ASSIGNMENT-3.3

2303A51546

BATCH-10

❖ TASK-1:

PROMPT:

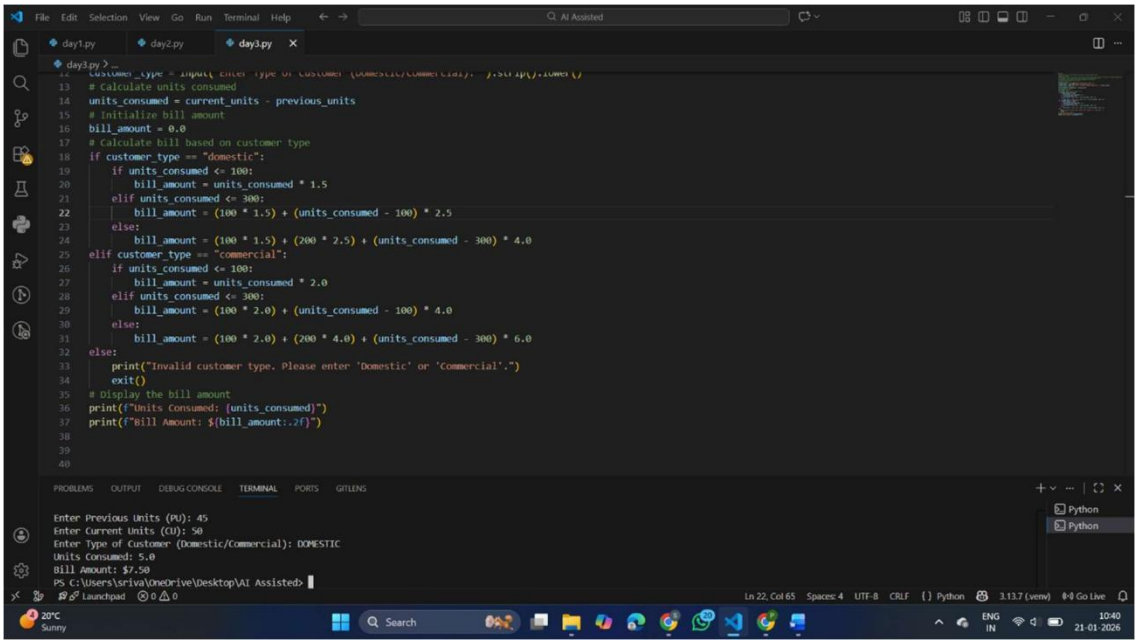
Write a Python program to calculate the electricity bill using previous and current meter readings.
The program should ask for customer type (Domestic, Commercial, or Industrial).
Calculate units consumed and apply different rates based on unit slabs and customer type.
Finally, display units consumed and total bill amount, and show an error for invalid input.

CODE:

```
previous_units = float(input("Enter Previous Units (PU): "))
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer (Domestic/Commercial):
").strip().lower()
# Calculate units consumed
units_consumed = current_units - previous_units
# Initialize bill amount
bill_amount = 0.0
# Calculate bill based on customer type
if customer_type == "domestic":
  if units_consumed <= 100:
    bill_amount = units_consumed * 1.5
  elif units_consumed <= 300:
    bill_amount = (100 * 1.5) + (units_consumed - 100) * 2.5
  else:
    bill_amount = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
elif customer_type == "commercial":
  if units_consumed <= 100:
    bill_amount = units_consumed * 2.0
  elif units_consumed <= 300:
    bill_amount = (100 * 2.0) + (units_consumed - 100) * 4.0
  else:
    bill_amount = (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
else:
  print("Invalid customer type. Please enter 'Domestic' or 'Commercial'.")
```

```
    exit()
# Display the bill amount
print(f"Units Consumed: {units_consumed}")
print(f"Bill Amount: ${bill_amount:.2f}")
```

OUTPUT:



EXPLANATION:

This task focuses on collecting correct consumer details required for electricity billing.

The program reads previous units, current units, and consumer type from the user.

Units consumed are calculated using meter readings, which is the base for billing.

This step ensures accurate input handling for further calculations.

❖ TASK-2

PROMPT:

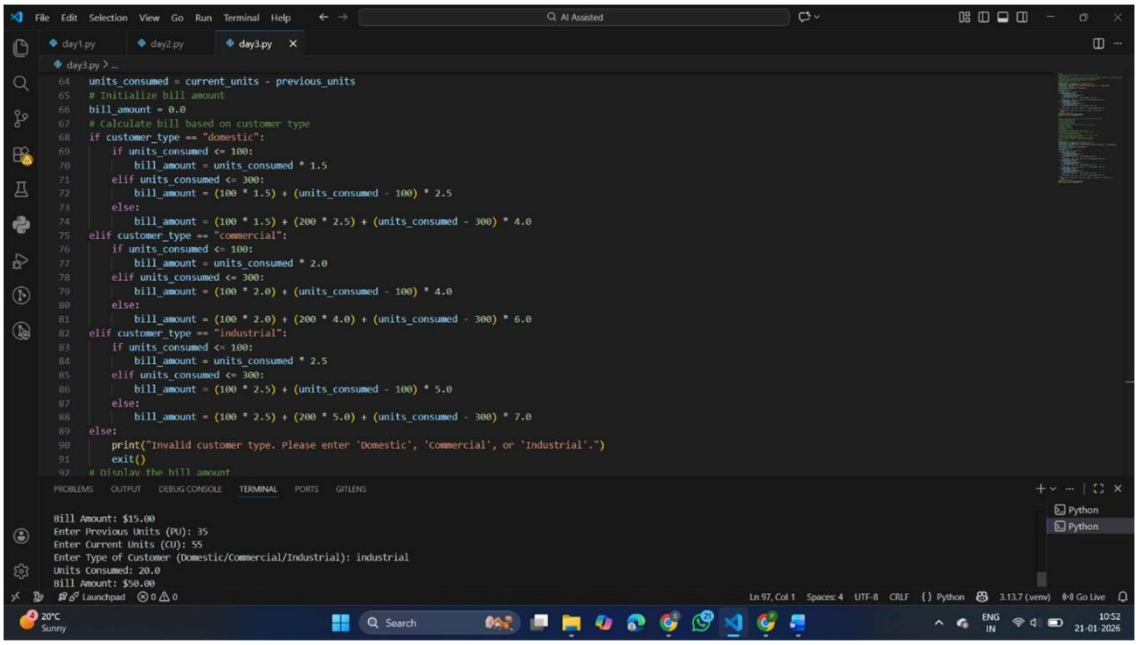
Create a Python program that calculates electricity charges depending on how many units a customer used and their category (Domestic, Commercial, Industrial).
Use conditional statements to apply different tariff rates.
Print the total bill clearly and explain how the calculation works.

CODE:

```
previous_units = float(input("Enter Previous Units (PU): "))
```

```
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer
(Domestic/Commercial/Industrial): ").strip().lower()
# Calculate units consumed
units_consumed = current_units - previous_units
# Initialize bill amount
bill_amount = 0.0
# Calculate bill based on customer type
if customer_type == "domestic":
  if units_consumed <= 100:
    bill_amount = units_consumed * 1.5
  elif units_consumed <= 300:
    bill_amount = (100 * 1.5) + (units_consumed - 100) * 2.5
  else:
    bill_amount = (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
elif customer_type == "commercial":
  if units_consumed <= 100:
    bill_amount = units_consumed * 2.0
  elif units_consumed <= 300:
    bill_amount = (100 * 2.0) + (units_consumed - 100) * 4.0
  else:
    bill_amount = (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
elif customer_type == "industrial":
  if units_consumed <= 100:
    bill_amount = units_consumed * 2.5
  elif units_consumed <= 300:
    bill_amount = (100 * 2.5) + (units_consumed - 100) * 5.0
  else:
    bill_amount = (100 * 2.5) + (200 * 5.0) + (units_consumed - 300) * 7.0
else:
  print("Invalid customer type. Please enter 'Domestic', 'Commercial', or
'Industrial'.")
exit()
# Display the bill amount
print(f"Units Consumed: {units_consumed}")
print(f"Bill Amount: ${bill_amount:.2f}")
```

OUTPUT:



EXPLANATION:

In this task, energy charges are calculated based on units consumed and consumer type.

Conditional logic is used to apply different tariff rates for domestic, commercial, and industrial users.

❖ TASK-3:

PROMPT:

Write a Python electricity billing program using functions.

Create separate functions to calculate Energy Charges and Fixed Charges based on customer type.

Take inputs, compute units consumed, and display all charges with proper labels.

Add comments, sample input/output, and at least three test cases.

CODE:

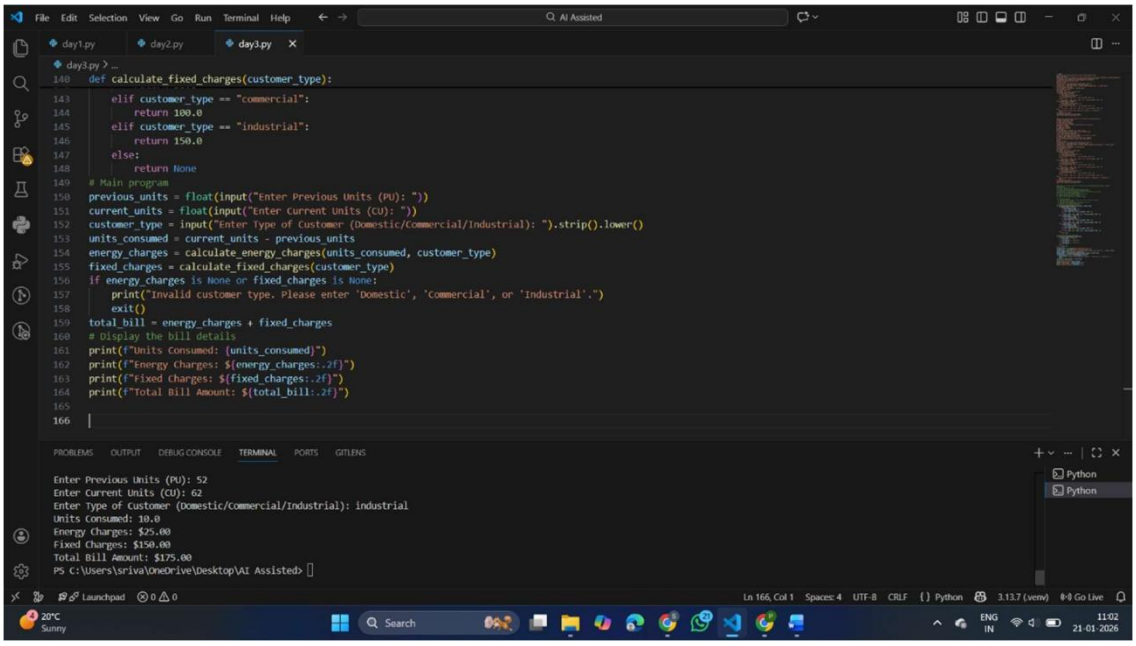
```
def calculate_energy_charges(units_consumed, customer_type):
  if customer_type == "domestic":
    if units_consumed <= 100:
      return units_consumed * 1.5
    elif units_consumed <= 300:
      return (100 * 1.5) + (units_consumed - 100) * 2.5
    else:
      return (100 * 1.5) + (200 * 2.5) + (units_consumed - 300) * 4.0
  elif customer_type == "commercial":
    if units_consumed <= 100:
      return units_consumed * 2.0
    elif units_consumed <= 300:
```

```
        return (100 * 2.0) + (units_consumed - 100) * 4.0
      else:
        return (100 * 2.0) + (200 * 4.0) + (units_consumed - 300) * 6.0
    elif customer_type == "industrial":
      if units_consumed <= 100:
        return units_consumed * 2.5
      elif units_consumed <= 300:
        return (100 * 2.5) + (units_consumed - 100) * 5.0
      else:
        return (100 * 2.5) + (200 * 5.0) + (units_consumed - 300) * 7.0
    else:
      return None

# Function to calculate Fixed Charges
def calculate_fixed_charges(customer_type):
    if customer_type == "domestic":
      return 50.0
    elif customer_type == "commercial":
      return 100.0
    elif customer_type == "industrial":
      return 150.0
    else:
      return None
# Main program
previous_units = float(input("Enter Previous Units (PU): "))
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer
(Domestic/Commercial/Industrial): ").strip().lower()
```

```
units_consumed = current_units - previous_units
energy_charges = calculate_energy_charges(units_consumed,
customer_type)
fixed_charges = calculate_fixed_charges(customer_type)
if energy_charges is None or fixed_charges is None:
  print("Invalid customer type. Please enter 'Domestic', 'Commercial', or
'Industrial'.")
  exit()
total_bill = energy_charges + fixed_charges
# Display the bill details
print(f"Units Consumed: {units_consumed}")
print(f"Energy Charges: ${energy_charges:.2f}")
print(f"Fixed Charges: ${fixed_charges:.2f}")
print(f"Total Bill Amount: ${total_bill:.2f}")
```

OUTPUT:



EXPLANATION:

introduces modular programming using user-defined methods.
Separate methods are used to calculate energy charges and fixed charges.
This makes the program reusable and easier to maintain.
Modular design improves code structure and readability.

❖ TASK-4:

PROMPT:

Extend the electricity billing program by adding more charges like Fixed Charges and Electricity Duty.
Electricity Duty should be calculated as a percentage of Energy Charges.
Display each charge separately and make the output clear and easy to read.
Ensure all calculations are correct.

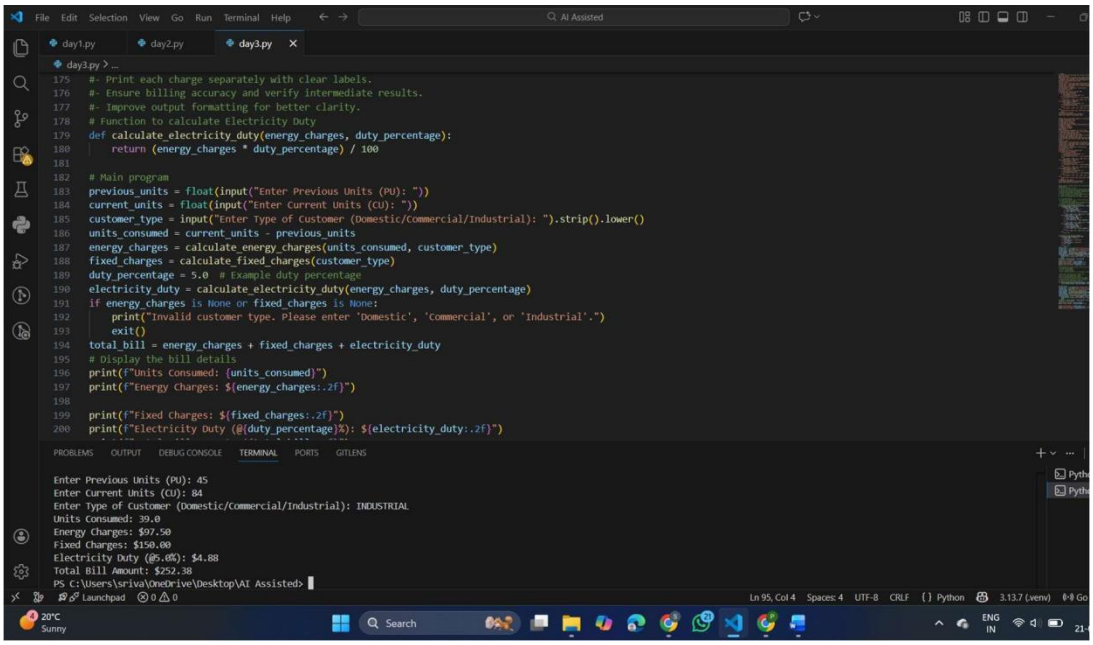
CODE:

```
def calculate_electricity_duty(energy_charges, duty_percentage):
  return (energy_charges * duty_percentage) / 100

# Main program
previous_units = float(input("Enter Previous Units (PU): "))
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer
(Domestic/Commercial/Industrial): ").strip().lower()
units_consumed = current_units - previous_units
energy_charges = calculate_energy_charges(units_consumed,
customer_type)
fixed_charges = calculate_fixed_charges(customer_type)
duty_percentage = 5.0  # Example duty percentage
electricity_duty = calculate_electricity_duty(energy_charges,
duty_percentage)
if energy_charges is None or fixed_charges is None:
  print("Invalid customer type. Please enter 'Domestic', 'Commercial',
or 'Industrial'.")
  exit()
total_bill = energy_charges + fixed_charges + electricity_duty
# Display the bill details
print(f"Units Consumed: {units_consumed}")
print(f"Energy Charges: ${energy_charges:.2f}")

print(f"Fixed Charges: ${fixed_charges:.2f}")
print(f"Electricity Duty (@{duty_percentage}%):
${electricity_duty:.2f}")
print(f"Total Bill Amount: ${total_bill:.2f}")
```

OUTPUT:



EXPLANATION:

This task extends billing by adding additional charges like fixed charges, customer charges, and electricity duty.

Electricity duty is calculated as a percentage of energy charges.

Printing individual charges helps verify calculation accuracy.

This step makes the bill more realistic and detailed.

❖ TASK-5:

PROMPT:

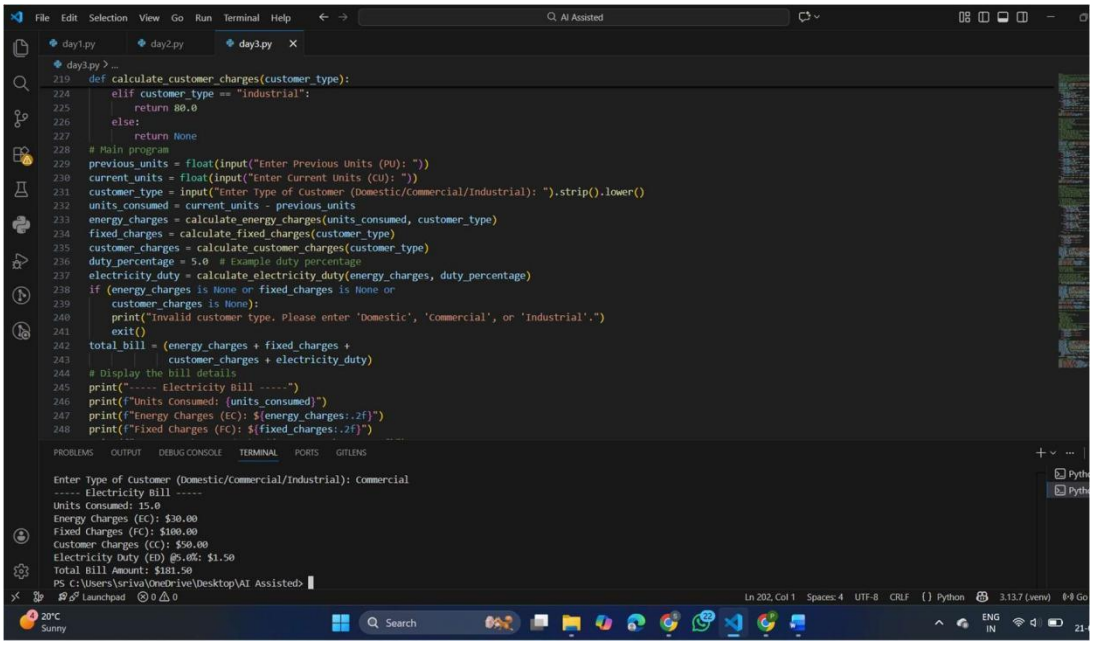
Create a complete electricity billing system in Python.
Calculate Energy Charges (EC), Fixed Charges (FC), Customer Charges (CC), and Electricity Duty (ED).
Find the total bill using all these values.
Display the final result in a neat format that looks like a real electricity bill.

CODE:

```
def calculate_customer_charges(customer_type):
  if customer_type == "domestic":
    return 20.0
```

```
  elif customer_type == "commercial":
    return 50.0
  elif customer_type == "industrial":
    return 80.0
  else:
    return None
# Main program
previous_units = float(input("Enter Previous Units (PU): "))
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer
(Domestic/Commercial/Industrial): ").strip().lower()
units_consumed = current_units - previous_units
energy_charges = calculate_energy_charges(units_consumed,
customer_type)
fixed_charges = calculate_fixed_charges(customer_type)
customer_charges = calculate_customer_charges(customer_type)
duty_percentage = 5.0  # Example duty percentage
electricity_duty = calculate_electricity_duty(energy_charges,
duty_percentage)
if (energy_charges is None or fixed_charges is None or
  customer_charges is None):
  print("Invalid customer type. Please enter 'Domestic', 'Commercial',
or 'Industrial'.")
  exit()
total_bill = (energy_charges + fixed_charges +
        customer_charges + electricity_duty)
# Display the bill details
print("----- Electricity Bill ---")
print(f"Units Consumed: {units_consumed}")
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) @{duty_percentage}%:
${electricity_duty:.2f}")
print(f"Total Bill Amount: ${total_bill:.2f}")
```

OUTPUT:



EXPLANATION:

Task 5 generates the final electricity bill by combining all charge components.

The total bill amount is calculated by adding EC, FC, CC, and ED.

The output is displayed in a neat, bill-like format for clarity.

This task demonstrates a complete, real-world electricity billing application