

Assignment 3.3

AIAC

Name: D.Shruthi

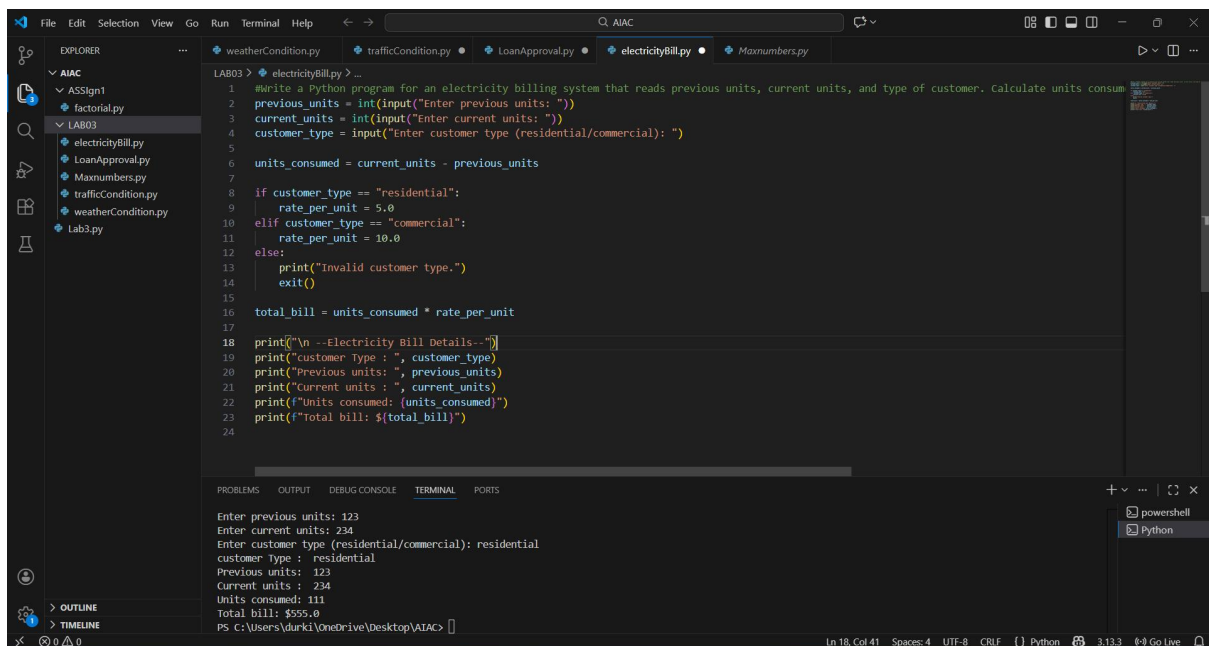
HT No: 2303A52407

Batch: 31

Lab 3: Application for TGNPDCL - Electricity Bill Generation Using Python & AI Tools

Task 1:

Prompt: #Write a Python program for an electricity billing system that reads previous units, current units, and type of customer. Calculate units consumed and display the bill details clearly without using functions



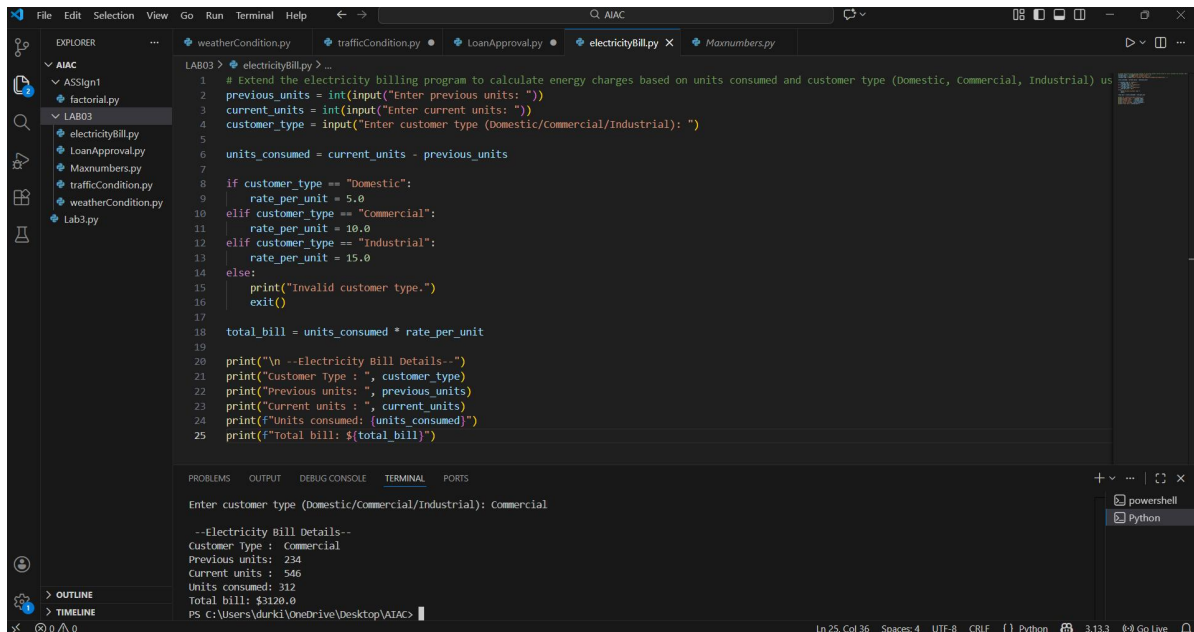
```
LAB03 > electricityBill.py > ...
1 #Write a Python program for an electricity billing system that reads previous units, current units, and type of customer. Calculate units consumed.
2 previous_units = int(input("Enter previous units: "))
3 current_units = int(input("Enter current units: "))
4 customer_type = input("Enter customer type (residential/commercial): ")
5
6 units_consumed = current_units - previous_units
7
8 if customer_type == "residential":
9     rate_per_unit = 5.0
10 elif customer_type == "commercial":
11     rate_per_unit = 10.0
12 else:
13     print("Invalid customer type.")
14     exit()
15
16 total_bill = units_consumed * rate_per_unit
17
18 print("\n --Electricity Bill Details--\n")
19 print("customer Type : ", customer_type)
20 print("Previous units : ", previous_units)
21 print("Current units : ", current_units)
22 print(f"Units consumed: {units_consumed}")
23 print(f"Total bill: ${total_bill}")
24
```

Enter previous units: 123
Enter current units: 234
Enter customer type (residential/commercial): residential
customer Type : residential
Previous units : 123
Current units : 234
Units consumed: 111
Total bill: \$555.0
PS C:\Users\durki\OneDrive\Desktop\AIAC>

This task focuses on creating a simple Python program that reads previous units, current units, and customer type from the user. The program calculates units consumed by finding the difference between the meter readings. It then displays basic bill details clearly without using any user-defined functions.

Task 2:

Prompt: # Extend the electricity billing program to calculate energy charges based on units consumed and customer type (Domestic, Commercial, Industrial) using simple and optimized conditional statements



```
1 # Extend the electricity billing program to calculate energy charges based on units consumed and customer type (Domestic, Commercial, Industrial) us
2 previous_units = int(input("Enter previous units: "))
3 current_units = int(input("Enter current units: "))
4 customer_type = input("Enter customer type (Domestic/Commercial/Industrial): ")
5
6 units_consumed = current_units - previous_units
7
8 if customer_type == "Domestic":
9     rate_per_unit = 5.0
10 elif customer_type == "Commercial":
11     rate_per_unit = 10.0
12 elif customer_type == "Industrial":
13     rate_per_unit = 15.0
14 else:
15     print("Invalid customer type.")
16     exit()
17
18 total_bill = units_consumed * rate_per_unit
19
20 print("\n --Electricity Bill Details--")
21 print("Customer Type : ", customer_type)
22 print("Previous units: ", previous_units)
23 print("Current units : ", current_units)
24 print(f"Units consumed: {units_consumed}")
25 print(f"Total bill: ${total_bill}")
```

Enter customer type (Domestic/Commercial/Industrial): Commercial

--Electricity Bill Details--
Customer Type : Commercial
Previous units: 234
Current units : 546
Units consumed: 312
Total bill: \$3120.0

In this task, the program is extended to calculate energy charges based on units consumed and customer type. Simple and optimized conditional statements are used to apply different rates for Domestic, Commercial, and Industrial consumers. This improves clarity while keeping the logic easy to understand.

Task3:

Prompt: # Generate a Python program using user-defined functions to calculate energy charges and fixed charges for different types of electricity consumers. The functions should return values and include clear comments

```

LAB03 > electricityBill.py
1 # Generate a Python program using user-defined functions to calculate energy charges
2 def calculate_energy_charge(consumer_type, units_consumed):
3     """
4     Calculate the energy charge based on consumer type and units consumed
5
6     Parameters:
7     consumer_type (str): Type of consumer ('residential', 'commercial', 'industrial')
8     units_consumed (float): Number of electricity units consumed
9
10    Returns:
11    float: Energy charge for the consumed units
12    """
13    if consumer_type == 'residential':
14        if units_consumed <= 100:
15            rate = 0.5
16        elif units_consumed <= 300:
17            rate = 0.75
18        else:
19            rate = 1.0
20    elif consumer_type == 'commercial':
21        if units_consumed <= 200:
22            rate = 1.0
23        elif units_consumed <= 500:
24            rate = 1.5
25        else:
26            rate = 2.0
27    elif consumer_type == 'industrial':
28        if units_consumed <= 500:
29            rate = 2.5
30        else:
31            rate = 3.0
32    return rate * units_consumed
33
34 # Example usage:
35 consumer_type = 'residential'
36 units_consumed = 350
37 energy_charge = calculate_energy_charge(consumer_type, units_consumed)
38 fixed_charge = calculate_fixed_charge(consumer_type)
39 total_bill = energy_charge + fixed_charge
40 print(f"Consumer Type: {consumer_type}")
41 print(f"Units Consumed: {units_consumed}")
42 print(f"Energy Charge: ${energy_charge:.2f}")
43 print(f"Fixed Charge: ${fixed_charge:.2f}")
44 print(f"Total Electricity Bill: ${total_bill:.2f}")

```

```

LAB03 > electricityBill.py
36 def calculate_fixed_charge(consumer_type):
37     """
38     Calculate the fixed charge based on consumer type.
39
40     Parameters:
41     consumer_type (str): Type of consumer ('residential', 'commercial', 'industrial')
42
43     Returns:
44     float: Fixed charge for the consumer type
45     """
46     if consumer_type == 'residential':
47         return 50.0
48     elif consumer_type == 'commercial':
49         return 100.0
50     elif consumer_type == 'industrial':
51         return 200.0
52     else:
53         raise ValueError("Invalid consumer type")
54
55 # Example usage:
56 consumer_type = 'residential'
57 units_consumed = 350
58 energy_charge = calculate_energy_charge(consumer_type, units_consumed)
59 fixed_charge = calculate_fixed_charge(consumer_type)
60 total_bill = energy_charge + fixed_charge
61 print(f"Consumer Type: {consumer_type}")
62 print(f"Units Consumed: {units_consumed}")
63 print(f"Energy Charge: ${energy_charge:.2f}")
64 print(f"Fixed Charge: ${fixed_charge:.2f}")
65 print(f"Total Electricity Bill: ${total_bill:.2f}")

```

```

ty0bill.py
Consumer Type: residential
Units Consumed: 350
Energy Charge: $350.00
Fixed Charge: $50.00
Total Electricity Bill: $400.00

```

This task introduces user-defined functions to calculate energy charges and fixed charges. Each function returns calculated values and includes proper comments for better readability. Using functions makes the program modular, reusable, and easier to maintain.

Task 4:

Prompt: # Extend the electricity billing program to calculate additional charges such as fixed charges, customer charges, and electricity duty. Display each charge separately to improve billing accuracy

```

LAB03 > electricityBill.py
1 # Extend the electricity billing program to calculate additional charges such as fixed charges, customer charges, and electricity duty. Display each charge separately
2 def calculate_electricity_bill(units_consumed):
3     """
4     Calculate the electricity bill based on units consumed
5
6     Parameters:
7     units_consumed (float): Number of electricity units consumed
8
9     Returns:
10    float: Total electricity bill amount
11    """
12    # Define the rates and charges
13    rate_per_unit = 0.15 # Rate per unit of electricity
14    fixed_charge = 5.00 # Fixed charge
15    customer_charge = 2.00 # Customer charge
16    electricity_duty_rate = 0.05 # 5% electricity duty
17
18    # Calculate the basic bill amount
19    basic_bill = units_consumed * rate_per_unit
20
21    # Calculate electricity duty
22    electricity_duty = basic_bill * electricity_duty_rate
23
24    # Calculate total bill amount
25    total_bill = basic_bill + fixed_charge + customer_charge + electricity_duty
26
27    # Display the breakdown of charges
28    print(f"Units Consumed: {units_consumed} units")
29    print(f"Basic Bill Amount: ${basic_bill:.2f}")
30    print(f"Fixed Charge: ${fixed_charge:.2f}")
31    print(f"Customer Charge: ${customer_charge:.2f}")
32    print(f"Electricity Duty (5%): ${electricity_duty:.2f}")
33    print(f"Total Bill Amount: ${total_bill:.2f}")
34
35 # Example usage:
36 units_consumed = 350
37 calculate_electricity_bill(units_consumed)

```

```

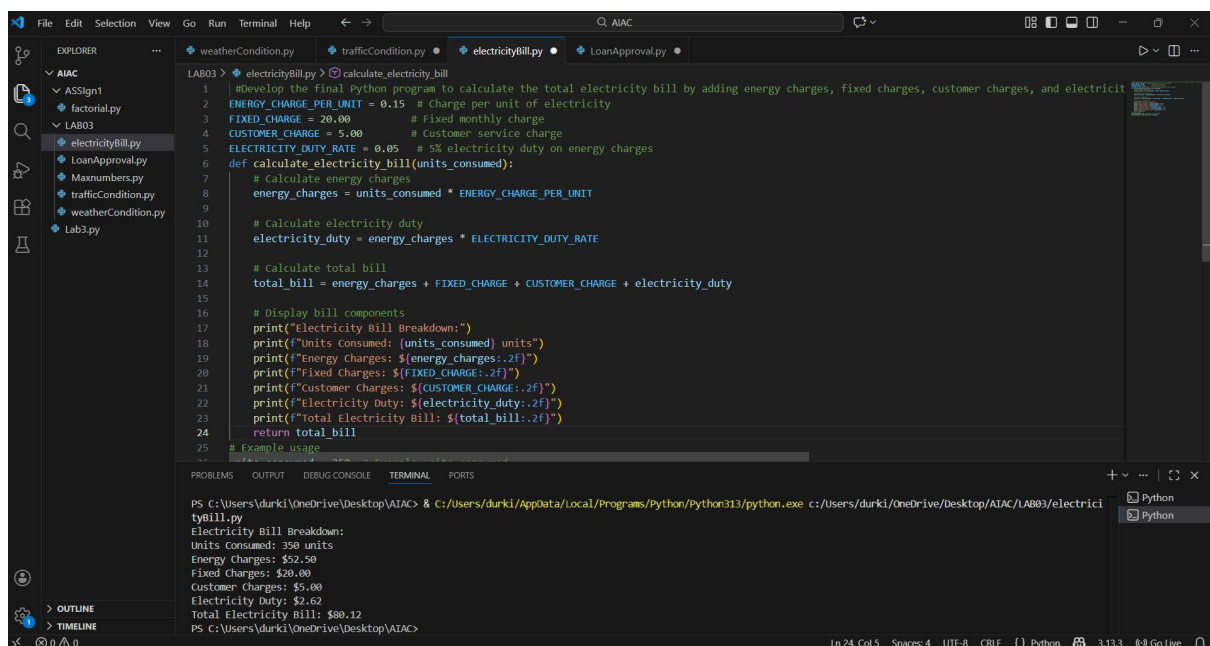
Units Consumed: 350 units
Basic Bill Amount: $52.50
Fixed Charge: $5.00
Customer Charge: $2.00
Electricity Duty (5%): $2.62
Total Bill Amount: $62.12

```

The program is further enhanced to include additional charges such as fixed charges, customer charges, and electricity duty. Each charge is calculated separately based on customer type and usage. Displaying individual charges improves transparency and billing accuracy

Task5:

Prompt: Develop the final Python program to calculate the total electricity bill by adding energy charges, fixed charges, customer charges, and electricity duty. Display all bill components clearly in a structured format.



The screenshot shows a Python IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'AIAC' with several files, including 'electricityBill.py'. The code editor displays the following Python code:

```
1 #Develop the final Python program to calculate the total electricity bill by adding energy charges, fixed charges, customer charges, and electricit
2 ENERGY_CHARGE_PER_UNIT = 0.15 # Charge per unit of electricity
3 FIXED_CHARGE = 20.00 # Fixed monthly charge
4 CUSTOMER_CHARGE = 5.00 # Customer service charge
5 ELECTRICITY_DUTY_RATE = 0.05 # 5% electricity duty on energy charges
6 def calculate_electricity_bill(units_consumed):
7     # Calculate energy charges
8     energy_charges = units_consumed * ENERGY_CHARGE_PER_UNIT
9
10    # Calculate electricity duty
11    electricity_duty = energy_charges * ELECTRICITY_DUTY_RATE
12
13    # Calculate total bill
14    total_bill = energy_charges + FIXED_CHARGE + CUSTOMER_CHARGE + electricity_duty
15
16    # Display bill components
17    print("Electricity Bill Breakdown:")
18    print(f"Units Consumed: {units_consumed} units")
19    print(f"Energy Charges: ${energy_charges:.2f}")
20    print(f"Fixed Charges: ${FIXED_CHARGE:.2f}")
21    print(f"Customer Charges: ${CUSTOMER_CHARGE:.2f}")
22    print(f"Electricity Duty: ${electricity_duty:.2f}")
23    print(f"Total Electricity Bill: ${total_bill:.2f}")
24    return total_bill
25 # Example usage
```

The terminal at the bottom shows the output of the program:

```
PS C:\Users\durki\OneDrive\Desktop\AIAC> & C:/Users/durki/AppData/Local/Programs/Python/Python313/python.exe c:/Users/durki/OneDrive/Desktop/AIAC/LAB03/electrici
tyBill.py
Electricity Bill Breakdown:
Units Consumed: 350 units
Energy Charges: $52.50
Fixed Charges: $20.00
Customer Charges: $5.00
Electricity Duty: $2.62
Total Electricity Bill: $80.12
PS C:\Users\durki\OneDrive\Desktop\AIAC>
```

In the final task, all calculated components are combined to generate the total electricity bill. Energy charges, fixed charges, customer charges, and electricity duty are added together. The bill is displayed in a structured format, resembling a real-world electricity bill.