

NAME:A.SHASHIDHAR H.NO: 2303A51798 BATCH:26

ASSIGNMENT-3.3

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	Academic Year:2025-2026
Course Coordinator Name		Dr. Rishabh Mittal	
Instructor(s) Name		Mr. S Naresh Kumar	
		Ms. B. Swathi	
		Dr. Sasanko Shekhar Gantayat	
		Mr. Md Sallauddin	
		Dr. Mathivanan	
		Mr. Y Srikanth	
		Ms. N Shilpa	
		Dr. Rishabh Mittal (Coordinator)	
		Dr. R. Prashant Kumar	
		Mr. Ankushavali MD	
		Mr. B Viswanath	
		Ms. Sujitha Reddy	
		Ms. A. Anitha	
		Ms. M.Madhuri	
		Ms. Katherashala Swetha	
		Ms. Velpula sumalatha	
		Mr. Bingi Raju	
Course Code	23CS002PC304	Course Title	AI Assisted Coding
Year/Sem	III/I	Regulation	R23
Date and Day of Assignment	Week 2 - Wednesday	Time(s)	23CSBTB01 To 23CSBTB52
Duration	2 Hours	Applicable to Batches	All batches
Assignment Number: 3.3(Present assignment number)/24(Total number of assignments)			

Q.No.	Question	Expected Time to complete
1	<p>Lab 3: Application for TGNPDCL – Electricity Bill Generation Using Python & AI Tools</p> <p>Lab Objectives</p> <ul style="list-style-type: none"> To design a real-world electricity billing application using Python To use AI-assisted coding tools for logic generation and optimization To understand conditional logic and arithmetic operations To generate structured billing output similar to utility bills <p>Lab Outcomes (LOs) After completing this lab, students will be able to:</p>	Week2 - Wednesday

ASSIGNMENT-3.3

- Read and validate user input in Python
- Apply conditional logic for tariff-based billing
- Use AI tools to assist in program development
- Calculate and display electricity bill components
- Build a complete real-time application

Task 1: AI-Generated Logic for Reading Consumer Details**Scenario**

An electricity billing system must collect accurate consumer data.

Task Description

Use an AI tool (GitHub Copilot / Gemini) to generate a Python program that:

- Reads:
 - Previous Units (PU)
 - Current Units (CU)
 - Type of Customer
- Calculates units consumed
- Implements logic directly in the main program (no functions)

Expected Output

- Correct input reading
- Units consumed calculation
- Screenshot showing AI-generated code
- Sample input and output

```

C:\Users\shash> python AAC A (3.3).py
1 # Simple electricity consumer data reader and units calculator
2
3 # Read inputs
4 previous_units = float(input("Enter Previous Units (PU): "))
5 current_units = float(input("Enter Current Units (CU): "))
6 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")
7
8 # Calculate units consumed
9 units_consumed = current_units - previous_units
10
11 # Output the result
12 print(f"Units Consumed: {units_consumed}")
  
```

```

PS C:\Users\shash> & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\vs
231' '-' 'c:\Users\shash\AAC A (3.3).py'
Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 130.0
PS C:\Users\shash>
  
```

Task 2: Energy Charges Calculation Based on Units Consumed**Scenario**

Energy charges depend on the number of units consumed and customer type.

Task Description

Review the AI-generated code from Task 1 and extend it to:

- Calculate **Energy Charges (EC)**
- Use conditional statements based on:
 - Domestic
 - Commercial
 - Industrial consumers
- Improve readability using AI prompts such as:

ASSIGNMENT-3.3

- "Simplify energy charge calculation logic"
- "Optimize conditional statements"

Expected Output

- Correct EC calculation
- Clear conditional logic
- Original and improved versions (optional)
- Sample execution results

```

1  # Simplify energy charge calculation logic
2  # Optimize conditional statements
3
4  previous_units = float(input("Enter Previous Units (PU): "))
5  current_units = float(input("Enter Current Units (CU): "))
6  customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")
7
8  # Calculate units consumed
9  units_consumed = current_units - previous_units
10
11 # Calculate Energy Charges (EC) based on type and slabs
12 if customer_type == "Domestic":
13     if units_consumed <= 100:
14         ec = units_consumed * 1.0
15     elif units_consumed <= 200:
16         ec = 100 * 1.0 + (units_consumed - 100) * 2.0
17     else:
18         ec = 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
19 elif customer_type == "Commercial":
20     if units_consumed <= 100:
21         ec = units_consumed * 1.5
22     elif units_consumed <= 200:
23         ec = 100 * 1.5 + (units_consumed - 100) * 2.5
24     else:
25         ec = 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0
26 elif customer_type == "Industrial":
27     if units_consumed <= 100:
28         ec = units_consumed * 2.0
29     elif units_consumed <= 200:
30         ec = 100 * 2.0 + (units_consumed - 100) * 3.0
31     else:
32         ec = 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
33 else:
34     ec = 0 # Invalid type
35     print("Invalid customer type!")
36
37 # Output
38 print(f"Units Consumed: {units_consumed}")
39 print(f"Energy Charges (EC): ${ec:.2f}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 130.0
PS C:\Users\shash> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\AAC A (3.3).py'
Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 130.0
Energy Charges (EC): $160.00
PS C:\Users\shash>

```

Task 3: Modular Design Using AI Assistance (Using Functions)**Scenario**

Billing logic must be reusable for multiple consumers.

Task Description

Use AI assistance to generate a Python program that:

- Uses user-defined functions to:
 - Calculate Energy Charges
 - Calculate Fixed Charges
- Returns calculated values
- Includes meaningful comments

Expected Output

- Function-based Python program
- Correct EC and FC values
- Screenshots of AI-assisted function generation
- Test cases with outputs

ASSIGNMENT-3.3

```

C:\Users\shash> AAC A (3.3).py > ...
1  # Modular Electricity Billing System
2
3  def calculate_energy_charges(customer_type, units_consumed):
4      """
5      Calculate Energy Charges based on customer type and units consumed.
6      Slabs: Domestic (1/2/3), Commercial (1.5/2.5/4), Industrial (2/3/5) per unit tiers.
7      """
8      if customer_type == "Domestic":
9          if units_consumed <= 100:
10             return units_consumed * 1.0
11          elif units_consumed <= 200:
12             return 100 * 1.0 + (units_consumed - 100) * 2.0
13          else:
14             return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
15      elif customer_type == "Commercial":
16          if units_consumed <= 100:
17             return units_consumed * 1.5
18          elif units_consumed <= 200:
19             return 100 * 1.5 + (units_consumed - 100) * 2.5
20          else:
21             return 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0
22      elif customer_type == "Industrial":
23          if units_consumed <= 100:
24             return units_consumed * 2.0
25          elif units_consumed <= 200:
26             return 100 * 2.0 + (units_consumed - 100) * 3.0
27          else:
28             return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
29      return 0 # Invalid type
30
31 def calculate_fixed_charges(customer_type):
32     """
33     Calculate Fixed Charges based on customer type.
34     Domestic: $100, Commercial: $200, Industrial: $300.
35     """
36     if customer_type == "Domestic":
37         return 100.0
38     elif customer_type == "Commercial":
39         return 200.0
40     elif customer_type == "Industrial":
41         return 300.0
42     return 0 # Invalid type
43
44 # Main program
45 previous_units = float(input("Enter Previous Units (PU): "))
46 current_units = float(input("Enter Current Units (CU): "))
47 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")
48
49 units_consumed = current_units - previous_units
50 ec = calculate_energy_charges(customer_type, units_consumed)
51 fc = calculate_fixed_charges(customer_type)
52
53 print(f"Units Consumed: {units_consumed}")
54 print(f"Energy Charges (EC): ${ec:.2f}")
55 print(f"Fixed Charges (FC): ${fc:.2f}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

shvAAC A (3.3).py
Units Consumed: 130.0
Energy Charges (EC): $160.00
PS C:\Users\shash> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\shvAAC A (3.3).py'
shvAAC A (3.3).py
Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 130.0
Energy Charges (EC): $160.00
Fixed Charges (FC): $100.00
PS C:\Users\shash>

```

Welcome SubSetSum.java lab - 2.html # lab - 2.css JS lab - 2.js resume dev

```

C:\Users\shash> AAC A (3.3).py > ...
3  def calculate_energy_charges(customer_type, units_consumed):
25      elif units_consumed <= 200:
26          return 100 * 2.0 + (units_consumed - 100) * 3.0
27      else:
28          return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
29      return 0 # Invalid type
30
31 def calculate_fixed_charges(customer_type):
32     """
33     Calculate Fixed Charges based on customer type.
34     Domestic: $100, Commercial: $200, Industrial: $300.
35     """
36     if customer_type == "Domestic":
37         return 100.0
38     elif customer_type == "Commercial":
39         return 200.0
40     elif customer_type == "Industrial":
41         return 300.0
42     return 0 # Invalid type
43
44 # Main program
45 previous_units = float(input("Enter Previous Units (PU): "))
46 current_units = float(input("Enter Current Units (CU): "))
47 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")
48
49 units_consumed = current_units - previous_units
50 ec = calculate_energy_charges(customer_type, units_consumed)
51 fc = calculate_fixed_charges(customer_type)
52
53 print(f"Units Consumed: {units_consumed}")
54 print(f"Energy Charges (EC): ${ec:.2f}")
55 print(f"Fixed Charges (FC): ${fc:.2f}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\shash> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\shvAAC A (3.3).py'
shvAAC A (3.3).py
Fixed Charges (FC): $100.00
PS C:\Users\shash> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\shvAAC A (3.3).py'
shvAAC A (3.3).py
Enter Previous Units (PU): 0
Enter Current Units (CU): 250
Enter Type of Customer (Domestic/Commercial/Industrial): Commercial
Units Consumed: 250.0
Energy Charges (EC): $600.00
Fixed Charges (FC): $200.00
PS C:\Users\shash>

```

ASSIGNMENT-3.3

Task 4: Calculation of Additional Charges**Scenario**

Electricity bills include multiple additional charges.

Task Description

Extend the program to calculate:

- **FC** – Fixed Charges
- **CC** – Customer Charges
- **ED** – Electricity Duty (percentage of EC)

Use AI prompts like:

- *"Add electricity duty calculation"*
- *"Improve billing accuracy"*

Expected Output

- Individual charge values printed
- Correct duty calculation
- Well-structured output
- Verified intermediate results

```

C:\Users\shash > AAC A (3.3).py > ...
1  # Extended Electricity Billing with Additional Charges
2
3  def calculate_energy_charges(customer_type, units_consumed):
4      """
5      Calculate Energy Charges based on customer type and units consumed.
6      Slabs: Domestic (1/2/3), Commercial (1.5/2.5/4), Industrial (2/3/5) per unit tiers.
7      """
8      if customer_type == "Domestic":
9          if units_consumed <= 100:
10             return units_consumed * 1.0
11          elif units_consumed <= 200:
12             return 100 * 1.0 + (units_consumed - 100) * 2.0
13          else:
14             return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
15      elif customer_type == "Commercial":
16          if units_consumed <= 100:
17             return units_consumed * 1.5
18          elif units_consumed <= 200:
19             return 100 * 1.5 + (units_consumed - 100) * 2.5
20          else:
21             return 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0
22      elif customer_type == "Industrial":
23          if units_consumed <= 100:
24             return units_consumed * 2.0
25          elif units_consumed <= 200:
26             return 100 * 2.0 + (units_consumed - 100) * 3.0
27          else:
28             return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
29      return 0 # Invalid type
30
31  def calculate_fixed_charges(customer_type):
32      """
33      Calculate Fixed Charges based on customer type.
34      Domestic: $100, Commercial: $200, Industrial: $300.
35      """
36      if customer_type == "Domestic":
37          return 100.0
38      elif customer_type == "Commercial":
39          return 200.0
40      elif customer_type == "Industrial":
41          return 300.0
42      return 0 # Invalid type
43
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
conda : The term 'conda' is not recognized as the name of a cmdlet, function, script file, or operable program.
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\shash> & "c:\Users\shash\anaconda3\envs\Shashidhar\python.exe" "c:\Users\shash\.vscode\extensions\ms-python.python\python\python.exe"
Enter Previous Units (PU): 150
Enter Current Units (CU): 200
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 130.0
Energy Charges (EC): $160.00
Fixed Charges (FC): $100.00
Customer Charges (CC): $50.00
Electricity Duty (ED): $16.00
PS C:\Users\shash>

```

ASSIGNMENT-3.3

The screenshot shows a VS Code editor with a Python script named `AAC A (3.3).py`. The script defines functions for calculating fixed charges, customer charges, electricity duty, and a main program that takes user input and prints the bill details. Below the editor, the terminal window shows the execution of the script, displaying the calculated values for units consumed, energy charges, fixed charges, customer charges, electricity duty, and the total bill amount.

```

31 def calculate_fixed_charges(customer_type):
32     return 100.0
33     elif customer_type == "Commercial":
34         return 200.0
35     elif customer_type == "Industrial":
36         return 300.0
37     return 0 # Invalid type
38
39 def calculate_customer_charges():
40     """Fixed Customer Charges: $50 for all types."""
41     return 50.0
42
43 def calculate_electricity_duty(ec):
44     """Electricity Duty: 10% of Energy Charges."""
45     return 0.10 * ec
46
47 # Main program
48 previous_units = float(input("Enter Previous Units (PU): "))
49 current_units = float(input("Enter Current Units (CU): "))
50 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip()
51
52 if customer_type not in ["Domestic", "Commercial", "Industrial"]:
53     print("Invalid type! Defaulting to Domestic.")
54     customer_type = "Domestic"
55
56 units_consumed = current_units - previous_units
57 ec = calculate_energy_charges(customer_type, units_consumed)
58 fc = calculate_fixed_charges(customer_type)
59 cc = calculate_customer_charges()
60 ed = calculate_electricity_duty(ec)
61
62 # Print individual charges
63 print(f"Units Consumed: {units_consumed}")
64 print(f"Energy Charges (EC): ${ec:.2f}")
65 print(f"Fixed Charges (FC): ${fc:.2f}")
66 print(f"Customer Charges (CC): ${cc:.2f}")
67 print(f"Electricity Duty (ED): ${ed:.2f}")

```

```

PS C:\Users\shash> & "c:\Users\shash\anaconda3\envs\Shashidhan\python.exe" "c:\Users\shash\vscode\extens
Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
Units Consumed: 130.0
Energy Charges (EC): $160.00
Fixed Charges (FC): $100.00
Customer Charges (CC): $50.00
Electricity Duty (ED): $16.00
PS C:\Users\shash>

```

Task 5: Final Bill Generation and Output Analysis Scenario

The final electricity bill must present all values clearly.

Task Description

Develop the final Python application to:

- Calculate total bill:
- Total Bill = EC + FC + CC + ED
- Display:
 - Energy Charges (EC)
 - Fixed Charges (FC)
 - Customer Charges (CC)
 - Electricity Duty (ED)
 - Total Bill Amount
- Analyze the program based on:
 - Accuracy
 - Readability
 - Real-world applicability

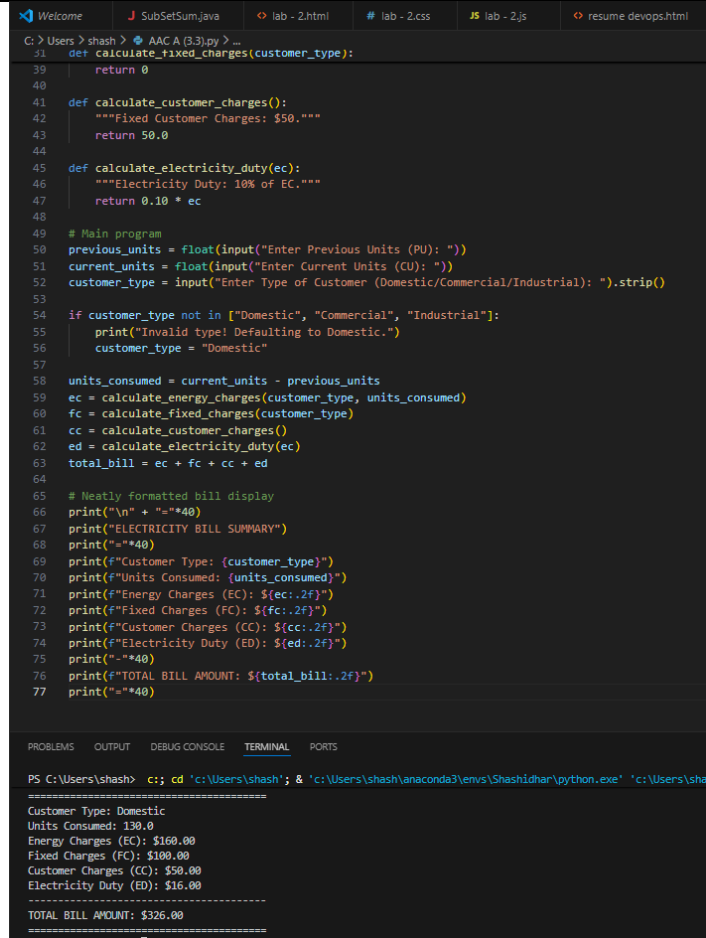
Expected Output

- Complete electricity bill output
- Neatly formatted display
- Sample input/output
- Short analysis paragraph

ASSIGNMENT-3.3

```
C: > Users > shash > AAC A (3.3).py > ...
1 # Final Electricity Bill Generator
2
3 def calculate_energy_charges(customer_type, units_consumed):
4     """
5     Calculate Energy Charges based on customer type and units consumed.
6     Slabs: Domestic (1/2/3), Commercial (1.5/2.5/4), Industrial (2/3/5) per unit tiers.
7     """
8     if customer_type == "Domestic":
9         if units_consumed <= 100:
10             return units_consumed * 1.0
11         elif units_consumed <= 200:
12             return 100 * 1.0 + (units_consumed - 100) * 2.0
13         else:
14             return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
15     elif customer_type == "Commercial":
16         if units_consumed <= 100:
17             return units_consumed * 1.5
18         elif units_consumed <= 200:
19             return 100 * 1.5 + (units_consumed - 100) * 2.5
20         else:
21             return 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0
22     elif customer_type == "Industrial":
23         if units_consumed <= 100:
24             return units_consumed * 2.0
25         elif units_consumed <= 200:
26             return 100 * 2.0 + (units_consumed - 100) * 3.0
27         else:
28             return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
29     return 0
30
31 def calculate_fixed_charges(customer_type):
32     """Fixed Charges: Domestic $100, Commercial $200, Industrial $300."""
33     if customer_type == "Domestic":
34         return 100.0
35     elif customer_type == "Commercial":
36         return 200.0
37     elif customer_type == "Industrial":
38         return 300.0
39     return 0
40
41 def calculate_customer_charges():
42     """Fixed Customer Charges: $50 """
43
44 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
45
46 PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\V
47 Enter Previous Units (PU): 150
48 Enter Current Units (CU): 280
49 Enter Type of Customer (Domestic/Commercial/Industrial): Domestic
50
51 *****
52 ELECTRICITY BILL SUMMARY
53 *****
54 Customer Type: Domestic
55 Units Consumed: 130.0
56 Energy Charges (EC): $160.00
57 Fixed Charges (FC): $100.00
```

ASSIGNMENT-3.3



```
39     return 0
40
41 def calculate_customer_charges():
42     """Fixed Customer Charges: $50."""
43     return 50.0
44
45 def calculate_electricity_duty(ec):
46     """Electricity Duty: 10% of EC."""
47     return 0.10 * ec
48
49 # Main program
50 previous_units = float(input("Enter Previous Units (PU): "))
51 current_units = float(input("Enter Current Units (CU): "))
52 customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ").strip()
53
54 if customer_type not in ["Domestic", "Commercial", "Industrial"]:
55     print("Invalid type! Defaulting to Domestic.")
56     customer_type = "Domestic"
57
58 units_consumed = current_units - previous_units
59 ec = calculate_energy_charges(customer_type, units_consumed)
60 fc = calculate_fixed_charges(customer_type)
61 cc = calculate_customer_charges()
62 ed = calculate_electricity_duty(ec)
63 total_bill = ec + fc + cc + ed
64
65 # Neatly formatted bill display
66 print("\n" + "="*40)
67 print("ELECTRICITY BILL SUMMARY")
68 print("="*40)
69 print(f"Customer Type: {customer_type}")
70 print(f"Units Consumed: {units_consumed}")
71 print(f"Energy Charges (EC): ${ec:.2f}")
72 print(f"Fixed Charges (FC): ${fc:.2f}")
73 print(f"Customer Charges (CC): ${cc:.2f}")
74 print(f"Electricity Duty (ED): ${ed:.2f}")
75 print("="*40)
76 print(f"TOTAL BILL AMOUNT: ${total_bill:.2f}")
77 print("="*40)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash> cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\...'
Customer Type: Domestic
Units Consumed: 130.0
Energy Charges (EC): $160.00
Fixed Charges (FC): $100.00
Customer Charges (CC): $50.00
Electricity Duty (ED): $16.00
TOTAL BILL AMOUNT: $326.00
```

Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.

NAME:A.SHASHIDHAR

H.NO:2303A51798

BATCH:26

ASSIGNMENT – 3.4

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING																		
Program Name: B. Tech		Assignment Type: Lab	Academic Year: 2025-2026																	
Course Coordinator Name		Dr. Rishabh Mittal																		
Instructor(s) Name		<table border="1"> <tr><td>Mr. S Naresh Kumar</td></tr> <tr><td>Ms. B. Swathi</td></tr> <tr><td>Dr. Sasanko Shekhar Gantayat</td></tr> <tr><td>Mr. Md Sallauddin</td></tr> <tr><td>Dr. Mathivanan</td></tr> <tr><td>Mr. Y Srikanth</td></tr> <tr><td>Ms. N Shilpa</td></tr> <tr><td>Dr. Rishabh Mittal (Coordinator)</td></tr> <tr><td>Dr. R. Prashant Kumar</td></tr> <tr><td>Mr. Ankushavali MD</td></tr> <tr><td>Mr. B Viswanath</td></tr> <tr><td>Ms. Sujitha Reddy</td></tr> <tr><td>Ms. A. Anitha</td></tr> <tr><td>Ms. M.Madhuri</td></tr> <tr><td>Ms. Katherashala Swetha</td></tr> <tr><td>Ms. Velpula sumalatha</td></tr> <tr><td>Mr. Bingi Raju</td></tr> </table>		Mr. S Naresh Kumar	Ms. B. Swathi	Dr. Sasanko Shekhar Gantayat	Mr. Md Sallauddin	Dr. Mathivanan	Mr. Y Srikanth	Ms. N Shilpa	Dr. Rishabh Mittal (Coordinator)	Dr. R. Prashant Kumar	Mr. Ankushavali MD	Mr. B Viswanath	Ms. Sujitha Reddy	Ms. A. Anitha	Ms. M.Madhuri	Ms. Katherashala Swetha	Ms. Velpula sumalatha	Mr. Bingi Raju
Mr. S Naresh Kumar																				
Ms. B. Swathi																				
Dr. Sasanko Shekhar Gantayat																				
Mr. Md Sallauddin																				
Dr. Mathivanan																				
Mr. Y Srikanth																				
Ms. N Shilpa																				
Dr. Rishabh Mittal (Coordinator)																				
Dr. R. Prashant Kumar																				
Mr. Ankushavali MD																				
Mr. B Viswanath																				
Ms. Sujitha Reddy																				
Ms. A. Anitha																				
Ms. M.Madhuri																				
Ms. Katherashala Swetha																				
Ms. Velpula sumalatha																				
Mr. Bingi Raju																				
CourseCode	23CS002PC304	Course Title	AI Assisted Coding																	
Year/Sem	III/II	Regulation	R23																	
Date and Day of Assignment	Week2	Time(s)	23CSBTB01 To 23CSBTB52																	
Duration	2 Hours	Applicable to Batches	All batches																	
Assignment Number: 3.4 (Present assignment number)/ 24 (Total number of assignments)																				
Q.No.	Question		Expected Time to complete																	
1	Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and		Week2																	

	<p>Few-shot Techniques</p> <p>Task 1: Zero-shot Prompt – Fibonacci Series Generator</p> <p>Task Description #1</p> <ul style="list-style-type: none">• Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers. <p>Expected Output #1</p> <ul style="list-style-type: none">• A complete Python function generated by Copilot without any example provided.• Correct output for sample input N = 7 → 0 1 1 2 3 5 8• Observation on how Copilot understood the instruction with zero context.	
--	---	--

```
C: > Users > shash > AAC A(3.4).py > ...
1  def print_fibonacci(n):
2      if n <= 0:
3          return
4      a, b = 0, 1
5      print(a, end=" ")
6      if n > 1:
7          print(b, end=" ")
8      for i in range(2, n):
9          a, b = b, a + b
10         print(b, end=" ")
11     print()
12
13     # Test with input N = 7
14     print_fibonacci(7)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Sha
Customer Charges (CC): $50.00
Electricity Duty (ED): $16.00
-----
TOTAL BILL AMOUNT: $326.00
=====
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Sha
PS C:\Users\shash> 7
7
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Sha
0 1 1 2 3 5 8
PS C:\Users\shash> 
```

Task 2: One-shot Prompt – List Reversal Function

Task Description #2

- Write a comment prompt to reverse a list and provide one example below the comment to guide Copilot.

Expected Output #2

- Copilot-generated function to reverse a list using slicing or loop.
- Output: [3, 2, 1] for input [1, 2, 3]
- Observation on how adding a single example improved Copilot’s

accuracy.

```
C: > Users > shash > AAC A(3.4).py > ...
1  def reverse_list(lst):
2      return lst[::-1]
3
4  # Test with input [1, 2, 3]
5  result = reverse_list([1, 2, 3])
6  print(result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash> c::; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\er
-----
TOTAL BILL AMOUNT: $326.00
=====
PS C:\Users\shash> c::; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\er
PS C:\Users\shash> 7
7
PS C:\Users\shash> c::; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\er
0 1 1 2 3 5 8
PS C:\Users\shash> c::; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\er
[3, 2, 1]
PS C:\Users\shash> █
```

Task 3: Few-shot Prompt – String Pattern Matching

Task Description #3

- Write a comment with 2–3 examples to help Copilot understand how to check if a string starts with a capital letter and ends with a period.

Expected Output #3

- A function is `_valid()` that checks the pattern.
- Output: True or False based on input.

- Students reflect on how multiple examples guide Copilot to generate more accurate code.

```

C: > Users > shash > AAC A(3,4).py > ...
1  def is_valid(s):
2      if not s: # Empty string
3          return False
4      return s[0].isupper() and s[-1] == '.'
5
6  # Test inputs
7  print(is_valid("Hello.")) # True
8  print(is_valid("hello.")) # False
9  print(is_valid("Hello")) # False

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```

PS C:\Users\shash> c:: cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shas
PS C:\Users\shash> 7
7
PS C:\Users\shash> c:: cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shas
0 1 1 2 3 5 8
PS C:\Users\shash> c:: cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shas
[3, 2, 1]
PS C:\Users\shash> c:: cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shas
True
False
False
PS C:\Users\shash>

```

Task 4: Zero-shot vs Few-shot – Email Validator

Task Description #4

- First, prompt Copilot to write an email validation function using zero-shot (just the task in comment).
- Then, rewrite the prompt using few-shot examples.

Expected Output #4

- Compare both outputs:

Zero-shot may result in basic or generic validation.

Few-shot gives detailed and specific logic (e.g., @ and domain checking).

- Submit both code versions and note how few-shot improves

reliability.

```
C: > Users > shash > AAC A(3.4).py > ...
1  import re
2
3  def validate_email(email):
4      pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
5      return bool(re.match(pattern, email))
6
7  # Test inputs
8  print(validate_email("user@example.com")) # True
9  print(validate_email("user@"))           # False
10 print(validate_email("user.example.com")) # False
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.e
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.e
[3, 2, 1]
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.e
True
False
False
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.e
True
False
False
PS C:\Users\shash>
```

Task 5: Prompt Tuning – Summing Digits of a Number

Task Description #5

- Experiment with 2 different prompt styles to generate a function that returns the sum of digits of a number.

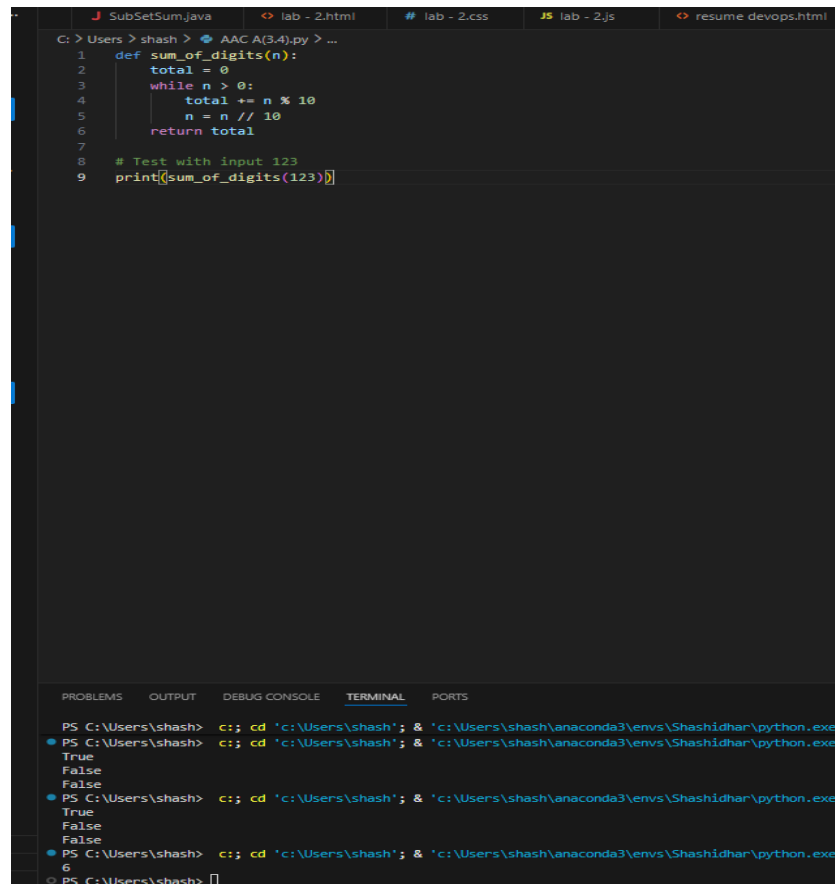
Style 1: Generic task prompt

Style 2: Task + Input/Output example

Expected Output #5

- Two versions of the `sum_of_digits()` function.
- Example Output: `sum_of_digits(123) → 6`
- Short analysis: which prompt produced cleaner or more

optimized code and why?



```
C: > Users > shash > AAC A(3.4).py > ...  
1 def sum_of_digits(n):  
2     total = 0  
3     while n > 0:  
4         total += n % 10  
5         n = n // 10  
6     return total  
7  
8 # Test with input 123  
9 print(sum_of_digits(123))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'  
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'  
True  
False  
False  
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'  
True  
False  
False  
● PS C:\Users\shash> c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe'  
6  
○ PS C:\Users\shash>
```

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots