Name:T.Ramyasri

H.no:2303A52136

Assignment-3.3

TASK-1:PROMPT:

Generate a Python program that reads previous units, current units, and customer type from the user, calculates the units consumed, and prints the result. Do not use functions.

```
# Simple program to read previous units, current units, and customer type,
# calculate units consumed, and print the result.
# No functions are used.

prev_input = input("Enter previous units: ").strip()
curr_input = input("Enter current units: ").strip()
customer_type = input("Enter customer type: ").strip()

# Convert inputs to numbers (allows integer or decimal values)
try:
    previous = float(prev_input)
    current = float(curr_input)
except ValueError:
    print("Error: Previous and current units must be numeric.")
    raise SystemExit(1)

# Basic validation
if previous < 0 or current < 0:
    print("Error: Unit readings must be non-negative.")
    raise SystemExit(1)

if current < previous:
    print("Error: Current units cannot be less than previous units.")
    raise SystemExit(1)

units_consumed = current - previous

# Print result (show as integer if it's a whole number)
if units_consumed.is_integer():
    units_consumed_display = int(units_consumed)
else:
    units_consumed_display = round(units_consumed, 2)

print(f"Customer type: {customer_type}")
print(f"Units consumed: {units_consumed_display}")
```

```
Enter previous units: 1000
Enter current units: 1250
Enter customer type: Domestic
Customer type: Domestic
Units consumed: 250
```

How can I install Python libraries? | Load data from Google Drive | Show an example of training a

What can I help you build?

Gemini 2.5 Flash

TASK-2:

PROMPT:

Extend the electricity billing code to calculate energy charges based on customer type using if-else conditions for Domestic, Commercial, and Industrial consumers.”

```python
# Electricity billing calculator (no functions)
# Reads previous units, current units, and customer type,
# computes units consumed and energy charges for
# Domestic, Commercial, and Industrial consumers using if-else.

prev_input = input("Enter previous units: ").strip()
curr_input = input("Enter current units: ").strip()
customer_type = input("Enter customer type (Domestic/Commercial/Industrial): ").strip()

# Convert inputs to numbers
try:
    previous = float(prev_input)
    current = float(curr_input)
except ValueError:
    print("Error: Previous and current units must be numeric.")
    raise SystemExit(1)

# Basic validation
if previous < 0 or current < 0:
    print("Error: Unit readings must be non-negative.")
    raise SystemExit(1)

if current < previous:
    print("Error: Current units cannot be less than previous units.")
    raise SystemExit(1)

units_consumed = current - previous

# Normalize customer type for comparison
ctype = customer_type.strip().lower()
```

```
# Define per-unit rates (modify these values as needed)
# These are simple flat rates per unit for each customer type.
domestic_rate = 1.50     # currency units per unit
commercial_rate = 3.00   # currency units per unit
industrial_rate = 2.50   # currency units per unit

# Determine rate using if-else based on customer type
if ctype in ("domestic", "d"):
    rate = domestic_rate
    ctype_display = "Domestic"
elif ctype in ("commercial", "c"):
    rate = commercial_rate
    ctype_display = "Commercial"
elif ctype in ("industrial", "i"):
    rate = industrial_rate
    ctype_display = "Industrial"
else:
    print("Error: Unknown customer type. Please enter Domestic, Commercial, or Industrial.")
    raise SystemExit(1)

# Calculate energy charge
energy_charge = units_consumed * rate

# Format units_consumed display (integer if whole number)
if units_consumed.is_integer():
    units_display = int(units_consumed)
else:
    units_display = round(units_consumed, 2)
```

3.test with Domestic

```
    units_display = round(units_consumed, 2)

# Print bill summary
print("\n--- Bill Summary ---")
print(f"Customer type: {ctype_display}")
print(f"Previous reading: {previous}")
print(f"Current reading: {current}")
print(f"Units consumed: {units_display}")
print(f"Rate per unit: {rate:.2f}")
print(f"Energy charge: {energy_charge:.2f}")
```

```
Enter previous units: 1000
Enter current units: 1250
Enter customer type (Domestic/Commercial/Industrial): Domestic

--- Bill Summary ---
Customer type: Domestic
Previous reading: 1000.0
Current reading: 1250.0
Units consumed: 250
Rate per unit: 1.50
Energy charge: 375.00
```

Test with commercial

```
print(f"Units consumed: {units_display}")
print(f"Rate per unit: {rate:.2f}")
print(f"Energy charge: {energy_charge:.2f}")
```

```
Enter previous units: 1000
Enter current units: 1250
Enter customer type (Domestic/Commercial/Industrial): Commercial

--- Bill Summary ---
Customer type: Commercial
Previous reading: 1000.0
Current reading: 1250.0
Units consumed: 250
Rate per unit: 3.00
Energy charge: 750.00
```

4.Prompt Improvement with AI:

“Simplify energy charge calculation logic”
“Optimize conditional statements for better readability”

5.Comparision bet

What I changed and why:

- Replaced the long if/elif chain with a simple dictionary lookup for rates to improve readability and make it easy to update or extend tariffs.
- Kept input parsing and validation compact and clear.
- Kept all logic inline (no functions) as requested.
- Next steps you might want: switch to slab-based rates (if tariffs are tiered), add taxes/fees, or validate/normalize customer-type variations further.

```python
# Simplified electricity billing calculator (no functions)
# - Uses a rates lookup dict instead of multiple if/elif branches
# - Clear, compact validation and output

prev_input = input("Enter previous units: ").strip()
curr_input = input("Enter current units: ").strip()
customer_type = input("Enter customer type (Domestic/Commercial/Industrial): ").strip()

# Parse numeric inputs
try:
    previous = float(prev_input)
    current = float(curr_input)
except ValueError:
    print("Error: Previous and current units must be numeric.")
    raise SystemExit(1)

# Validate readings
if previous < 0 or current < 0:
    print("Error: Readings must be non-negative.")
    raise SystemExit(1)
if current < previous:
    print("Error: Current reading cannot be less than previous reading.")
    raise SystemExit(1)

units_consumed = current - previous

# Rate lookup (flat rates). Change values here to update tariffs.
rates = {
    "domestic": 1.50,
    "d": 1.50,
    "commercial": 3.00,
    "c": 3.00,
    "industrial": 2.50,
    "i": 2.50,
}
```

```python
key = customer_type.strip().lower()
rate = rates.get(key)
if rate is None:
    print("Error: Unknown customer type. Use Domestic, Commercial, or Industrial (or D/C/I).")
    raise SystemExit(1)

energy_charge = units_consumed * rate

# Nicely format units (integer when whole) and currency
units_display = int(units_consumed) if units_consumed.is_integer() else round(units_consumed, 2)

print("\n--- Bill Summary ---")
print(f"Customer type: {customer_type.strip().title()}")
print(f"Previous reading: {previous}")
print(f"Current reading:  {current}")
print(f"Units consumed:   {units_display}")
print(f"Rate per unit:    {rate:.2f}")
print(f"Energy charge:    {energy_charge:.2f}")
```

```
Enter previous units: 1000
Enter current units: 1250
Enter customer type (Domestic/Commercial/Industrial): Domestic

--- Bill Summary ---
Customer type: Domestic
Previous reading: 1000.0
Current reading:  1250.0
Units consumed:   250
Rate per unit:    1.50
Energy charge:    375.00
```

TASK-03:

PROMPT:

“Rewrite the electricity billing program using functions to calculate Energy Charges and Fixed Charges. Include comments explaining each function.”

```python
# Electricity billing program using functions
# - calculate_energy_charge(units, customer_type): computes energy charge from per-unit rates
# - calculate_fixed_charge(customer_type): returns fixed monthly charge for the customer type
# The main section reads inputs, validates them, calls the functions, and prints the bill.

def calculate_energy_charge(units, customer_type):
    """
    Calculate the energy (consumption) charge.
    Parameters:
    - units (float): number of units consumed
    - customer_type (str): customer category (Domestic, Commercial, Industrial, or short forms D/C/I)
    Returns:
    - tuple (energy_charge (float), rate_per_unit (float))
    Raises:
    - ValueError if customer_type is unrecognized
    Notes:
    - Uses a simple flat per-unit rate lookup. Modify the 'rates' dict to change tariffs.
    """
    # Per-unit flat rates (currency units per unit)
    rates = {
        "domestic": 1.50,
        "d": 1.50,
        "commercial": 3.00,
        "c": 3.00,
        "industrial": 2.50,
        "i": 2.50,
    }

    key = customer_type.strip().lower()
    rate = rates.get(key)
    if rate is None:
        raise ValueError("Unknown customer type for energy rate. Use Domestic, Commercial, or Industrial (or D

    energy_charge = units * rate
```

```python
def calculate_fixed_charge(customer_type):
    """
    Return the fixed (standing) charge for the billing period based on customer type.
    Parameters:
    - customer_type (str): customer category (Domestic, Commercial, Industrial, or short forms D/C/I)
    Returns:
    - fixed_charge (float)
    Raises:
    - ValueError if customer_type is unrecognized
    Notes:
    - These are example fixed charges. Modify the 'fixed_charges' dict to change values.
    """
    fixed_charges = {
        "domestic": 50.00,     # flat monthly fixed charge for domestic customers
        "d": 50.00,
        "commercial": 100.00,  # flat monthly fixed charge for commercial customers
        "c": 100.00,
        "industrial": 200.00,  # flat monthly fixed charge for industrial customers
        "i": 200.00,
    }

    key = customer_type.strip().lower()
    fixed = fixed_charges.get(key)
    if fixed is None:
        raise ValueError("Unknown customer type for fixed charge. Use Domestic, Commercial, or Industrial (or
    return fixed


# -------------------- Main script --------------------

# Read inputs
prev_input = input("Enter previous units: ").strip()
```

```
curr_input = input("Enter current units: ").strip()
customer_type = input("Enter customer type (Domestic/Commercial/Industrial): ").strip()

# Parse numeric inputs
try:
    previous = float(prev_input)
    current = float(curr_input)
except ValueError:
    print("Error: Previous and current units must be numeric.")
    raise SystemExit(1)

# Validate readings
if previous < 0 or current < 0:
    print("Error: Readings must be non-negative.")
    raise SystemExit(1)
if current < previous:
    print("Error: Current reading cannot be less than previous reading.")
    raise SystemExit(1)

# Compute units consumed
units_consumed = current - previous

# Use functions to compute charges
try:
    energy_charge, rate_per_unit = calculate_energy_charge(units_consumed, customer_type)
    fixed_charge = calculate_fixed_charge(customer_type)
except ValueError as e:
    print(f"Error: {e}")
    raise SystemExit(1)

# Total bill
total_amount = energy_charge + fixed_charge

# Nicely format units (integer when whole) and currency
units_display = int(units_consumed) if units_consumed.is_integer() else round(units_consumed, 2)
```

```
    print("\n--- Bill Summary ---")
    print(f"Customer type:     {customer_type.strip().title()}")
    print(f"Previous reading:  {previous}")
    print(f"Current reading:   {current}")
    print(f"Units consumed:    {units_display}")
    print(f"Rate per unit:     {rate_per_unit:.2f}")
    print(f"Energy charge:     {energy_charge:.2f}")
    print(f"Fixed charge:      {fixed_charge:.2f}")
    print(f"Total amount due:  {total_amount:.2f}")
```

```
Enter previous units: 1000
Enter current units: 1250
Enter customer type (Domestic/Commercial/Industrial): Domestic

--- Bill Summary ---
Customer type:     Domestic
Previous reading:  1000.0
Current reading:   1250.0
Units consumed:    250
Rate per unit:     1.50
Energy charge:     375.00
Fixed charge:      50.00
Total amount due:  425.00
```

```
Enter previous units: 1000
Enter current units: 1250
Enter customer type (Domestic/Commercial/Industrial): Commercial

--- Bill Summary ---
Customer type:      Commercial
Previous reading:   1000.0
Current reading:    1250.0
Units consumed:     250
Rate per unit:      3.00
Energy charge:      750.00
Fixed charge:       100.00
Customer charge:    60.00
Electricity duty (10%): 75.00
Total amount due:   985.00
```

```
Enter previous units: 1000
Enter current units: 1250
Enter customer type (Domestic/Commercial/Industrial): Industrial

--- Bill Summary ---
Customer type:      Industrial
Previous reading:   1000.0
Current reading:    1250.0
Units consumed:     250
Rate per unit:      2.50
Energy charge:      625.00
Fixed charge:       200.00
Customer charge:    120.00
Electricity duty (10%): 62.50
Total amount due:   1007.50
```

TASK-4:

```python
# Electricity billing program using functions
# - calculate_energy_charge(units, customer_type): computes energy charge from per-unit rates
# - calculate_fixed_charge(customer_type): returns fixed monthly/standing charge for customer type
# - calculate_customer_charge(customer_type): returns customer-specific charge (additional)
# - calculate_electricity_duty(energy_charge): computes duty as 10% of energy charge
# The main section reads inputs, validates them, calls the functions, and prints the bill.

def calculate_energy_charge(units, customer_type):
    """
    Calculate the energy (consumption) charge.
    Parameters:
    - units (float): number of units consumed
    - customer_type (str): customer category (Domestic, Commercial, Industrial, or short forms D/C/I)
    Returns:
    - tuple (energy_charge (float), rate_per_unit (float))
    Raises:
    - ValueError if customer_type is unrecognized
    Notes:
    - Uses flat per-unit rates. Modify the 'rates' dict to change tariffs.
    """
    rates = {
        "domestic": 1.50,
        "d": 1.50,
        "commercial": 3.00,
        "c": 3.00,
        "industrial": 2.50,
        "i": 2.50,
    }

    key = customer_type.strip().lower()
    rate = rates.get(key)
    if rate is None:
        raise ValueError("Unknown customer type for energy rate. Use Domestic, Commercial, or Industrial (or D

    energy_charge = units * rate
    return energy_charge, rate


def calculate_fixed_charge(customer_type):
    """
    Return the fixed (standing) charge for the billing period based on customer type.
    Parameters:
    - customer_type (str): customer category (Domestic, Commercial, Industrial, or short forms D/C/I)
    Returns:
    - fixed_charge (float)
    Raises:
    - ValueError if customer_type is unrecognized
    Notes:
    - Modify the 'fixed_charges' dict to change values.
    """
    fixed_charges = {
        "domestic": 50.00,
        "d": 50.00,
        "commercial": 100.00,
        "c": 100.00,
        "industrial": 200.00,
        "i": 200.00,
    }

    key = customer_type.strip().lower()
    fixed = fixed_charges.get(key)
    if fixed is None:
        raise ValueError("Unknown customer type for fixed charge. Use Domestic, Commercial, or Industrial (or
    return fixed
```

```python
def calculate_customer_charge(customer_type):
    """
    Return a customer charge which is an additional charge depending on customer type.
    Parameters:
    - customer_type (str): customer category (Domestic, Commercial, Industrial, or short forms D/C/I)
    Returns:
    - customer_charge (float)
    Raises:
    - ValueError if customer_type is unrecognized
    Notes:
    - These are example values to illustrate a per-customer-type surcharge.
    """
    customer_charges = {
        "domestic": 30.00,
        "d": 30.00,
        "commercial": 60.00,
        "c": 60.00,
        "industrial": 120.00,
        "i": 120.00,
    }

    key = customer_type.strip().lower()
    cust_charge = customer_charges.get(key)
    if cust_charge is None:
        raise ValueError("Unknown customer type for customer charge. Use Domestic, Commercial, or Industrial (
    return cust_charge
```

```python
def calculate_electricity_duty(energy_charge):
    """
    Calculate electricity duty as 10% of the energy charge.
    Parameters:
    - energy_charge (float)
    Returns:
    - duty (float)
    """
    duty_rate = 0.10  # 10%
    return energy_charge * duty_rate


# -------------------- Main script --------------------

# Read inputs
prev_input = input("Enter previous units: ").strip()
curr_input = input("Enter current units: ").strip()
customer_type = input("Enter customer type (Domestic/Commercial/Industrial): ").strip()

# Parse numeric inputs
try:
    previous = float(prev_input)
    current = float(curr_input)
except ValueError:
    print("Error: Previous and current units must be numeric.")
    raise SystemExit(1)

# Validate readings
if previous < 0 or current < 0:
    print("Error: Readings must be non-negative.")
    raise SystemExit(1)
if current < previous:
    print("Error: Current reading cannot be less than previous reading.")
    raise SystemExit(1)
```

```python
# Compute units consumed
units_consumed = current - previous

# Use functions to compute charges
try:
    energy_charge, rate_per_unit = calculate_energy_charge(units_consumed, customer_type)
    fixed_charge = calculate_fixed_charge(customer_type)
    customer_charge = calculate_customer_charge(customer_type)
    electricity_duty = calculate_electricity_duty(energy_charge)
except ValueError as e:
    print(f"Error: {e}")
    raise SystemExit(1)

# Total bill
total_amount = energy_charge + fixed_charge + customer_charge + electricity_duty

# Nicely format units (integer when whole) and currency
units_display = int(units_consumed) if units_consumed.is_integer() else round(units_consumed, 2)

print("\n--- Bill Summary ---")
print(f"Customer type:      {customer_type.strip().title()}")
print(f"Previous reading:   {previous}")
print(f"Current reading:    {current}")
print(f"Units consumed:     {units_display}")
print(f"Rate per unit:      {rate_per_unit:.2f}")
print(f"Energy charge:      {energy_charge:.2f}")
print(f"Fixed charge:       {fixed_charge:.2f}")
print(f"Customer charge:    {customer_charge:.2f}")
print(f"Electricity duty (10%): {electricity_duty:.2f}")
print(f"Total amount due:   {total_amount:.2f}")
```

```
print(f"Total amount due:   {total_amount:.2f}")
```

```
Enter previous units: 1000
Enter current units: 1250
Enter customer type (Domestic/Commercial/Industrial): Domestic

--- Bill Summary ---
Customer type:      Domestic
Previous reading:   1000.0
Current reading:    1250.0
Units consumed:     250
Rate per unit:      1.50
Energy charge:      375.00
Fixed charge:       50.00
Customer charge:    30.00
Electricity duty (10%): 37.50
Total amount due:   492.50
```

Task-05

Prompt: “Complete the electricity billing system by calculating and printing a detailed bill including EC, FC, CC, ED, and total.”

```python
# Complete Electricity Billing System
# - Calculates and prints a detailed bill including:
#   EC : Energy Charge
#   FC : Fixed Charge
#   CC : Customer Charge
#   ED : Electricity Duty (10% of EC)
#   Total amount due
# - Uses functions to compute each charge (each function includes a docstring)
# - Main script handles input, validation, and prints a neatly formatted bill

def calculate_energy_charge(units, customer_type):
    """
    Calculate the energy (consumption) charge (EC).
    Parameters:
    - units (float): number of units consumed
    - customer_type (str): customer category (Domestic, Commercial, Industrial, or short forms D/C/I)
    Returns:
    - tuple (energy_charge (float), rate_per_unit (float))
    Raises:
    - ValueError if customer_type is unrecognized
    Notes:
    - Uses flat per-unit rates. Modify the 'rates' dict to update tariffs.
    """
    rates = {
        "domestic": 1.50,
        "d": 1.50,
        "commercial": 3.00,
        "c": 3.00,
        "industrial": 2.50,
        "i": 2.50,
    }

    key = customer_type.strip().lower()
    rate = rates.get(key)
    if rate is None:
        raise ValueError("Unknown customer type for energy rate. Use Domestic, Commercial, or Industrial (or D/C/I).")

    energy_charge = units * rate
    return energy_charge, rate


def calculate_fixed_charge(customer_type):
    """
    Return the fixed (standing) charge (FC) for the billing period based on customer type.
    Parameters:
    - customer_type (str)
    Returns:
    - fixed_charge (float)
    Raises:
    - ValueError if customer_type is unrecognized
    """
    fixed_charges = {
        "domestic": 50.00,
        "d": 50.00,
        "commercial": 100.00,
        "c": 100.00,
        "industrial": 200.00,
        "i": 200.00,
    }

    key = customer_type.strip().lower()
    fixed = fixed_charges.get(key)
    if fixed is None:
        raise ValueError("Unknown customer type for fixed charge. Use Domestic, Commercial, or Industrial (or D/C/I).")
    return fixed
```

```python
def calculate_customer_charge(customer_type):
    """
    Return an additional customer charge (CC) depending on customer type.
    Parameters:
    - customer_type (str)
    Returns:
    - customer_charge (float)
    Raises:
    - ValueError if customer_type is unrecognized
    """
    customer_charges = {
        "domestic": 30.00,
        "d": 30.00,
        "commercial": 60.00,
        "c": 60.00,
        "industrial": 120.00,
        "i": 120.00,
    }

    key = customer_type.strip().lower()
    cust_charge = customer_charges.get(key)
    if cust_charge is None:
        raise ValueError("Unknown customer type for customer charge. Use Domestic, Commercial, or Industrial (
    return cust_charge
```

```python
def calculate_electricity_duty(energy_charge):
    """
    Calculate electricity duty (ED) as 10% of the energy charge (EC).
    Parameters:
    - energy_charge (float)
    Returns:
    - duty (float)
    """
    duty_rate = 0.10  # 10%
    return energy_charge * duty_rate


# -------------------- Main script --------------------

# Read inputs
prev_input = input("Enter previous units: ").strip()
curr_input = input("Enter current units: ").strip()
customer_type = input("Enter customer type (Domestic/Commercial/Industrial): ").strip()

# Parse numeric inputs
try:
    previous = float(prev_input)
    current = float(curr_input)
except ValueError:
    print("Error: Previous and current units must be numeric.")
    raise SystemExit(1)

# Validate readings
if previous < 0 or current < 0:
    print("Error: Readings must be non-negative.")
    raise SystemExit(1)
if current < previous:
    print("Error: Current reading cannot be less than previous reading.")
    raise SystemExit(1)
```

```python
# Compute units consumed
units_consumed = current - previous

# Compute charges using functions
try:
    ec, rate_per_unit = calculate_energy_charge(units_consumed, customer_type)  # Energy Charge (EC)
    fc = calculate_fixed_charge(customer_type)                                  # Fixed Charge (FC)
    cc = calculate_customer_charge(customer_type)                               # Customer Charge (CC)
    ed = calculate_electricity_duty(ec)                                         # Electricity Duty (ED)
except ValueError as e:
    print(f"Error: {e}")
    raise SystemExit(1)

# Total amount due
total_amount = ec + fc + cc + ed

# Format values for display
units_display = int(units_consumed) if units_consumed.is_integer() else round(units_consumed, 2)
rate_display = f"{rate_per_unit:.2f}"
ec_display = f"{ec:.2f}"
fc_display = f"{fc:.2f}"
cc_display = f"{cc:.2f}"
ed_display = f"{ed:.2f}"
total_display = f"{total_amount:.2f}"

# Print a detailed bill
print("\n==================== DETAILED BILL ====================")
print(f"Customer type      : {customer_type.strip().title()}")
print(f"Previous reading   : {previous}")
print(f"Current reading    : {current}")
print(f"Units consumed     : {units_display}")
print("-------------------------------------------------------")
print(f"EC (Energy Charge) : {ec_display}  @ {rate_display} per unit")
print(f"FC (Fixed Charge)  : {fc_display}")
print(f"CC (Customer Chg.) : {cc_display}")
```

```python
# Print a detailed bill
print("\n==================== DETAILED BILL ====================")
print(f"Customer type      : {customer_type.strip().title()}")
print(f"Previous reading   : {previous}")
print(f"Current reading    : {current}")
print(f"Units consumed     : {units_display}")
print("-------------------------------------------------------")
print(f"EC (Energy Charge) : {ec_display}  @ {rate_display} per unit")
print(f"FC (Fixed Charge)  : {fc_display}")
print(f"CC (Customer Chg.) : {cc_display}")
print(f"ED (Electricity Duty, 10% of EC) : {ed_display}")
print("-------------------------------------------------------")
print(f"TOTAL AMOUNT DUE   : {total_display}")
print("=======================================================\n")
```

```
Enter previous units: 1200
Enter current units: 1450
Enter customer type (Domestic/Commercial/Industrial): Commercial

==================== DETAILED BILL ====================
Customer type      : Commercial
Previous reading   : 1200.0
Current reading    : 1450.0
Units consumed     : 250
-------------------------------------------------------
EC (Energy Charge) : 750.00  @ 3.00 per unit
FC (Fixed Charge)  : 100.00
CC (Customer Chg.) : 60.00
ED (Electricity Duty, 10% of EC) : 75.00
-------------------------------------------------------
TOTAL AMOUNT DUE   : 985.00
=======================================================
```

Analysis (for Report):

- **Accuracy:** Verified against manual calculation.
- **Readability:** Improved via functions, comments, and formatted output.
- **Applicability:** Modular design suitable for multi-customer billing systems.