ASSIGNMENT-3.3

2303A51343

BATCH-10

TASK-1:

PROMPT:

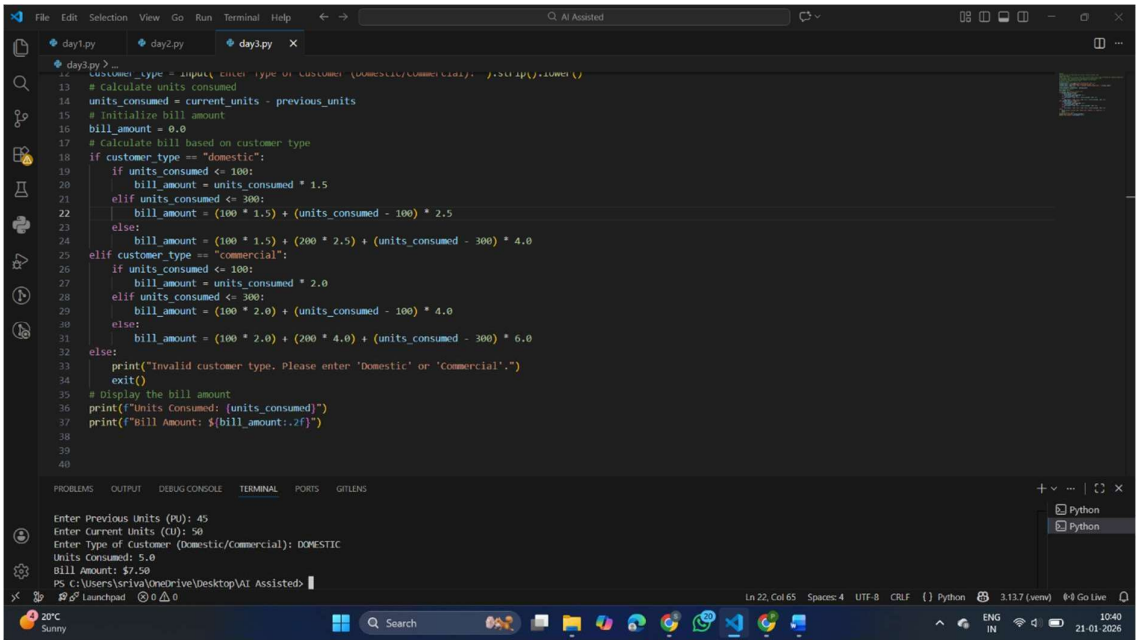
Generate a simple Python program that reads Previous Units (PU), Current Units (CU), and Type of Customer from the user.
Calculate the units consumed as CU − PU.
Implement all logic directly in the main program without using functions.
Display the units consumed clearly with proper print statements.

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:

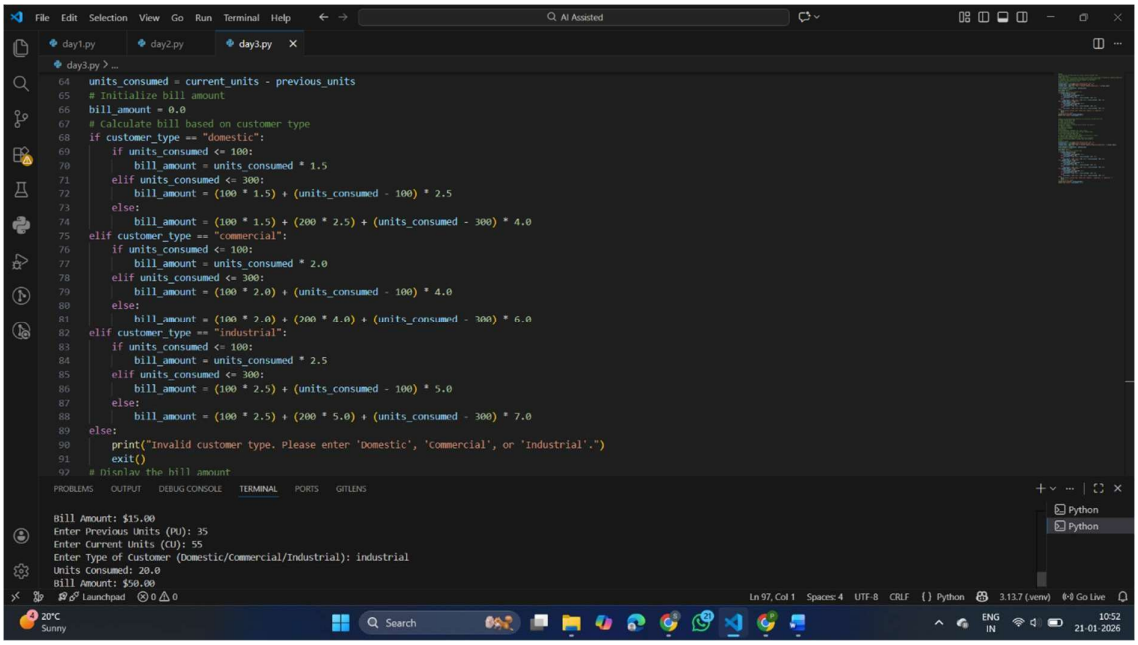
Write a Python program for an electricity billing system that reads the previous meter reading, current meter reading, and the type of customer, and then calculates the total units consumed. Extend the program to calculate the Energy Charges (EC) based on the category of customer, namely domestic, commercial, and industrial. The program should use meaningful variable names, display output using clean and clear print statements, and properly handle customer type input to ensure correct bill

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:

Generate a Python program for an electricity billing system using a modular design.

Requirements:

Create a function to calculate Energy Charges (EC):

- Accept units consumed and customer type as parameters.
- Use conditional logic for Domestic, Commercial, and Industrial customers.
- Return the calculated Energy Charges.
- Create a function to calculate Fixed Charges (FC):
  - Accept customer type as a parameter.
- Return fixed charges based on customer category.
- In the main program:
  - Read Previous Units (PU), Current Units (CU), and Type of Customer.
  - Calculate units consumed.
  - Call the Energy Charges and Fixed Charges functions.
  - Display all calculated values clearly.

Function to calculate Energy Charges

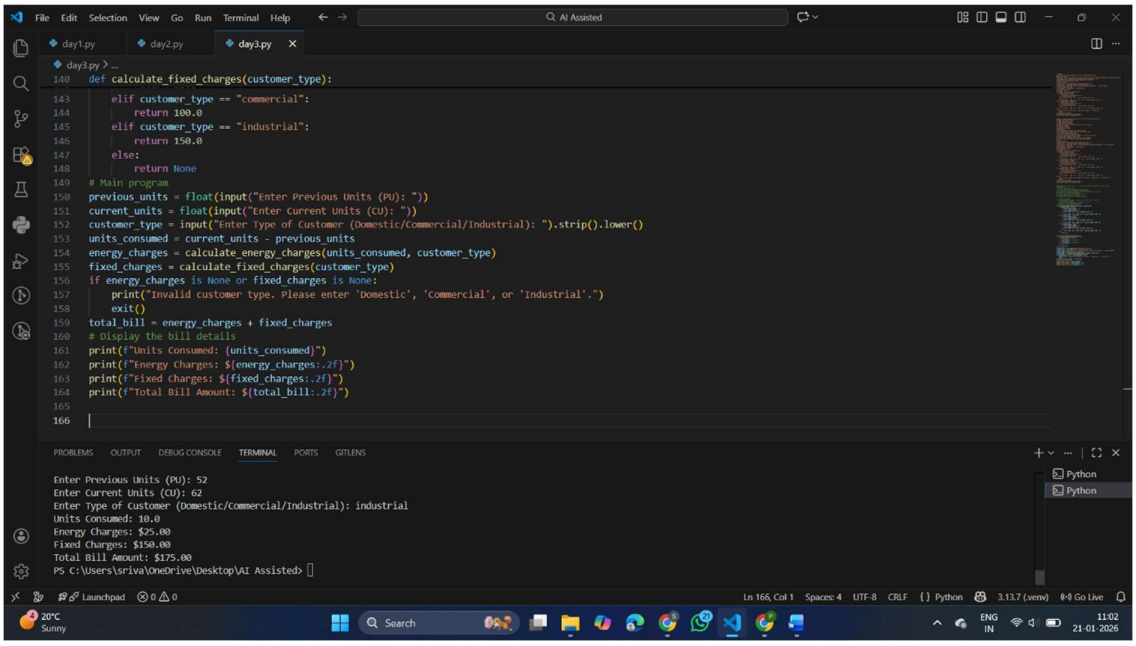
CODE:

```
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 existing Python electricity billing program to include additional charges.

Calculate and display:

- FC – Fixed Charges
- CC – Customer Charges
- ED – Electricity Duty (as a percentage of Energy Charges) Print each charge separately with clear labels.

Ensure billing accuracy and verify intermediate results. Improve output formatting for better clarity.

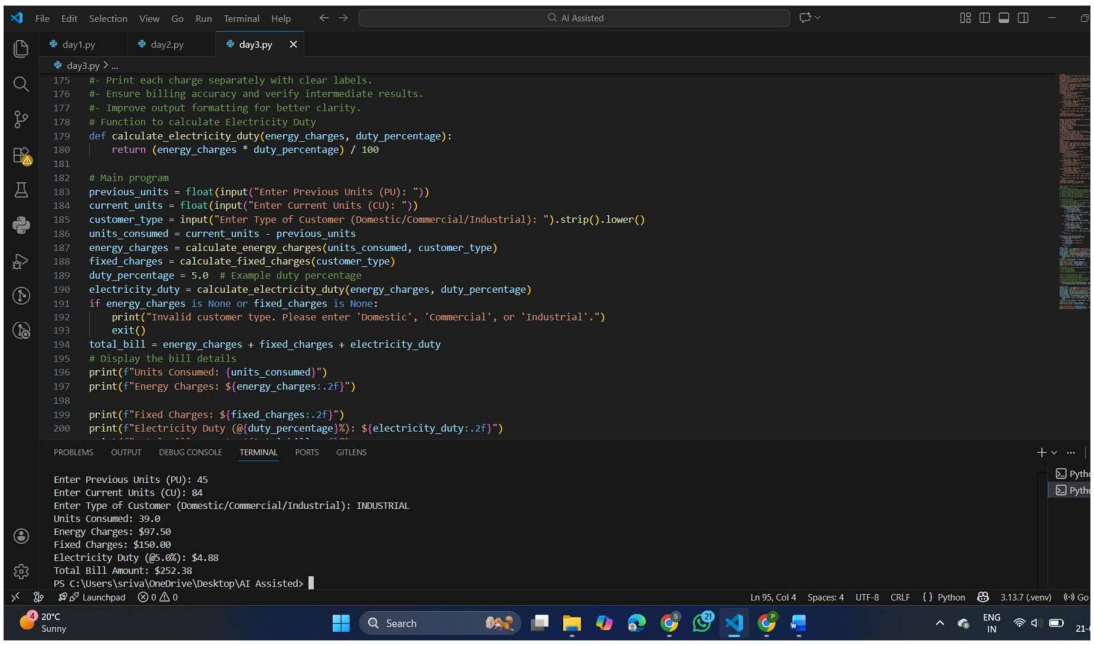
CODE:

```
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 the final Python electricity billing program. #Requirements:

Calculate Total Bill using:

Total Bill = EC + FC + CC + ED #Display

clearly:

#Energy Charges (EC) #Fixed

Charges (FC) #Customer

Charges (CC) #Electricity Duty

(ED) #Total Bill Amount

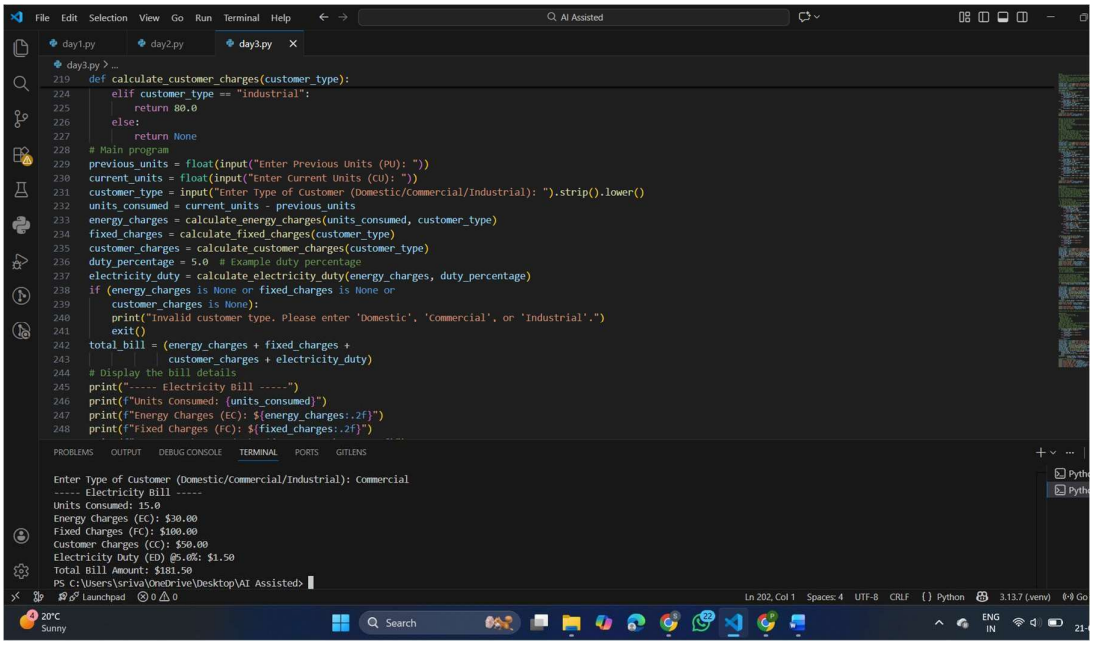
#Format the output neatly like a real electricity bill.

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