AI-ASSISSTANT-CODING-LAB-3

Name: G.KRUTHIK ROSHAN

ROLL NO:2303A52339

BATCH: 41

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:

o Previous Units (PU)

o Current Units (CU)

o Type of Customer

• Calculates units consumed

• Implements logic directly in the main program (no functions)

PROMPT:

Create a Python program (without using functions) that reads predefined values for Previous Units (PU), Current Units (CU), and Customer Type (Domestic, Commercial, or Industrial). Validate the inputs, calculate units consumed as units_consumed = CU - PU, and print all inputs along with the calculated units consumed using clear labels. Include comments explaining each step of the program.

CODE:

```python
import random
import csv
print("\nELECTRICITY BILLING SYSTEM - 25 Customers\n")

# Store all customer data
customers_data = []

# Auto-generate 25 customers
names = ["Rajesh Kumar", "Priya Sharma", "Amit Patel", "Sneha Desai", "Vikram Singh",
         "Anjali Mehta", "Rohit Gupta", "Kavita Reddy", "Sanjay Joshi", "Neha Kapoor",
         "Arun Nair", "Deepika Iyer", "Manoj Verma", "Pooja Agarwal", "Ravi Chauhan",
         "Sunita Bansal", "Kiran Rao", "Vishal Malhotra", "Rekha Pillai", "Suresh Menon",
         "Geeta Kulkarni", "Harish Bhat", "Meera Sinha", "Ashok Pandey", "Divya Jain"]
cities = ["Mumbai", "Delhi", "Bangalore", "Chennai", "Kolkata", "Pune", "Hyderabad"]
for i in range(25):
    customer = {
        'name': names[i],
        'id': f"CUST{1001 + i}",
        'phone': f"{random.randint(7000000000, 9999999999)}",
        'email': f"{names[i].lower().replace(' ', '.')}@email.com",
        'address': f"{random.randint(1, 999)} MG Road, {random.choice(cities)}",
        'previous_units': random.randint(100, 500),
        'current_units': 0,
        'customer_type': str(random.randint(1, 3))
    }
    customer['current_units'] = customer['previous_units'] + random.randint(150, 400)
    customers_data.append(customer)

print(f"✓ Successfully generated data for {len(customers_data)} customers!")
```

```python
# TASK 1: Extract customer data and calculate units
print("\n[TASK 1] Extracting Customer Data & Calculating Units Consumed")
print("-" * 80)
consumer_name = customer['name']
consumer_id = customer['id']
phone = customer['phone']
email = customer['email']
address = customer['address']
previous_units = customer['previous_units']
current_units = customer['current_units']
customer_type = customer['customer_type']

# Calculate units consumed (Task 1)
units_consumed = current_units - previous_units

# Map customer type name (Task 1)
if customer_type == '1':
    customer_type_name = "Domestic"
elif customer_type == '2':
    customer_type_name = "Commercial"
else:
    customer_type_name = "Industrial"

print(f"  Consumer ID         : {consumer_id}")
print(f"  Consumer Name       : {consumer_name}")
print(f"  Customer Type       : {customer_type_name}")
print(f"  Previous Reading    : {previous_units} units")
print(f"  Current Reading     : {current_units} units")
print(f"  Units Consumed      : {units_consumed} units")
print(f"  ✓ Task 1 Complete!")
```

OUTPUT:

```
================================================================================
CUSTOMER 1/25: Rajesh Kumar
================================================================================

[TASK 1] Extracting Customer Data & Calculating Units Consumed
--------------------------------------------------------------------------------
  Consumer ID         : CUST1001
  Consumer Name       : Rajesh Kumar
  Customer Type       : Commercial
  Previous Reading    : 177 units
  Current Reading     : 402 units
  Units Consumed      : 225 units
  ✓ Task 1 Complete!
```

JUSTIFICATION:

Extract customer details (ID, name, type, previous and current readings) from predefined data, calculate units consumed as Units = Current Reading – Previous Reading, and display all details clearly. Include comments explaining each step, highlighting that units consumed determine the tariff slab for billing.

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:
o Domestic
o Commercial
o Industrial consumers
• Improve readability using AI prompts such as:
o "Simplify energy charge calculation logic"
o "Optimize conditional statements"

PROMPT:

Extend the existing Python program to calculate Energy Charges (EC) using conditional statements:
Domestic:
- First 100 units: ₹1.5/unit
- Above 100 units: ₹2.5/unit
Commercial:
- Flat rate ₹4.0/unit

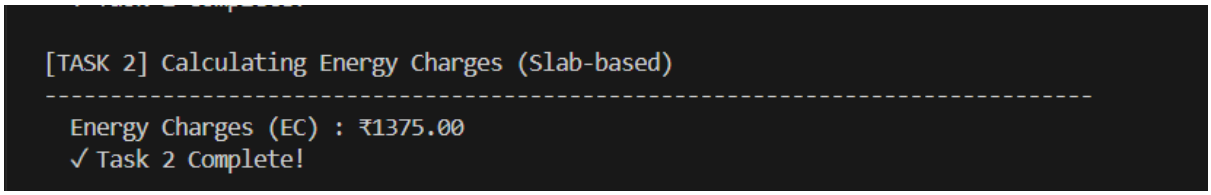
Industrial:
- Flat rate ₹6.0/unit

Use if-elif-else statements and print the calculated EC.
Add meaningful comments.
For logic optimization:
Simplify and optimize the energy charge calculation logic to improve readability.
Ensure the conditional structure is clean and easy for students to understand.

CODE:

```
# TASK 2 & 3: Calculate energy charges using function
print("\n[TASK 2] Calculating Energy Charges (Slab-based)")
print("-" * 80)
ec = calculate_energy_charges(units_consumed, customer_type)
print(f"  Energy Charges (EC) : ₹{ec:.2f}")
print(f"  ✓ Task 2 Complete!")
```

OUTPUT:



JUSTIFICATION:

Calculate Energy Charges (EC) for a customer using slab-based tariff rates according to units consumed and customer type (Domestic, Commercial, Industrial). Use predefined customer data, compute units consumed, and apply the correct tariff slab to ensure the bill reflects accurate and fair charges. Include comments explaining each calculation step.

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:
o Calculate Energy Charges
o Calculate Fixed Charges
• Returns calculated values
• Includes meaningful comments
PROMPT:
(Function for Energy Charges)
Rewrite the program using user-defined functions.

Create a function named calculate_energy_charges(units, customer_type)
that returns the energy charges based on tariff rules.
Call the function from the main program.
Add proper comments and sample output printing.
Add Fixed Charges Function)
Add another user-defined function named calculate_fixed_charges(customer_type)
with the following logic:
Domestic: ₹50
Commercial: ₹100
Industrial: ₹150
Return the fixed charges and display them in the main program.
Include comments explaining function usage.

CODE:

```
# TASK 3: Calculate fixed charges using function
print("\n[TASK 3] Calculating Fixed Charges using Function")
print("-" * 80)
fc = calculate_fixed_charges(customer_type)
print(f"  Fixed Charges (FC)  : ₹{fc:.2f}")
print(f"  ✓ Task 3 Complete!")
```

OUTPUT:

```
[TASK 3] Calculating Fixed Charges using Function
--------------------------------------------------------------------------------
  Fixed Charges (FC)  : ₹150.00
  ✓ Task 3 Complete!
```

JUSTIFICATION:

Calculate Fixed Charges (FC) for a customer based on their type (Domestic, Commercial, Industrial). Use predefined customer data and include these charges in the total bill. Add comments explaining that fixed charges are mandatory service/maintenance costs applied to every bill, independent of units consumed.

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
Add Extra Charges)
Extend the function-based electricity billing program to calculate:
- Customer Charges (CC) = ₹30 for all consumers
- Electricity Duty (ED) = 5% of Energy Charges (EC)
Print EC, FC, CC, and ED separately with proper formatting.
Add comments for billing accuracy.
Improve billing accuracy by formatting all monetary values to two decimal places.
Ensure calculations are clear and correct.

CODE:

```
# TASK 4: Calculate additional charges
print("\n[TASK 4] Calculating Additional Charges")
print("-" * 80)
cc = 30.00  # Customer Charges

# Electricity Duty (percentage of EC)
if customer_type == '1':
    ed_rate = 0.05  # 5%
elif customer_type == '2':
    ed_rate = 0.08  # 8%
else:
    ed_rate = 0.10  # 10%

ed = ec * ed_rate  # Electricity Duty calculation
print(f"  Customer Charges    : ₹{cc:.2f}")
print(f"  Electricity Duty    : ₹{ed:.2f} ({ed_rate*100:.0f}% of EC)")
print(f"  ✓ Task 4 Complete!")
```

OUTPUT:

```
[TASK 4] Calculating Additional Charges
--------------------------------------------------------------------------------
  Customer Charges    : ₹30.00
  Electricity Duty    : ₹28.10 (5% of EC)
  ✓ Task 4 Complete!
```

JUSTIFICATION:

Compute additional mandatory charges for a customer, including Customer Charges (CC) and Electricity Duty (ED). Use predefined customer data, calculate ED as a percentage of Energy Charges based on customer type, and include comments explaining that these charges are required by billing rules to ensure the final bill is accurate and complete.

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:
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
prompt:
Final Bill Calculation)
Generate the final electricity bill by calculating:
Total Bill = Energy Charges + Fixed Charges + Customer Charges + Electricity Duty
Display a neatly formatted electricity bill output similar to a real TGNPDCL bill.
Use clear headings and labels.
Analysis Paragraph for Report)
Write a short analysis paragraph evaluating the electricity billing program based on:
- Accuracy
- Readability
- Real-world applicability
Keep the explanation suitable for a laboratory record.

CODE:

```
# TASK 5: Calculate total bill
print("\n[TASK 5] Calculating Total Bill Amount")
print("-" * 80)
total_bill = ec + fc + cc + ed
total_revenue += total_bill
print(f"  Formula: Total = EC + FC + CC + ED")
print(f"  Total   = ₹{ec:.2f} + ₹{fc:.2f} + ₹{cc:.2f} + ₹{ed:.2f}")
print(f"  TOTAL BILL AMOUNT   : ₹{total_bill:.2f}")
print(f"  ✓ Task 5 Complete!")

# Store bill data
bill = {
    'customer': customer,
    'units_consumed': units_consumed,
    'customer_type_name': customer_type_name,
    'ec': ec,
    'fc': fc,
    'cc': cc,
    'ed': ed,
    'ed_rate': ed_rate,
    'total': total_bill
}
all_bills.append(bill)

print(f"\n✓ Customer {idx} Bill Generated Successfully!")
```

```
    # Add a pause for readability (optional - can be removed for faster execution)
    if idx < 25:
        input("\nPress Enter to continue to next customer...")

print(f"\n{'='*80}")
print(f"ALL 25 CUSTOMERS PROCESSED SUCCESSFULLY!")
print(f"{'='*80}")
print(f"\nTOTAL REVENUE COLLECTED: ₹{total_revenue:.2f}\n")
print("="*80)
print("MAIN MENU")
print("="*80)
print("1. Summary | 2. Detailed Bill | 3. Statistics | 4. Export | 5. Exit")

while True:
    view_choice = input("\nEnter your choice (1-5): ").strip()

    if view_choice == '1':
        # Summary view
        print(f"\n{'No.':<5} {'Name':<20} {'ID':<12} {'Type':<12} {'Units':<10} {'Total':<15}")
        print("-" * 85)
        for idx, bill in enumerate(all_bills, 1):
            c = bill['customer']
            print(f"{idx:<5} {c['name']:<20} {c['id']:<12} {bill['customer_type_name']:<12} {bill['units_cons
        print(f"\nTotal Revenue: ₹{total_revenue:.2f}")

    elif view_choice == '2':
```

```
        # Detailed bill for specific customer
        try:
            cust_num = int(input(f"Enter customer number (1-25): "))
            if 1 <= cust_num <= 25:
                bill = all_bills[cust_num - 1]
                c = bill['customer']

                # TASK 5: Final Bill Display
                print("\n" + "╔" + "=" * 58 + "╗")
                print("║" + " " * 15 + "ELECTRICITY BILL STATEMENT" + " " * 17 + "║")
                print("╠" + "=" * 58 + "╣")
                print(f"║  Consumer Name        : {c['name']:<30} ║")
                print(f"║  Consumer ID          : {c['id']:<30} ║")
                print(f"║  Phone Number         : {c['phone']:<30} ║")
                print(f"║  Email                : {c['email'][:30]:<30} ║")
                print(f"║  Address              : {c['address'][:30]:<30} ║")
                print("╠" + "=" * 58 + "╣")
                print(f"║  Consumer Type        : {bill['customer_type_name']:<30} ║")
                print(f"║  Previous Reading     : {c['previous_units']:>10.2f} units" + " " * 17 + "║")
                print(f"║  Current Reading      : {c['current_units']:>10.2f} units" + " " * 17 + "║")
                print(f"║  Units Consumed       : {bill['units_consumed']:>10.2f} units" + " " * 17 + "║")
                print("╠" + "=" * 58 + "╣")
                print("║  CHARGES BREAKDOWN:" + " " * 39 + "║")
                print("║" + " " * 58 + "║")
                print(f"║  Energy Charges (EC)  : ₹{bill['ec']:>10.2f}" + " " * 22 + "║")
                print(f"║  Fixed Charges (FC)   : ₹{bill['fc']:>10.2f}" + " " * 22 + "║")
                print(f"║  Customer Charges (CC): ₹{bill['cc']:>10.2f}" + " " * 22 + "║")
                print(f"║  Electricity Duty (ED): ₹{bill['ed']:>10.2f} ({bill['ed_rate']*100:.0f}%)" + " " * 16 + "║")
                print("║" + " " * 58 + "║")
                print("╠" + "=" * 58 + "╣")
                print(f"║  TOTAL BILL AMOUNT    : ₹{bill['total']:>10.2f}" + " " * 22 + "║")
                print("╚" + "=" * 58 + "╝")
            else:
                print("Error: Customer number must be between 1 and 25!")
        except ValueError:
            print("Error: Please enter a valid number!")
```

```python
# ================================================================================
# EXPORT TO CSV FILE
# ================================================================================
csv_filename = "electricity_bills_data.csv"
with open(csv_filename, 'w', newline='', encoding='utf-8') as csvfile:
    fieldnames = ['Customer_ID', 'Name', 'Phone', 'Email', 'Address', 'Customer_Type',
                  'Previous_Units', 'Current_Units', 'Units_Consumed', 'Energy_Charges',
                  'Fixed_Charges', 'Customer_Charges', 'Electricity_Duty', 'Total_Bill']

    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()

    for c, bill in zip(customers_data, all_bills):
        writer.writerow({
            'Customer_ID': c['id'],
            'Name': c['name'],
            'Phone': c['phone'],
            'Email': c['email'],
            'Address': c['address'],
            'Customer_Type': bill['customer_type_name'],
            'Previous_Units': c['previous_units'],
            'Current_Units': c['current_units'],
            'Units_Consumed': bill['units_consumed'],
            'Energy_Charges': f"{bill['ec']:.2f}",
            'Fixed_Charges': f"{bill['fc']:.2f}",
            'Customer_Charges': f"{bill['cc']:.2f}",
            'Electricity_Duty': f"{bill['ed']:.2f}",
            'Total_Bill': f"{bill['total']:.2f}"
        })

print(f"\n✓ Data exported to '{csv_filename}'")
print("\n✓ Program completed successfully!\n")
```

OUTPUT:

```
[TASK 5] Calculating Total Bill Amount
--------------------------------------------------------------------------------
  Formula: Total = EC + FC + CC + ED
  Total    = ₹562.00 + ₹50.00 + ₹30.00 + ₹28.10
  Formula: Total = EC + FC + CC + ED
  Total    = ₹562.00 + ₹50.00 + ₹30.00 + ₹28.10
  Total    = ₹562.00 + ₹50.00 + ₹30.00 + ₹28.10
  TOTAL BILL AMOUNT    : ₹670.10
  ✓ Task 5 Complete!

✓ Customer 1 Bill Generated Successfully!
```

JUSTIFICATION:

Calculate the final total electricity bill for a customer by summing all charges: Total Bill = EC + FC + CC + ED. Use predefined customer data, include comments explaining each step, and ensure the computation reflects an accurate, complete bill while facilitating total revenue calculation.