Name: K.Sanjana

Roll.no:2303a52139

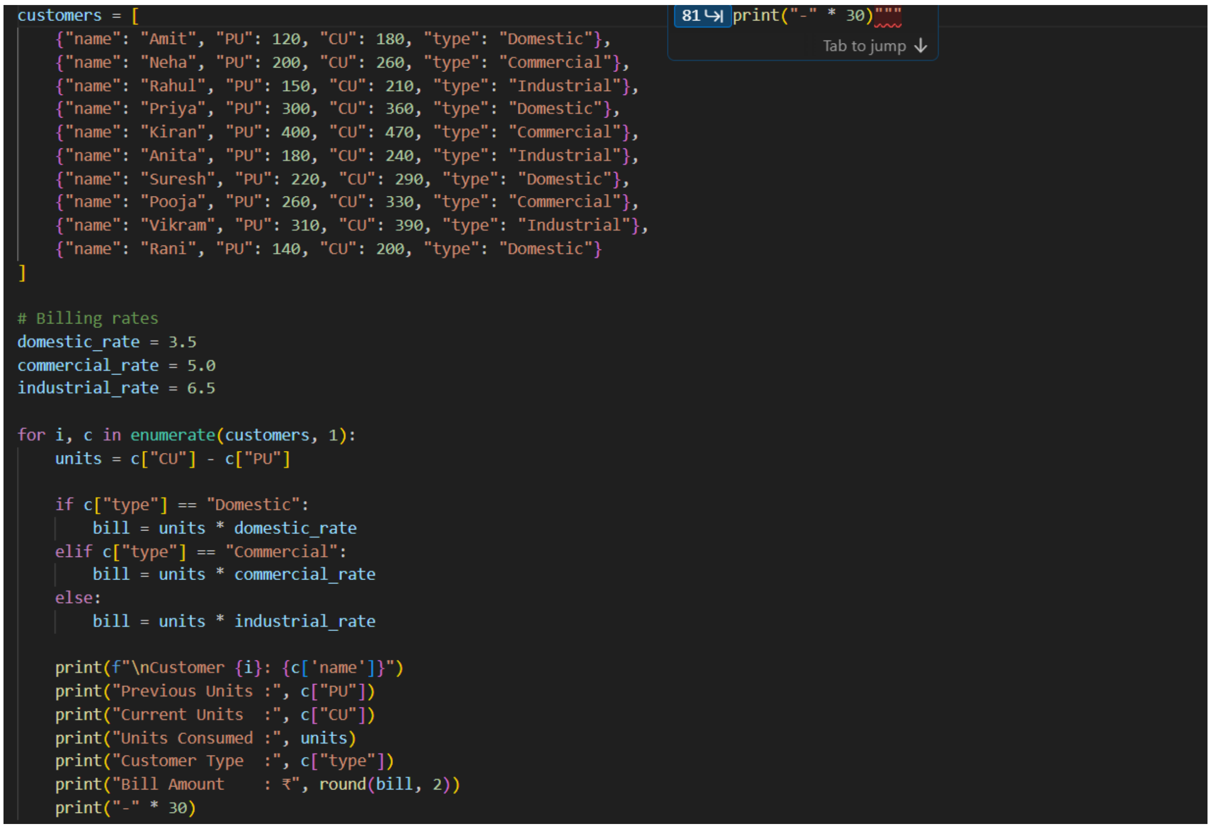
Batch.no:41

TASK1:

Prompt:

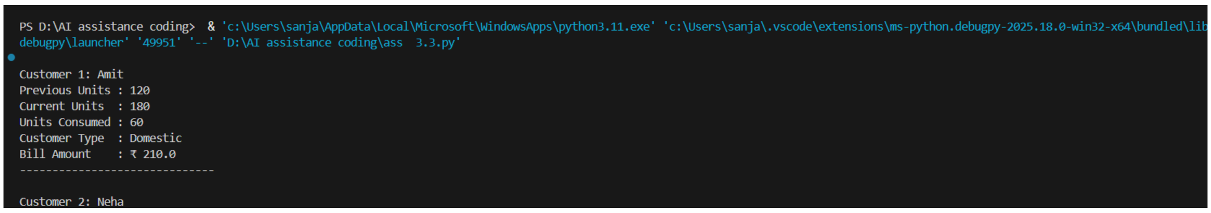
Generate a Python program (no functions, only main code).Create details for 10 customers in the code itself with:Previous Units (PU),Current Units (CU),Customer Type (Domestic / Commercial / Industrial)Calculate:Units Consumed = CU – PU.Bill Amount based on customer type rates.Use a loop to process all 10 customers and display the output in a table format showing:PU, CU, Units Consumed, Customer Type, Bill Amount

CODE:



```python
customers = [
    {"name": "Amit", "PU": 120, "CU": 180, "type": "Domestic"},
    {"name": "Neha", "PU": 200, "CU": 260, "type": "Commercial"},
    {"name": "Rahul", "PU": 150, "CU": 210, "type": "Industrial"},
    {"name": "Priya", "PU": 300, "CU": 360, "type": "Domestic"},
    {"name": "Kiran", "PU": 400, "CU": 470, "type": "Commercial"},
    {"name": "Anita", "PU": 180, "CU": 240, "type": "Industrial"},
    {"name": "Suresh", "PU": 220, "CU": 290, "type": "Domestic"},
    {"name": "Pooja", "PU": 260, "CU": 330, "type": "Commercial"},
    {"name": "Vikram", "PU": 310, "CU": 390, "type": "Industrial"},
    {"name": "Rani", "PU": 140, "CU": 200, "type": "Domestic"}
]

# Billing rates
domestic_rate = 3.5
commercial_rate = 5.0
industrial_rate = 6.5

for i, c in enumerate(customers, 1):
    units = c["CU"] - c["PU"]

    if c["type"] == "Domestic":
        bill = units * domestic_rate
    elif c["type"] == "Commercial":
        bill = units * commercial_rate
    else:
        bill = units * industrial_rate

    print(f"\nCustomer {i}: {c['name']}")
    print("Previous Units :", c["PU"])
    print("Current Units  :", c["CU"])
    print("Units Consumed :", units)
    print("Customer Type  :", c["type"])
    print("Bill Amount    : ₹", round(bill, 2))
    print("-" * 30)
```

SAMPLE OUTPUT:



```
PS D:\AI assistance coding>  & 'c:\Users\sanja\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\sanja\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\lib
debugpy\launcher' '49951' '--' 'D:\AI assistance coding\ass  3.3.py'

Customer 1: Amit
Previous Units : 120
Current Units  : 180
Units Consumed : 60
Customer Type  : Domestic
Bill Amount    : ₹ 210.0
------------------------------

Customer 2: Neha
```

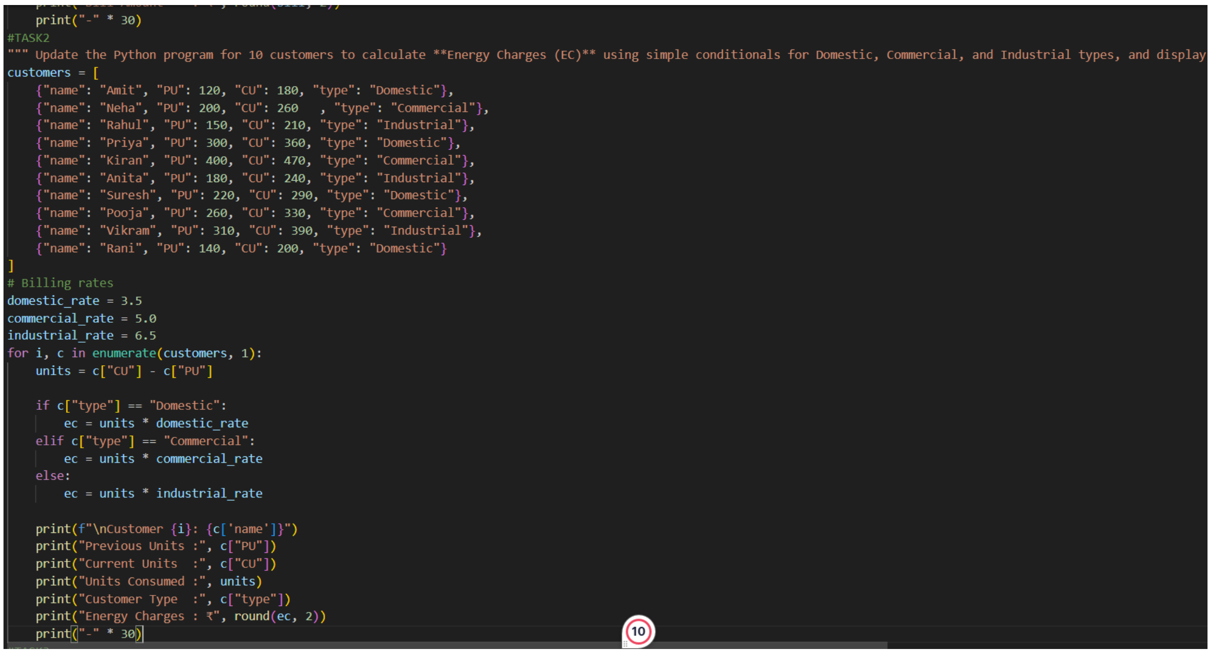
JUSTIFICATION:

This task establishes the foundation of the electricity billing system by collecting customer data such as Previous Units, Current Units, and Customer Type. Calculating the units consumed (CU – PU) ensures accurate measurement of energy usage. Implementing the logic directly in the main program helps beginners understand basic input handling and arithmetic operations. This task validates that the program can correctly process customer information before applying any billing logic.

TASK2:

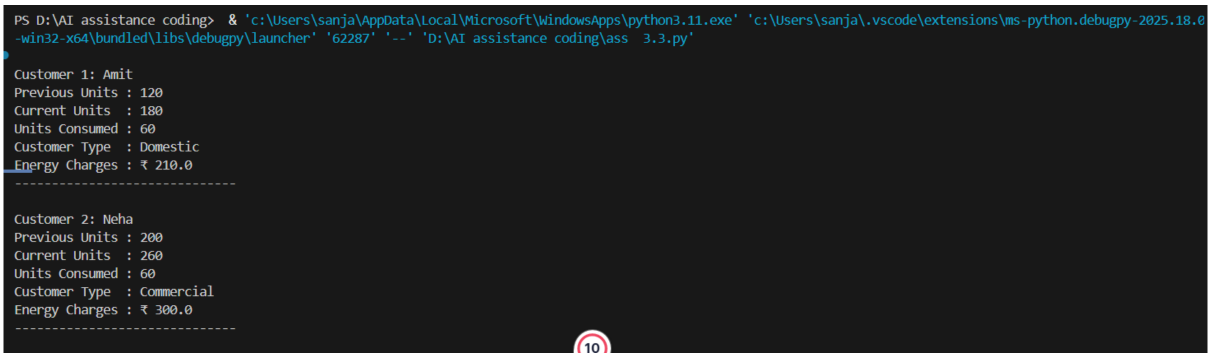
Prompt: Update the Python program for 10 customers to calculate **Energy Charges (EC)** using simple conditionals for Domestic, Commercial, and Industrial types, and display PU, CU, Units, Type, and EC line-wise with sample output.

CODE:



```
    print("-" * 30)
#TASK2
""" Update the Python program for 10 customers to calculate **Energy Charges (EC)** using simple conditionals for Domestic, Commercial, and Industrial types, and display
customers = [
    {"name": "Amit", "PU": 120, "CU": 180, "type": "Domestic"},
    {"name": "Neha", "PU": 200, "CU": 260   , "type": "Commercial"},
    {"name": "Rahul", "PU": 150, "CU": 210, "type": "Industrial"},
    {"name": "Priya", "PU": 300, "CU": 360, "type": "Domestic"},
    {"name": "Kiran", "PU": 400, "CU": 470, "type": "Commercial"},
    {"name": "Anita", "PU": 180, "CU": 240, "type": "Industrial"},
    {"name": "Suresh", "PU": 220, "CU": 290, "type": "Domestic"},
    {"name": "Pooja", "PU": 260, "CU": 330, "type": "Commercial"},
    {"name": "Vikram", "PU": 310, "CU": 390, "type": "Industrial"},
    {"name": "Rani", "PU": 140, "CU": 200, "type": "Domestic"}
]
# Billing rates
domestic_rate = 3.5
commercial_rate = 5.0
industrial_rate = 6.5
for i, c in enumerate(customers, 1):
    units = c["CU"] - c["PU"]

    if c["type"] == "Domestic":
        ec = units * domestic_rate
    elif c["type"] == "Commercial":
        ec = units * commercial_rate
    else:
        ec = units * industrial_rate

    print(f"\nCustomer {i}: {c['name']}")
    print("Previous Units :", c["PU"])
    print("Current Units  :", c["CU"])
    print("Units Consumed :", units)
    print("Customer Type  :", c["type"])
    print("Energy Charges : ₹", round(ec, 2))
    print("-" * 30)
```

OUTPUT:



```
PS D:\AI assistance coding>  & 'c:\Users\sanja\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\sanja\.vscode\extensions\ms-python.debugpy-2025.18.0
-win32-x64\bundled\libs\debugpy\launcher' '62287' '--' 'D:\AI assistance coding\ass  3.3.py'

Customer 1: Amit
Previous Units : 120
Current Units  : 180
Units Consumed : 60
Customer Type  : Domestic
Energy Charges : ₹ 210.0
------------------------------

Customer 2: Neha
Previous Units : 200
Current Units  : 260
Units Consumed : 60
Customer Type  : Commercial
Energy Charges : ₹ 300.0
------------------------------
```

JUSTIFICATION:

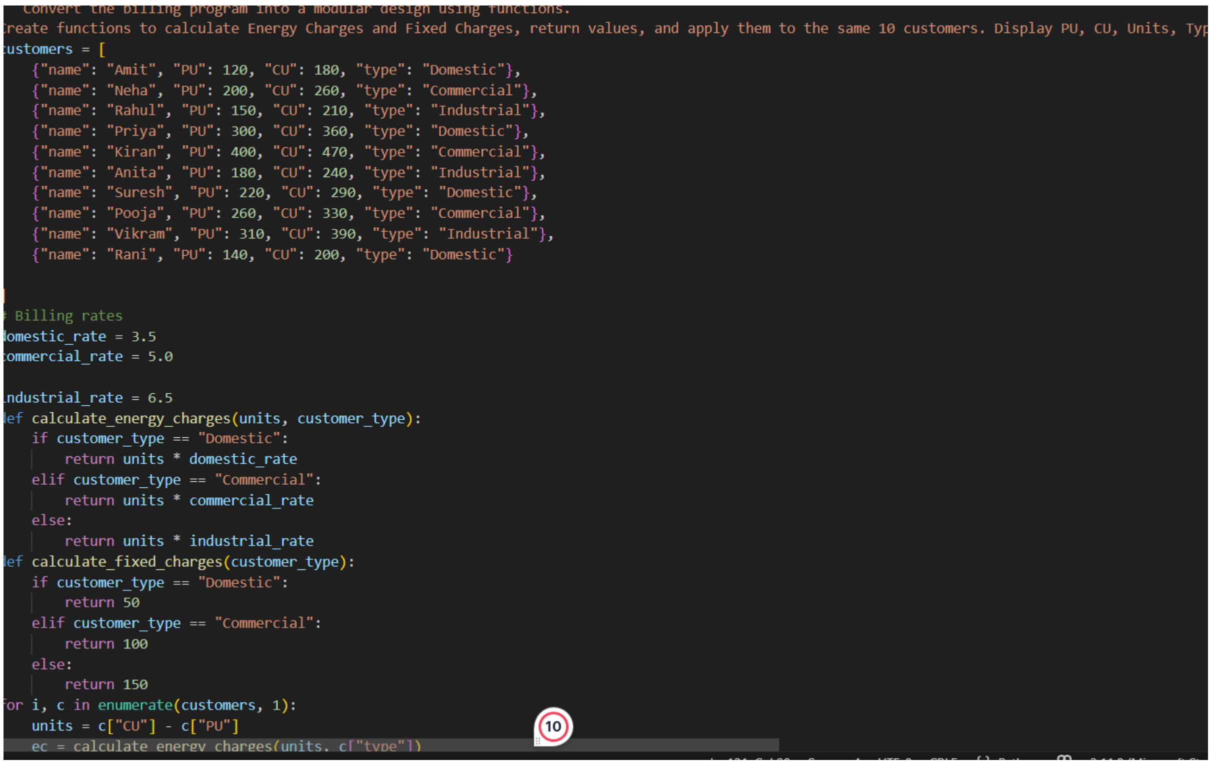
Task 2 extends the program by introducing Energy Charges based on the type of consumer (Domestic, Commercial, Industrial). Using conditional statements ensures that different billing rates are applied correctly for each category. This makes the program more realistic and applicable to real-world electricity billing systems, where tariffs vary by customer type. Improving readability and logic clarity also enhances maintainability and reduces errors.

TASK3:

PROMPT: Convert the billing program into a modular design using functions.

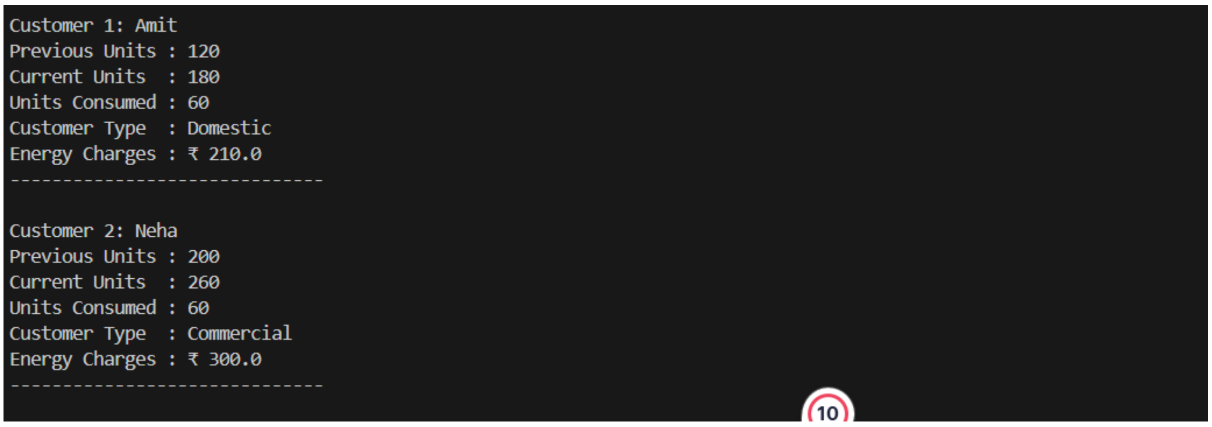
Create functions to calculate Energy Charges and Fixed Charges, return values, and apply them to the same 10 customers. Display PU, CU, Units, Type, EC, FC, and Total line-wise with comments and sample output.

CODE:



```
  Convert the billing program into a modular design using functions.
reate functions to calculate Energy Charges and Fixed Charges, return values, and apply them to the same 10 customers. Display PU, CU, Units, Typ
ustomers = [
    {"name": "Amit", "PU": 120, "CU": 180, "type": "Domestic"},
    {"name": "Neha", "PU": 200, "CU": 260, "type": "Commercial"},
    {"name": "Rahul", "PU": 150, "CU": 210, "type": "Industrial"},
    {"name": "Priya", "PU": 300, "CU": 360, "type": "Domestic"},
    {"name": "Kiran", "PU": 400, "CU": 470, "type": "Commercial"},
    {"name": "Anita", "PU": 180, "CU": 240, "type": "Industrial"},
    {"name": "Suresh", "PU": 220, "CU": 290, "type": "Domestic"},
    {"name": "Pooja", "PU": 260, "CU": 330, "type": "Commercial"},
    {"name": "Vikram", "PU": 310, "CU": 390, "type": "Industrial"},
    {"name": "Rani", "PU": 140, "CU": 200, "type": "Domestic"}


 Billing rates
omestic_rate = 3.5
ommercial_rate = 5.0

ndustrial_rate = 6.5
ef calculate_energy_charges(units, customer_type):
    if customer_type == "Domestic":
        return units * domestic_rate
    elif customer_type == "Commercial":
        return units * commercial_rate
    else:
        return units * industrial_rate
ef calculate_fixed_charges(customer_type):
    if customer_type == "Domestic":
        return 50
    elif customer_type == "Commercial":
        return 100
    else:
        return 150
or i, c in enumerate(customers, 1):
    units = c["CU"] - c["PU"]
    ec = calculate_energy_charges(units, c["type"])
```

OUTPUT:



```
Customer 1: Amit
Previous Units : 120
Current Units  : 180
Units Consumed : 60
Customer Type  : Domestic
Energy Charges : ₹ 210.0
------------------------------

Customer 2: Neha
Previous Units : 200
Current Units  : 260
Units Consumed : 60
Customer Type  : Commercial
Energy Charges : ₹ 300.0
------------------------------
```

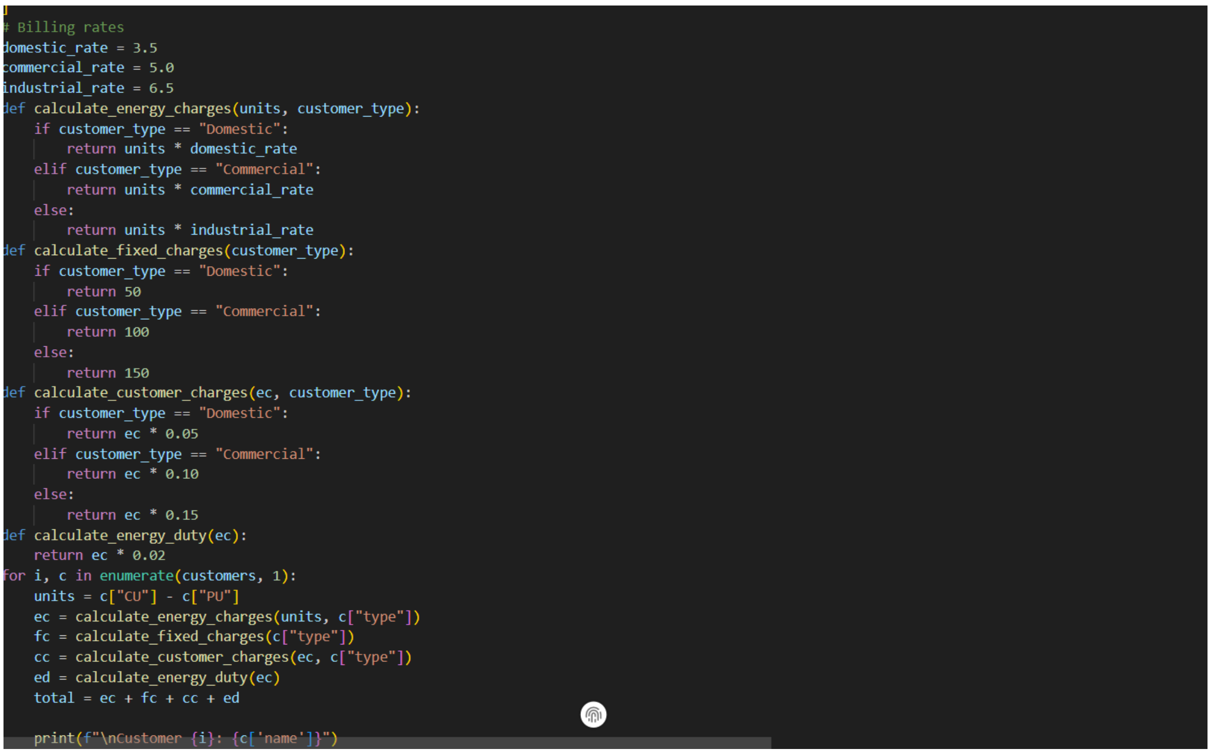
JUSTIFICATION:

This task focuses on modularity and reusability by introducing user-defined functions to calculate Energy Charges and Fixed Charges. Separating logic into functions improves code organization, readability, and scalability. It allows the same billing logic to be reused for multiple consumers without repetition. Modular design is important in real applications, as it simplifies debugging, updates, and future enhancements.

TASK4:

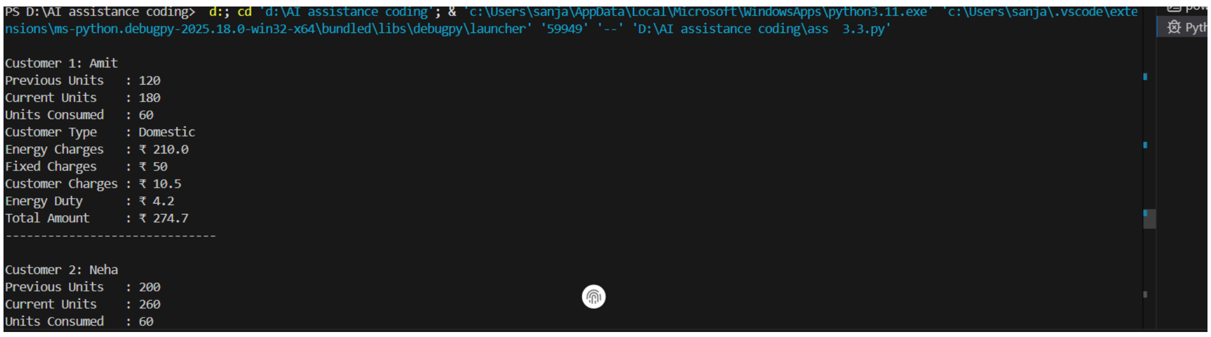
PROMPT: Update the billing program to compute FC, CC, and ED (percentage of EC) for the same customers, improve billing accuracy with clear functions, and print all charges line-wise

CODE:



```python
# Billing rates
domestic_rate = 3.5
commercial_rate = 5.0
industrial_rate = 6.5
def calculate_energy_charges(units, customer_type):
    if customer_type == "Domestic":
        return units * domestic_rate
    elif customer_type == "Commercial":
        return units * commercial_rate
    else:
        return units * industrial_rate
def calculate_fixed_charges(customer_type):
    if customer_type == "Domestic":
        return 50
    elif customer_type == "Commercial":
        return 100
    else:
        return 150
def calculate_customer_charges(ec, customer_type):
    if customer_type == "Domestic":
        return ec * 0.05
    elif customer_type == "Commercial":
        return ec * 0.10
    else:
        return ec * 0.15
def calculate_energy_duty(ec):
    return ec * 0.02
for i, c in enumerate(customers, 1):
    units = c["CU"] - c["PU"]
    ec = calculate_energy_charges(units, c["type"])
    fc = calculate_fixed_charges(c["type"])
    cc = calculate_customer_charges(ec, c["type"])
    ed = calculate_energy_duty(ec)
    total = ec + fc + cc + ed

    print(f"\nCustomer {i}: {c['name']}")
```

OUTPUT:



```
PS D:\AI assistance coding>  d:; cd 'd:\AI assistance coding'; & 'c:\Users\sanja\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\sanja\.vscode\exte
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59949' '--' 'D:\AI assistance coding\ass  3.3.py'

Customer 1: Amit
Previous Units    : 120
Current Units     : 180
Units Consumed    : 60
Customer Type     : Domestic
Energy Charges    : ₹ 210.0
Fixed Charges     : ₹ 50
Customer Charges  : ₹ 10.5
Energy Duty       : ₹ 4.2
Total Amount      : ₹ 274.7
------------------------------

Customer 2: Neha
Previous Units    : 200
Current Units     : 260
Units Consumed    : 60
```

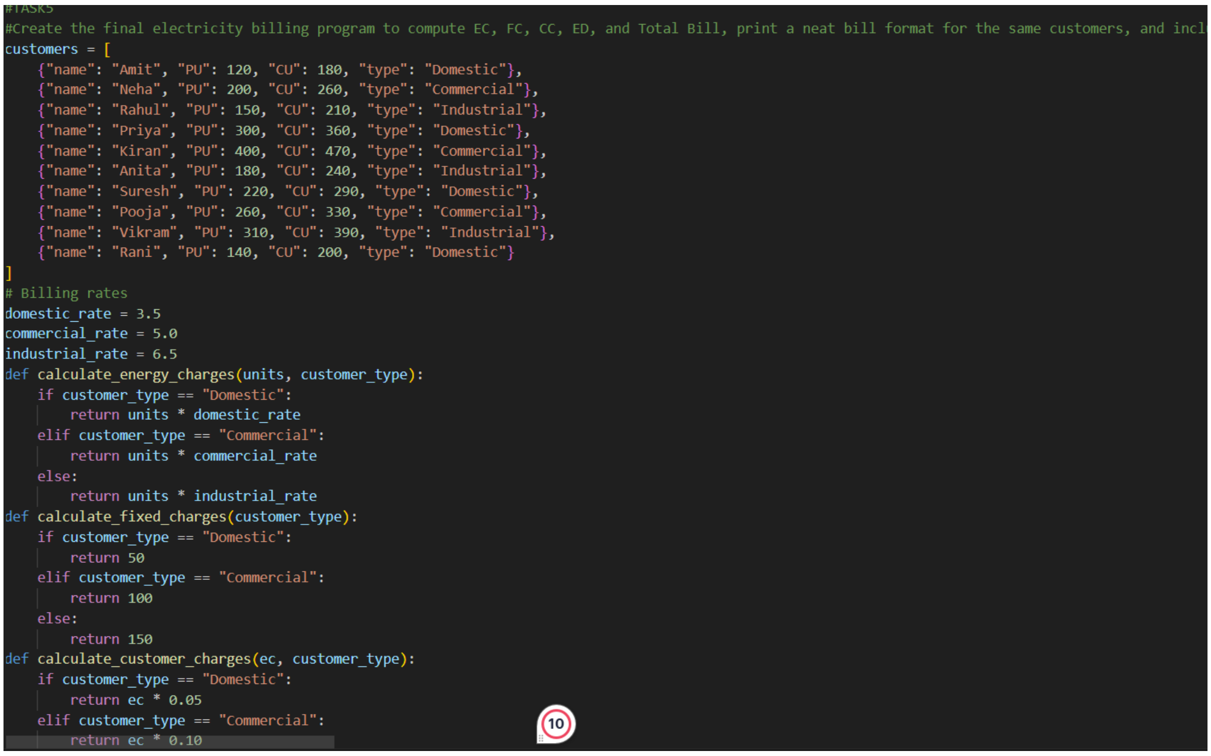
JUSTIFICATION:

In Task 4, the program is enhanced to include additional billing components such as Fixed Charges (FC), Customer Charges (CC), and Electricity Duty (ED). Calculating ED as a percentage of Energy Charges ensures regulatory compliance and realistic billing. Printing individual charge values improves transparency and verification. This task increases billing accuracy and reflects how actual utility bills include multiple components beyond basic energy usage.

TASK5:

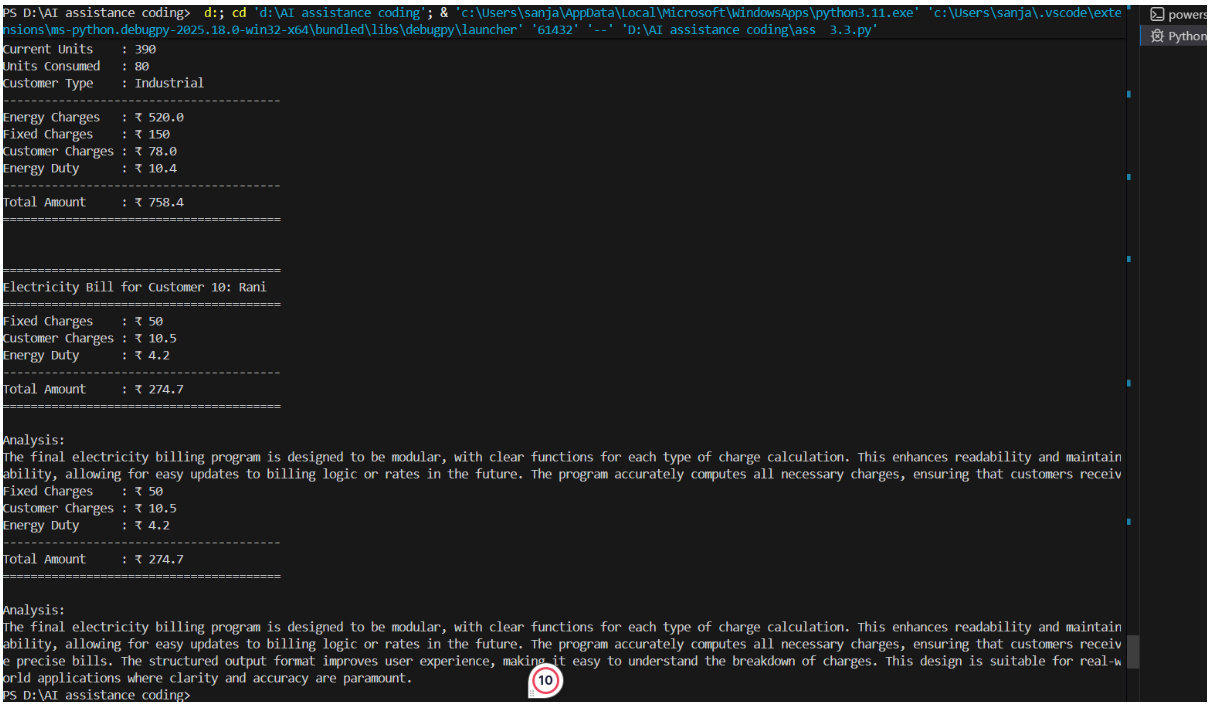
PROMPT: Create the final electricity billing program to compute EC, FC, CC, ED, and Total Bill, print a neat bill format for the same customers, and include sample output with a brief analysis on accuracy, readability, and real-world use.

CODE:



```python
#TASK5
#Create the final electricity billing program to compute EC, FC, CC, ED, and Total Bill, print a neat bill format for the same customers, and incl
customers = [
    {"name": "Amit", "PU": 120, "CU": 180, "type": "Domestic"},
    {"name": "Neha", "PU": 200, "CU": 260, "type": "Commercial"},
    {"name": "Rahul", "PU": 150, "CU": 210, "type": "Industrial"},
    {"name": "Priya", "PU": 300, "CU": 360, "type": "Domestic"},
    {"name": "Kiran", "PU": 400, "CU": 470, "type": "Commercial"},
    {"name": "Anita", "PU": 180, "CU": 240, "type": "Industrial"},
    {"name": "Suresh", "PU": 220, "CU": 290, "type": "Domestic"},
    {"name": "Pooja", "PU": 260, "CU": 330, "type": "Commercial"},
    {"name": "Vikram", "PU": 310, "CU": 390, "type": "Industrial"},
    {"name": "Rani", "PU": 140, "CU": 200, "type": "Domestic"}
]
# Billing rates
domestic_rate = 3.5
commercial_rate = 5.0
industrial_rate = 6.5
def calculate_energy_charges(units, customer_type):
    if customer_type == "Domestic":
        return units * domestic_rate
    elif customer_type == "Commercial":
        return units * commercial_rate
    else:
        return units * industrial_rate
def calculate_fixed_charges(customer_type):
    if customer_type == "Domestic":
        return 50
    elif customer_type == "Commercial":
        return 100
    else:
        return 150
def calculate_customer_charges(ec, customer_type):
    if customer_type == "Domestic":
        return ec * 0.05
    elif customer_type == "Commercial":
        return ec * 0.10
```

OUTPUT:



```
PS D:\AI assistance coding>  d:; cd 'd:\AI assistance coding'; & 'c:\Users\sanja\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\sanja\.vscode\exte
nsions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61432' '--' 'D:\AI assistance coding\ass  3.3.py'
Current Units    : 390
Units Consumed   : 80
Customer Type    : Industrial
----------------------------------------
Energy Charges   : ₹ 520.0
Fixed Charges    : ₹ 150
Customer Charges : ₹ 78.0
Energy Duty      : ₹ 10.4
----------------------------------------
Total Amount     : ₹ 758.4
========================================



========================================
Electricity Bill for Customer 10: Rani
========================================
Fixed Charges    : ₹ 50
Customer Charges : ₹ 10.5
Energy Duty      : ₹ 4.2
----------------------------------------
Total Amount     : ₹ 274.7
========================================

Analysis:
The final electricity billing program is designed to be modular, with clear functions for each type of charge calculation. This enhances readability and maintain
ability, allowing for easy updates to billing logic or rates in the future. The program accurately computes all necessary charges, ensuring that customers receiv
Fixed Charges    : ₹ 50
Customer Charges : ₹ 10.5
Energy Duty      : ₹ 4.2
----------------------------------------
Total Amount     : ₹ 274.7
========================================

Analysis:
The final electricity billing program is designed to be modular, with clear functions for each type of charge calculation. This enhances readability and maintain
ability, allowing for easy updates to billing logic or rates in the future. The program accurately computes all necessary charges, ensuring that customers receiv
e precise bills. The structured output format improves user experience, making it easy to understand the breakdown of charges. This design is suitable for real-w
orld applications where clarity and accuracy are paramount.
PS D:\AI assistance coding>
```

JUSTIFICATION:

The final task integrates all previous calculations to generate the complete electricity bill using the formula:
Total Bill = EC + FC + CC + ED.
It presents all charges clearly in a neat format, making the output user-friendly and professional. The analysis of accuracy, readability, and real-world applicability ensures the program is not only correct but also practical and maintainable. This task demonstrates how a simple program can evolve into a complete billing application suitable for real-life usage.