AI Assisted Coding

ASSIGNMENT 3.3

Name: Mohammed Farhan Ahmed
HT No: 2303A52040
Batch: 31

Lab 3: Application for TGNPDCL - Electricity Bill Generation Using Python & AI Tools

Lab Objectives
• To design a real-world electricity billing application using Python
• To use AI-assisted coding tools for logic generation and optimization
• To understand conditional logic and arithmetic operations
• To generate structured billing output similar to utility bills

Question 1:

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)

Expected Output:
• Correct input reading
• Units consumed calculation
• Screenshot showing AI-generated code
• Sample input and output

Prompt:

Write a Python program for an electricity billing system that reads Previous Units, Current Units, and Type of Customer from the user, calculates units consumed as (Current Units - Previous Units), implements all logic directly in the main program without using functions, and prints the inputs and units consumed.

Code:

```python
# Electricity Billing System

print("=== Electricity Billing System ===\n")

# Read inputs from user
previous_units = float(input("Enter Previous Units: "))
current_units = float(input("Enter Current Units: "))
customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")

# Calculate units consumed
units_consumed = current_units - previous_units

# Display inputs and units consumed
print("\n=== Bill Details ===")
print(f"Previous Units: {previous_units}")
print(f"Current Units: {current_units}")
print(f"Type of Customer: {customer_type}")
print(f"Units Consumed: {units_consumed}")
```

```python
# Calculate bill based on customer type
if units_consumed < 0:
    print("Error: Current Units cannot be less than Previous Units")
else:
    if customer_type.lower() == "domestic":
        rate = 5.0
    elif customer_type.lower() == "commercial":
        rate = 8.0
    elif customer_type.lower() == "industrial":
        rate = 10.0
    else:
        rate = 0
        print("Invalid customer type")

    if rate > 0:
        bill_amount = units_consumed * rate
        print(f"Rate per Unit: ₹{rate}")
        print(f"Total Bill Amount: ₹{bill_amount}")
```

Output(O/P)

```
=== Electricity Billing System ===

Enter Previous Units: 100
Enter Current Units: 250
Enter Type of Customer (Domestic/Commercial/Industrial): domestic

=== Bill Details ===
Previous Units: 100.0
Current Units: 250.0
Type of Customer: domestic
Units Consumed: 150.0
Rate per Unit: ₹5.0
Total Bill Amount: ₹750.0
```

```
 === Electricity Billing System ===

 Enter Previous Units: 500
 Enter Current Units: 600
 Enter Type of Customer (Domestic/Commercial/Industrial): commercial

 === Bill Details ===
 Previous Units: 500.0
 Current Units: 600.0
 Type of Customer: commercial
 Units Consumed: 100.0
 Rate per Unit: ₹8.0
 Total Bill Amount: ₹800.0
```

```
=== Electricity Billing System ===

Enter Previous Units: 10000
Enter Current Units: 25000
Enter Type of Customer (Domestic/Commercial/Industrial): industrial

=== Bill Details ===
Previous Units: 10000.0
Current Units: 25000.0
Type of Customer: industrial
Units Consumed: 15000.0
Rate per Unit: ₹10.0
Total Bill Amount: ₹150000.0
```

Explanation:

This task uses an AI tool to generate a Python program that reads consumer details such as Previous Units, Current Units, and Customer Type. The program calculates the units consumed by subtracting Previous Units from Current Units. All logic is written directly in the main program without using functions, ensuring simplicity and clarity. The output correctly displays the entered values and the calculated units consumed, along with sample input and output for verification.

Question 2:

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”

Expected Output

• Correct EC calculation

• Clear conditional logic

• Original and improved versions (optional)

• Sample execution results

Prompt:

Review the existing Python electricity billing program and extend it to calculate Energy Charges based on units consumed and customer type. Use conditional statements to handle Domestic, Commercial, and Industrial consumers with different charge rates. Improve code readability by simplifying the energy charge calculation logic and optimizing conditional statements. Print units consumed, customer type, and calculated Energy Charges.

Code:

```python
# Electricity Billing System

print("=== Electricity Billing System ===\n")

# Read inputs from user
previous_units = float(input("Enter Previous Units: "))
current_units = float(input("Enter Current Units: "))
customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")

# Calculate units consumed
units_consumed = current_units - previous_units

# Display inputs and units consumed
print("\n=== Bill Details ===")
print(f"Previous Units: {previous_units}")
print(f"Current Units: {current_units}")
print(f"Type of Customer: {customer_type}")
print(f"Units Consumed: {units_consumed}")

# Calculate bill based on customer type
if units_consumed < 0:
    print("Error: Current Units cannot be less than Previous Units")
else:
    # Define rates for each customer type
    rates = {
        "domestic": 5.0,
        "commercial": 8.0,
        "industrial": 10.0
    }

    customer_type_lower = customer_type.lower()

    if customer_type_lower not in rates:
        print("Invalid customer type. Please enter Domestic, Commercial, or Industrial.")
    else:
        rate = rates[customer_type_lower]
        energy_charges = units_consumed * rate

        print(f"\n=== Energy Charges ===")
        print(f"Rate per Unit: ₹{rate}")
        print(f"Energy Charges: ₹{energy_charges:.2f}")
        print(f"Total Bill Amount: ₹{energy_charges:.2f}")
```

Output:

```
=== Electricity Billing System ===

Enter Previous Units: 1000
Enter Current Units: 2500
Enter Type of Customer (Domestic/Commercial/Industrial): domestic

=== Bill Details ===
Previous Units: 1000.0
Current Units: 2500.0
Type of Customer: domestic
Units Consumed: 1500.0

=== Energy Charges ===
Rate per Unit: ₹5.0
Energy Charges: ₹7500.00
Total Bill Amount: ₹7500.00
```

Explanation:

In this task, the program generated in Task 1 is extended to calculate Energy Charges based on units consumed and the type of customer. Conditional statements are used to apply different charge rates for Domestic, Commercial, and Industrial consumers. AI assistance helps simplify the calculation logic and optimize conditions, improving readability. The program correctly computes and displays Energy Charges with sample execution results for verification.

Question 3:

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

Expected Output:

• Function-based Python program

• Correct EC and FC values

• Screenshots of AI-assisted function generation

• Test cases with outputs

Prompt:

Generate a Python electricity billing program using modular design. Create user-defined functions to calculate Energy Charges and Fixed Charges based on units consumed and customer type. Each function should return the calculated value and include meaningful comments. In the main program, read inputs, call the functions, and display the results.

Code:

```python
# Electricity Billing Program using Modular Design

def calculate_energy_charges(units, customer_type):
    """
    Calculate energy charges based on units consumed and customer type.

    Args:
        units: Number of units consumed
        customer_type: Type of customer ('residential', 'commercial', 'industrial')

    Returns:
        Energy charges as a float
    """
    # Define rate per unit for each customer type
    rates = {
        'residential': 5.0,
        'commercial': 7.5,
        'industrial': 4.0
    }

    # Get rate for customer type (default to residential if not found)
    rate = rates.get(customer_type.lower(), 5.0)

    # Calculate and return energy charges
    energy_charges = units * rate
    return energy_charges

```

```python

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

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

    Returns:
        Fixed charges as a float
    """
    # Define fixed charges for each customer type
    fixed_rates = {
        'residential': 100.0,
        'commercial': 300.0,
        'industrial': 500.0
    }

    # Get fixed charges for customer type (default to residential if not found)
    fixed_charges = fixed_rates.get(customer_type.lower(), 100.0)
    return fixed_charges

```

```python
def main():
    """
    Main program to read inputs, calculate charges, and display results.
    """
    print("===== Electricity Billing System =====\n")

    # Read customer information
    customer_name = input("Enter customer name: ")
    customer_type = input("Enter customer type (residential/commercial/industrial): ")
    units = float(input("Enter units consumed: "))

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

    # Calculate total bill
    total_bill = energy_charges + fixed_charges

    # Display results
    print("\n===== Bill Summary =====")
    print(f"Customer Name: {customer_name}")
    print(f"Customer Type: {customer_type}")
    print(f"Units Consumed: {units}")
    print(f"Energy Charges: ${energy_charges:.2f}")
    print(f"Fixed Charges: ${fixed_charges:.2f}")
    print(f"Total Bill: ${total_bill:.2f}")

```

```python
# Entry point of the program
if __name__ == "__main__":
    main()
```

Output:

```
===== Electricity Billing System =====

Enter customer name: Infosys
Enter customer type (residential/commercial/industrial): industrial
Enter units consumed: 500

===== Bill Summary =====
Customer Name: Infosys
Customer Type: industrial
Units Consumed: 500.0
Energy Charges: $2000.00
Fixed Charges: $500.00
Total Bill: $2500.00
```

Explanation:

This task applies modular design by using user-defined functions to calculate Energy Charges and Fixed Charges. Each function performs a specific calculation and returns the result, making the billing logic reusable for multiple consumers. Meaningful comments improve code clarity and understanding. The main program reads inputs, calls the functions, and displays correct EC and FC values, verified using a test case.

Question 4:

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”

Expected Output

- Individual charge values printed
- Correct duty calculation
- Well-structured output
- Verified intermediate results

Prompt:

Extend the existing Python electricity billing program to calculate additional charges, including Fixed Charges (FC), Customer Charges (CC), and Electricity Duty (ED as a percentage of Energy Charges). Print each charge separately along with intermediate calculation results. Ensure accurate duty calculation, improved billing accuracy, and well-structured, readable output.

Code:

```python
# Electricity Billing Program using Modular Design

def calculate_energy_charges(units, customer_type):
    """
    Calculate energy charges based on units consumed and customer type.

    Args:
        units: Number of units consumed
        customer_type: Type of customer ('residential', 'commercial', 'industrial')

    Returns:
        Energy charges as a float
    """
    # Define rate per unit for each customer type
    rates = {
        'residential': 5.0,
        'commercial': 7.5,
        'industrial': 4.0
    }

    # Get rate for customer type (default to residential if not found)
    rate = rates.get(customer_type.lower(), 5.0)

    # Calculate and return energy charges
    energy_charges = units * rate
    return energy_charges


```

```python
def calculate_fixed_charges(customer_type):
    """
    Calculate fixed monthly charges based on customer type.

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

    Returns:
        Fixed charges as a float
    """
    # Define fixed charges for each customer type
    fixed_rates = {
        'residential': 100.0,
        'commercial': 300.0,
        'industrial': 500.0
    }

    # Get fixed charges for customer type (default to residential if not found)
    fixed_charges = fixed_rates.get(customer_type.lower(), 100.0)
    return fixed_charges
```

```python
def calculate_customer_charges(customer_type):
    """
    Calculate customer charges based on customer type.

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

    Returns:
        Customer charges as a float
    """
    # Define customer charges for each customer type
    customer_rates = {
        'residential': 50.0,
        'commercial': 150.0,
        'industrial': 250.0
    }

    # Get customer charges for customer type (default to residential if not found)
    customer_charges = customer_rates.get(customer_type.lower(), 50.0)
    return customer_charges

```

```python
def calculate_electricity_duty(energy_charges, duty_percentage=10.0):
    """
    Calculate electricity duty as a percentage of energy charges.

    Args:
        energy_charges: Energy charges amount
        duty_percentage: Duty percentage (default 10%)

    Returns:
        Electricity duty as a float
    """
    # Calculate duty based on percentage of energy charges
    electricity_duty = energy_charges * (duty_percentage / 100)
    return electricity_duty

```

```python
def main():
    """
    Main program to read inputs, calculate charges, and display results.
    """
    print("===== Electricity Billing System =====\n")

    # Read customer information
    customer_name = input("Enter customer name: ")
    customer_type = input("Enter customer type (residential/commercial/industrial): ")
    units = float(input("Enter units consumed: "))

    # Calculate charges using modular functions
    energy_charges = calculate_energy_charges(units, customer_type)
    fixed_charges = calculate_fixed_charges(customer_type)
    customer_charges = calculate_customer_charges(customer_type)
    electricity_duty = calculate_electricity_duty(energy_charges)

    # Calculate total bill
    total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty

    # Display results with detailed breakdown
    print("\n" + "="*50)
    print("BILL SUMMARY".center(50))
    print("="*50)
    print(f"\nCustomer Information:")
    print(f"  Customer Name: {customer_name}")
    print(f"  Customer Type: {customer_type.capitalize()}")
    print(f"  Units Consumed: {units}")

```

```
118     print(f"\n{'Charge Breakdown':^50}")
119     print("-"*50)
120     print(f"Energy Charges (EC):      ${energy_charges:>10.2f}")
121     print(f"  Calculation: {units} units × rate")
122     print(f"\nFixed Charges (FC):       ${fixed_charges:>10.2f}")
123 
124     print(f"\nCustomer Charges (CC):    ${customer_charges:>10.2f}")
125 
126     print(f"\nElectricity Duty (ED):    ${electricity_duty:>10.2f}")
127     print(f"  Calculation: {electricity_duty/energy_charges*100:.1f}% of Energy Charges")
128 
129     print("-"*50)
130     print(f"TOTAL BILL:               ${total_bill:>10.2f}")
131     print("="*50)
132 
133 
134 # Entry point of the program
135 if __name__ == "__main__":
136     main()
```

Output:

```
==================================================
                   BILL SUMMARY
==================================================

Customer Information:
  Customer Name: SRU
  Customer Type: Commercial
  Units Consumed: 200.0

                Charge Breakdown
--------------------------------------------------
Energy Charges (EC):      $   1500.00
  Calculation: 200.0 units × rate

Fixed Charges (FC):       $    300.00

Customer Charges (CC):    $    150.00

Electricity Duty (ED):    $    150.00
  Calculation: 10.0% of Energy Charges
--------------------------------------------------
TOTAL BILL:               $   2100.00
==================================================
```

Explanation:

In this task, the electricity billing program is extended to calculate additional charges such as Fixed Charges, Customer Charges, and Electricity Duty, which is computed as a percentage of Energy Charges. AI assistance helps improve billing accuracy and output structure. The program prints each charge separately and verifies intermediate results, ensuring correctness and clarity in the bill calculation.

Question 5:

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

Expected Output

• Complete electricity bill output

• Neatly formatted display

• Sample input/output
• Short analysis paragraph

Prompt:

Develop the final Python electricity billing application that calculates Energy Charges, Fixed Charges, Customer Charges, and Electricity Duty, then computes the Total Bill as EC + FC + CC + ED. Display each charge clearly along with the total bill amount in a neatly formatted output.

Code:

finalbill.py

20260113 > finalbill.py > ...

```python
def calculate_electricity_bill(units_consumed):
    # Define rates
    ENERGY_CHARGE_RATE = 5.0  # per unit in currency
    FIXED_CHARGE = 50.0       # fixed charge in currency
    CUSTOMER_CHARGE = 30.0    # customer charge in currency
    ELECTRICITY_DUTY_RATE = 0.05  # 5% of energy charges

    # Calculate charges
    energy_charges = units_consumed * ENERGY_CHARGE_RATE
    fixed_charges = FIXED_CHARGE
    customer_charges = CUSTOMER_CHARGE
    electricity_duty = energy_charges * ELECTRICITY_DUTY_RATE

    # Calculate total bill
    total_bill = energy_charges + fixed_charges + customer_charges + electricity_duty

    # Display the bill details
    print("Electricity Bill Breakdown:")
    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 Amount: {total_bill:.2f}")

    return total_bill

if __name__ == "__main__":
    try:
        # Input: Number of units consumed
        units = float(input("Enter the number of units consumed: "))
        if units < 0:
            print("Units consumed cannot be negative.")
        else:
            calculate_electricity_bill(units)
    except ValueError:
        print("Invalid input. Please enter a numeric value for units consumed.")
```

Output:

```
Enter the number of units consumed: 500
Electricity Bill Breakdown:
Energy Charges (EC): 2500.00
Fixed Charges (FC): 50.00
Customer Charges (CC): 30.00
Electricity Duty (ED): 125.00
Total Bill Amount: 2705.00
```

Explanation:

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.