

Assignment 3.3

AIAC

Name: D.Shruthi

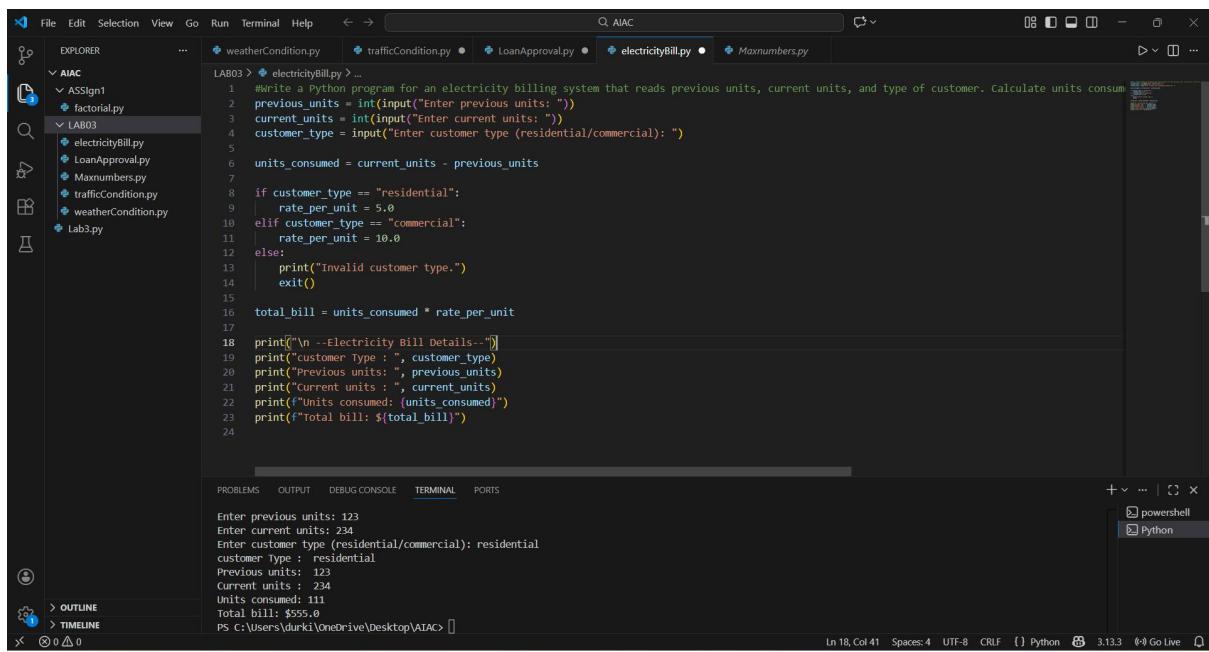
HT No: 2303A52407

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



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the AIAC folder, including ASign1, factorial.py, LAB03 (which contains electricityBill.py), Maxnumbers.py, trafficCondition.py, and weatherCondition.py.
- Code Editor:** Displays the content of electricityBill.py:

```
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 and display the bill details clearly without
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--")
19 print("Customer Type : ", customer_type)
20 print("Previous units: ", previous_units)
21 print("Current units : ", current_units)
22 print("Units consumed: (units_consumed)")
23 print("Total bill: ${total_bill}")
```
- Terminal:** Shows the output of running the program:

```
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
```
- Status Bar:** Shows file path (C:\Users\durki\OneDrive\Desktop\AIAC\electricityBill.py), line 18, column 41, and other standard status bar information.

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

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a project structure under 'AIAC' with files like 'weatherCondition.py', 'trafficCondition.py', 'LoanApproval.py', 'electricityBill.py', 'Maxnumbers.py', 'Lab3.py', and 'factorial.py'. 'electricityBill.py' is the active file.
- Code Editor:** Displays the Python code for calculating electricity bills based on customer type (Domestic, Commercial, Industrial). The code uses if-elif-else statements to determine the rate per unit and calculate the total bill.
- Terminal:** Shows the command-line output of the program. It prompts for 'customer type' (Commercial), prints details like 'Customer Type : Commercial', and calculates a total bill of '\$3120.0'.
- Status Bar:** Shows the current line (Ln 25, Col 36), character count (Spaces: 4), encoding (UTF-8), and file type (Python).

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
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')
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:
...

```

tyBill.py
Consumer Type: residential
Units Consumed: 350
Energy Charge: \$350.00
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
2 def calculate_electricity_bill(units_consumed):
3     """
4         Define the rates and charges
5         rate_per_unit = 0.15 # Rate per unit of electricity
6         fixed_charge = 5.00 # Fixed charge
7         customer_charge = 2.00 # Customer charge
8         electricity_duty_rate = 0.05 # 5% electricity duty
9
10        # Calculate the basic bill amount
11        basic_bill = units_consumed * rate_per_unit
12
13        # Calculate electricity duty
14        electricity_duty = basic_bill * electricity_duty_rate
15
16        # Calculate total bill amount
17        total_bill = basic_bill + fixed_charge + customer_charge + electricity_duty
18
19        # Display the breakdown of charges
20        print("Units Consumed: {units_consumed} units")
21        print("Basic Bill Amount: ${basic_bill:.2f}")
22        print("Fixed Charge: ${fixed_charge:.2f}")
23        print("Customer Charge: ${customer_charge:.2f}")
24        print("Electricity Duty (5%): ${electricity_duty:.2f}")
25        print("Total Bill Amount: ${total_bill:.2f}")
...

```

Units Consumed: 350 units
Basic Bill Amount: \$52.50
Units Consumed: 350 units
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 Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows files in the current workspace, including `ASSign1`, `LAB03`, and `AIAC` folders containing `electricityBill.py`, `trafficCondition.py`, `electricityBill.py`, `LoanApproval.py`, and `Lab3.py`.
- Editor:** The main editor area displays the content of `electricityBill.py`. The code calculates the total electricity bill by adding energy charges, fixed charges, customer charges, and electricity duty.
- Terminal:** The terminal at the bottom shows the execution of the script and its output. The command `python.exe c:/Users/durki/OneDrive/Desktop/AIAC/LAB03/electricityBill.py` was run, followed by the program's breakdown of components and the final total bill.
- Status Bar:** The status bar at the bottom right indicates the current file is `Python`, with 3.13.3 version information and a "Go Live" button.

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.