```

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

- **Error Handling**: Includes fallback to domestic rates for invalid customer types
- **Return Values**: Functions return calculated amounts as specified
- **Clean Output**: Formatted bill display with all charges and totals

The code runs successfully without errors and produces accurate billing calculations. The functions are well-documented and follow Python best practices for readability and maintainability.

FC – Fixed ChargesCC – Customer Charges ED – Electricity Duty (percentage of EC) "Add electricity duty calculation", "Improve billing accuracy Extend this program and calculate

✓ Read electricity_billing.py

✓ Edited electricity_billing.py +18 -2

✓ Checked electricity_billing.py, no problems found

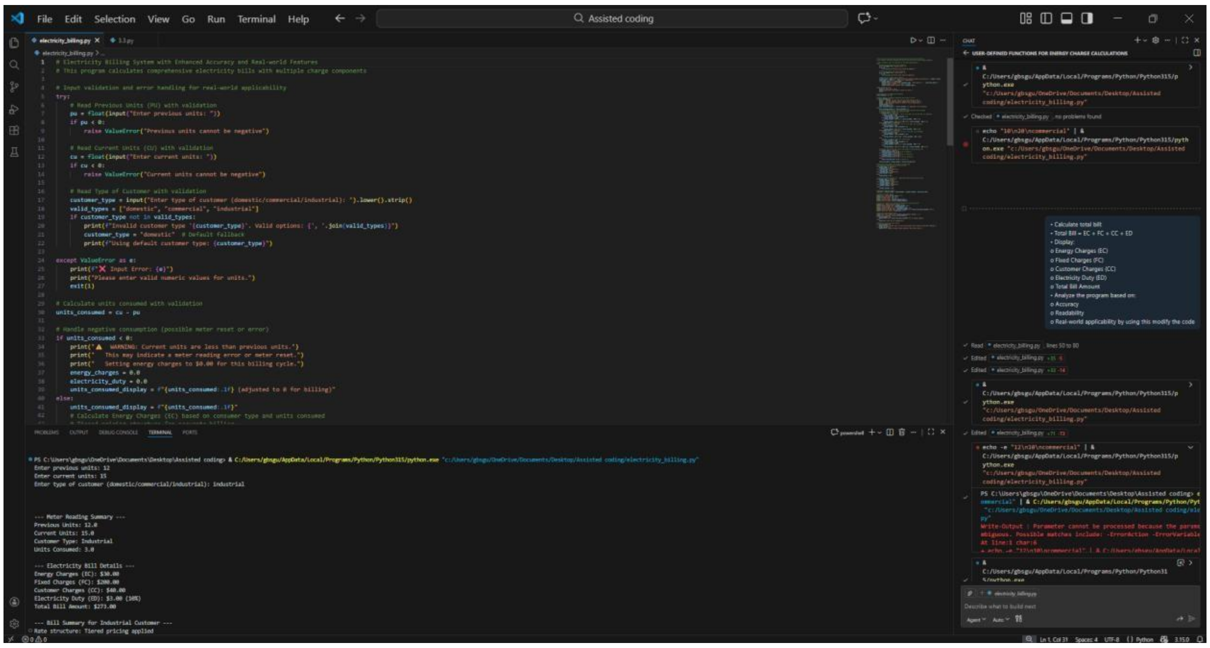
Run pwsh command?

echo "10\n20\ncommercial" | & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"

Task 5: Final Bill Generation and Output Analysis

Scenario

The final electricity bill must present all values clearly.



```python
# Electricity Billing System with Enhanced Accuracy and Real-world Features
# This program calculates comprehensive electricity bills with multiple charge components

# Input validation and error handling for real-world applicability
try:
    # Read Previous Units (PU) with validation
    pu = float(input("Enter previous units: "))
    if pu < 0:
        raise ValueError("Previous units cannot be negative")

    # Read Current Units (CU) with validation
    cu = float(input("Enter current units: "))
    if cu < 0:
        raise ValueError("Current units cannot be negative")

    # Read Type of Customer with validation
    customer_type = input("Enter type of customer (domestic/commercial/industrial): ").lower().strip()
    valid_types = ["domestic", "commercial", "industrial"]
    if customer_type not in valid_types:
        print(f"Invalid customer type '{customer_type}'. Valid options: {', '.join(valid_types)}")
        customer_type = "domestic"  # Default fallback
        print(f"Using default customer type: {customer_type}")

except ValueError as e:
    print(f"❌ Input Error: {e}")
    print("Please enter valid numeric values for units.")
    exit(1)

# Calculate units consumed with validation
units_consumed = cu - pu

# Handle negative consumption (possible meter reset or error)
if units_consumed < 0:
    print("⚠  WARNING: Current units are less than previous units.")
    print("   This may indicate a meter reading error or meter reset.")
    print("   Setting energy charges to $0.00 for this billing cycle.")
    energy_charges = 0.0
    electricity_duty = 0.0
    units_consumed_display = f"{units_consumed:.1f} (adjusted to 0 for billing)"
else:
    units_consumed_display = f"{units_consumed:.1f}"
    # Calculate Energy Charges (EC) based on consumer type and units consumed
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
Enter previous units: 12
Enter current units: 15
Enter type of customer (domestic/commercial/industrial): industrial



--- Meter Reading Summary ---
Previous Units: 12.0
Current Units: 15.0
Customer Type: Industrial
Units Consumed: 3.0

--- Electricity Bill Details ---
Energy Charges (EC): $30.00
Fixed Charges (FC): $200.00
Customer Charges (CC): $40.00
Electricity Duty (ED): $3.00 (10%)
Total Bill Amount: $273.00

--- Bill Summary for Industrial Customer ---
Rate structure: Tiered pricing applied
```

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

Checked electricity_billing.py, no problems found

```
echo "10\n20\ncommercial" | & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

- Calculate total bill
- Total Bill = EC + FC + CC + ED
- Display:
o Energy Charges (EC)
o Fixed Charges (FC)
o Customer Charges (CC)
o Electricity Duty (ED)
o Total Bill Amount
- Analyze the program based on:
o Accuracy
o Readability
o Real-world applicability by using this modify the code

Read electricity_billing.py, lines 50 to 80
Edited electricity_billing.py +35 -6
Edited electricity_billing.py +32 -14

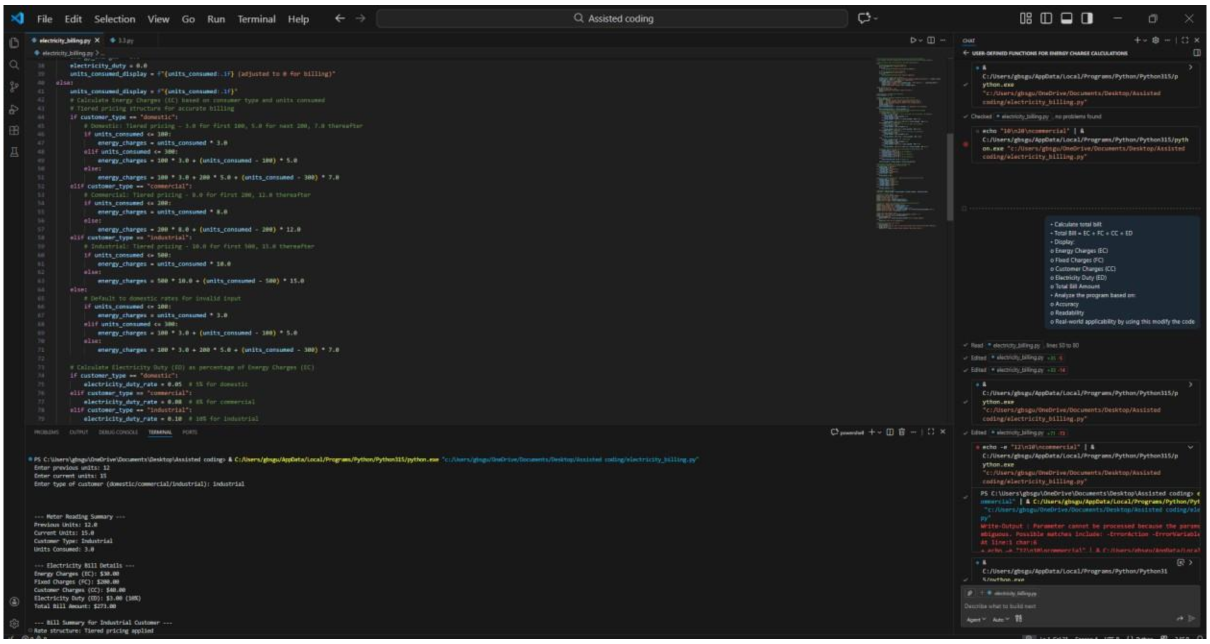
```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

Edited electricity_billing.py +71 -73

```
echo -e "12\n10\ncommercial" | & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> e
ommercial" | & C:/Users/gbsgu/AppData/Local/Programs/Python/Pyt
 "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/ele
py"
Write-Output : Parameter cannot be processed because the param
mbiguous. Possible matches include: -ErrorAction -ErrorVariable
At line:1 char:6
```

```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python31
5/python.exe
```



```python
    electricity_duty = 0.0
    units_consumed_display = f"{units_consumed:.1f} (adjusted to 0 for billing)"
else:
    units_consumed_display = f"{units_consumed:.1f}"
    # Calculate Energy Charges (EC) based on consumer type and units consumed
    # Tiered pricing structure for accurate billing
    if customer_type == "domestic":
        # Domestic: Tiered pricing - 3.0 for first 100, 5.0 for next 200, 7.0 thereafter
        if units_consumed <= 100:
            energy_charges = units_consumed * 3.0
        elif units_consumed <= 300:
            energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
        else:
            energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0
    elif customer_type == "commercial":
        # Commercial: Tiered pricing - 8.0 for first 200, 12.0 thereafter
        if units_consumed <= 200:
            energy_charges = units_consumed * 8.0
        else:
            energy_charges = 200 * 8.0 + (units_consumed - 200) * 12.0
    elif customer_type == "industrial":
        # Industrial: Tiered pricing - 10.0 for first 500, 15.0 thereafter
        if units_consumed <= 500:
            energy_charges = units_consumed * 10.0
        else:
            energy_charges = 500 * 10.0 + (units_consumed - 500) * 15.0
    else:
        # Default to domestic rates for invalid input
        if units_consumed <= 100:
            energy_charges = units_consumed * 3.0
        elif units_consumed <= 300:
            energy_charges = 100 * 3.0 + (units_consumed - 100) * 5.0
        else:
            energy_charges = 100 * 3.0 + 200 * 5.0 + (units_consumed - 300) * 7.0

    # Calculate Electricity Duty (ED) as percentage of Energy Charges (EC)
    if customer_type == "domestic":
        electricity_duty_rate = 0.05  # 5% for domestic
    elif customer_type == "commercial":
        electricity_duty_rate = 0.08  # 8% for commercial
    elif customer_type == "industrial":
        electricity_duty_rate = 0.10  # 10% for industrial
```

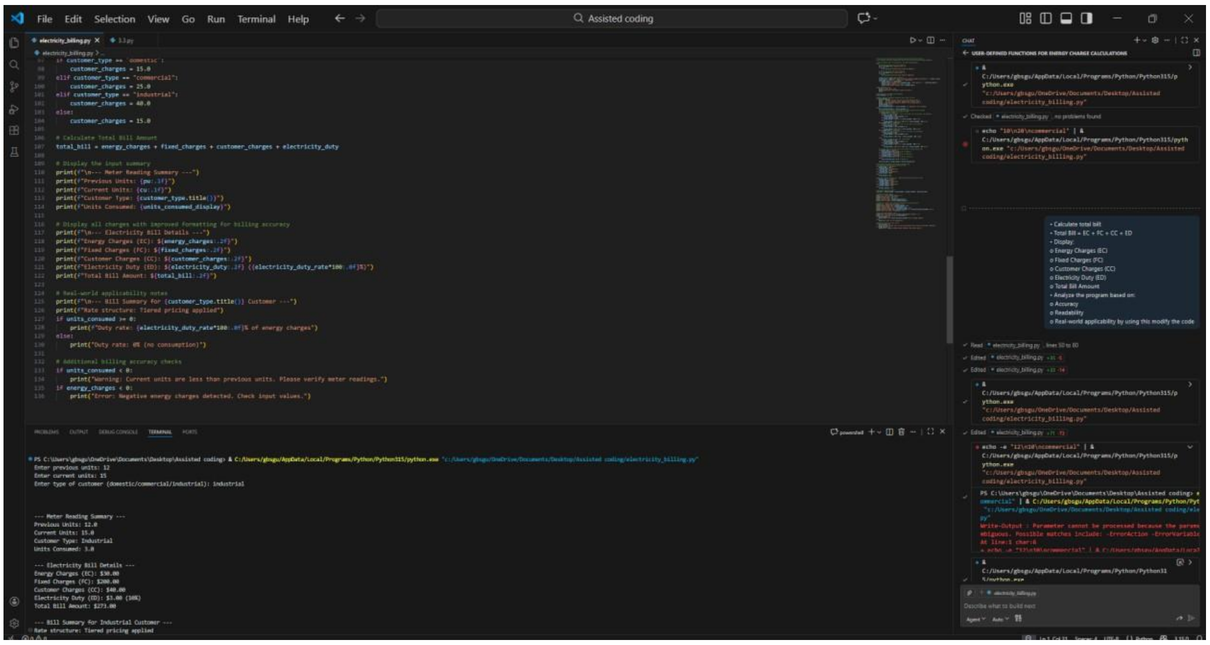
```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
Enter previous units: 12
Enter current units: 15
Enter type of customer (domestic/commercial/industrial): industrial



--- Meter Reading Summary ---
Previous Units: 12.0
Current Units: 15.0
Customer Type: Industrial
Units Consumed: 3.0

--- Electricity Bill Details ---
Energy Charges (EC): $30.00
Fixed Charges (FC): $200.00
Customer Charges (CC): $40.00
Electricity Duty (ED): $3.00 (10%)
Total Bill Amount: $273.00

--- Bill Summary for Industrial Customer ---
Rate structure: Tiered pricing applied
```



```python
    if customer_type == "domestic":
        customer_charges = 15.0
    elif customer_type == "commercial":
        customer_charges = 25.0
    elif customer_type == "industrial":
        customer_charges = 40.0
    else:
        customer_charges = 15.0

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

    # Display the input summary
    print(f"\n--- Meter Reading Summary ---")
    print(f"Previous Units: {pu:.1f}")
    print(f"Current Units: {cu:.1f}")
    print(f"Customer Type: {customer_type.title()}")
    print(f"Units Consumed: {units_consumed_display}")

    # Display all charges with improved formatting for billing accuracy
    print(f"\n--- Electricity Bill Details ---")
    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} ({electricity_duty_rate*100:.0f}%)")
    print(f"Total Bill Amount: ${total_bill:.2f}")

    # Real-world applicability notes
    print(f"\n--- Bill Summary for {customer_type.title()} Customer ---")
    print(f"Rate structure: Tiered pricing applied")
    if units_consumed >= 0:
        print(f"Duty rate: {electricity_duty_rate*100:.0f}% of energy charges")
    else:
        print("Duty rate: 0% (no consumption)")

    # Additional billing accuracy checks
    if units_consumed < 0:
        print("Warning: Current units are less than previous units. Please verify meter readings.")
    if energy_charges < 0:
        print("Error: Negative energy charges detected. Check input values.")
```

```
PS C:\Users\gbsgu\OneDrive\Documents\Desktop\Assisted coding> & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
Enter previous units: 12
Enter current units: 15
Enter type of customer (domestic/commercial/industrial): industrial


--- Meter Reading Summary ---
Previous Units: 12.0
Current Units: 15.0
Customer Type: Industrial
Units Consumed: 3.0

--- Electricity Bill Details ---
Energy Charges (EC): $30.00
Fixed Charges (FC): $200.00
Customer Charges (CC): $40.00
Electricity Duty (ED): $3.00 (10%)
Total Bill Amount: $273.00

--- Bill Summary for Industrial Customer ---
Rate structure: Tiered pricing applied
```

USER-DEFINED FUNCTIONS FOR ENERGY CHARGE CALCULATIONS

```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

Checked electricity_billing.py, no problems found

```
echo "10\n20\ncommercial" | & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

- 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 by using this modify the code

Read electricity_billing.py, lines 50 to 80
Edited electricity_billing.py
Edited electricity_billing.py

```
& C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
```

Edited electricity_billing.py

```
echo -e "12\n18\ncommercial" | & C:/Users/gbsgu/AppData/Local/Programs/Python/Python315/python.exe "c:/Users/gbsgu/OneDrive/Documents/Desktop/Assisted coding/electricity_billing.py"
Write-Output : Parameter cannot be processed because the param
ambiguous. Possible matches include: -ErrorAction -ErrorVariable
At line:1 char:6
```

This program accurately calculates the electricity bill by using basic arithmetic formulas. The code is easy to read because of meaningful variable names and clear print statements. It is applicable in real-world situations as it follows the standard electricity billing structure used by power departments. The formatted output helps users understand each charge clearly.