

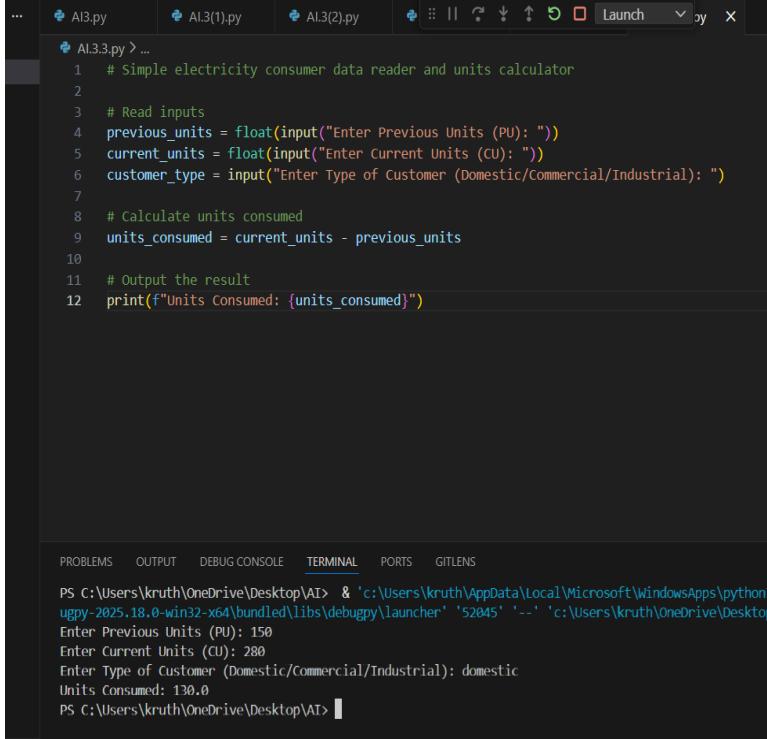
NAME:CH.Kruthankiran

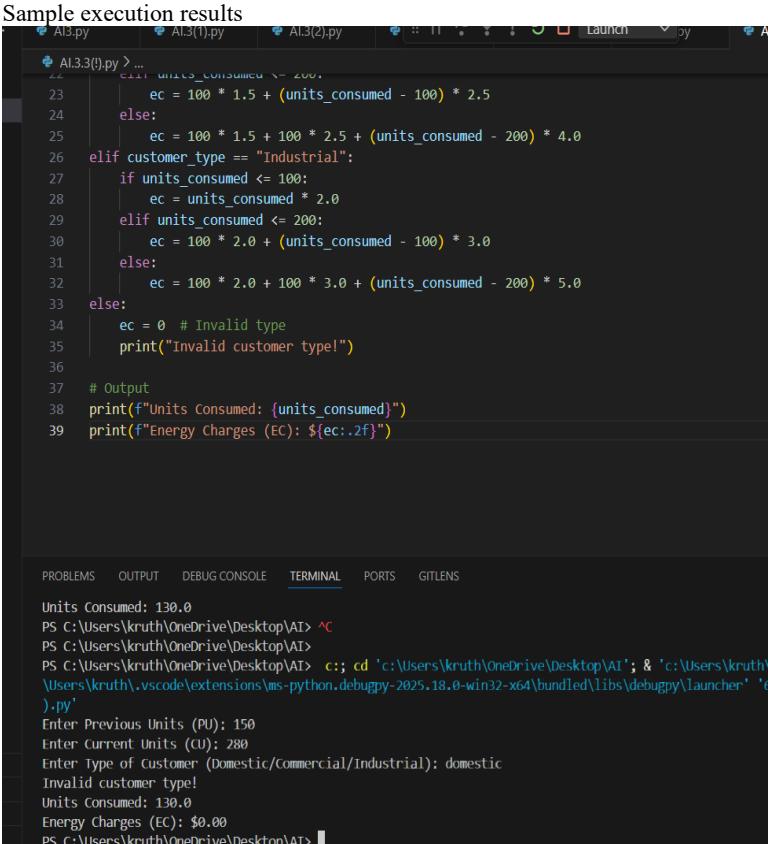
N.NO:2303A51404

BATCH:26

| 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 | Lab 3: Application for TGNPDCL – Electricity Bill Generation Using Python & AI Tools Lab Objectives <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 | Week2 - Wednesday |

| | | |
|--|--|--|
| | <ul style="list-style-type: none"> To generate structured billing output similar to utility bills <p>Lab Outcomes (LOs)</p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> 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 | |
| | <p>Task 1: AI-Generated Logic for Reading Consumer Details</p> <p>Scenario</p> <p>An electricity billing system must collect accurate consumer data.</p> <p>Task Description</p> <p>Use an AI tool (GitHub Copilot / Gemini) to generate a Python program that:</p> <ul style="list-style-type: none"> Reads: <ul style="list-style-type: none"> Previous Units (PU) Current Units (CU) Type of Customer Calculates units consumed Implements logic directly in the main program (no functions) <p>Expected Output</p> <ul style="list-style-type: none"> Correct input reading Units consumed calculation Screenshot showing AI-generated code Sample input and output  <pre> AI3.py AI3(1).py AI3(2).py Launch ... AI3.3.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}") </pre> <p>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS</p> <p>PS C:\Users\kruth\OneDrive\Desktop\AI> & 'c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\python3.10\pythonw.exe' 'c:\Users\kruth\OneDrive\Desktop\AI\AI3.3.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\kruth\OneDrive\Desktop\AI></p> | |
| | <p>Task 2: Energy Charges Calculation Based on Units Consumed</p> <p>Scenario</p> <p>Energy charges depend on the number of units consumed and customer type.</p> <p>Task Description</p> <p>Review the AI-generated code from Task 1 and extend it to:</p> <ul style="list-style-type: none"> Calculate Energy Charges (EC) | |

| | |
|--|--|
| | <ul style="list-style-type: none"> • Use conditional statements based on: <ul style="list-style-type: none"> ○ Domestic ○ Commercial ○ Industrial consumers • Improve readability using AI prompts such as: <ul style="list-style-type: none"> ○ “Simplify energy charge calculation logic” ○ “Optimize conditional statements” <p>Expected Output</p> <ul style="list-style-type: none"> • Correct EC calculation • Clear conditional logic • Original and improved versions (optional) • Sample execution results  <pre> AI3.py AI.1.py AI.2.py Launch A AI.3().py > ... `` ec = 100 * 1.5 + (units_consumed - 100) * 2.5 23 else: 24 ec = 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0 25 elif customer_type == "Industrial": 26 if units_consumed <= 100: 27 ec = units_consumed * 2.0 28 elif units_consumed <= 200: 29 ec = 100 * 2.0 + (units_consumed - 100) * 3.0 30 else: 31 ec = 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0 32 else: 33 ec = 0 # Invalid type 34 print("Invalid customer type!") 35 36 # Output 37 print(f"Units Consumed: {units_consumed}") 38 print(f"Energy Charges (EC): \${ec:.2f}") PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Units Consumed: 130.0 PS C:\Users\kruth\OneDrive\Desktop\AI> ^ PS C:\Users\kruth\OneDrive\Desktop\AI> PS C:\Users\kruth\OneDrive\Desktop\AI> c:; cd 'c:\Users\kruth\OneDrive\Desktop\AI'; & 'c:\Users\kruth\OneDrive\Desktop\AI\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' 'e).py' Enter Previous Units (PU): 150 Enter Current Units (CU): 280 Enter Type of Customer (Domestic/Commercial/Industrial): domestic Invalid customer type! Units Consumed: 130.0 Energy Charges (EC): \$0.00 PS C:\Users\kruth\OneDrive\Desktop\AI> </pre> <ul style="list-style-type: none"> • Function-based Python program • Correct EC and FC values • Screenshots of AI-assisted function generation • Test cases with outputs |
| | <p>Task 3: Modular Design Using AI Assistance (Using Functions)</p> <p>Scenario</p> <p>Billing logic must be reusable for multiple consumers.</p> <p>Task Description</p> <p>Use AI assistance to generate a Python program that:</p> <ul style="list-style-type: none"> • Uses user-defined functions to: <ul style="list-style-type: none"> ○ Calculate Energy Charges ○ Calculate Fixed Charges • Returns calculated values • Includes meaningful comments <p>Expected Output</p> <ul style="list-style-type: none"> • Function-based Python program • Correct EC and FC values • Screenshots of AI-assisted function generation • Test cases with outputs |

```

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
30 return 0 # Invalid type
31
32 def calculate_fixed_charges(customer_type):
33 """
34 Calculate Fixed Charges based on customer type.
35 Domestic: $100, Commercial: $200, Industrial: $300.
36 """
37 if customer_type == "Domestic":
38     return 100.0
39 elif customer_type == "Commercial":
40     return 200.0
41 elif customer_type == "Industrial":
42     return 300.0
43 return 0 # Invalid type
44
45
46
47
48
49
50
51
52
53
54
55

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

sh\AAC A (3.3).py'
Units Consumed: 130.0
Energy Charges (EC): $160.00
PS C:\Users\shash> c:\ 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): 200
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> |
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

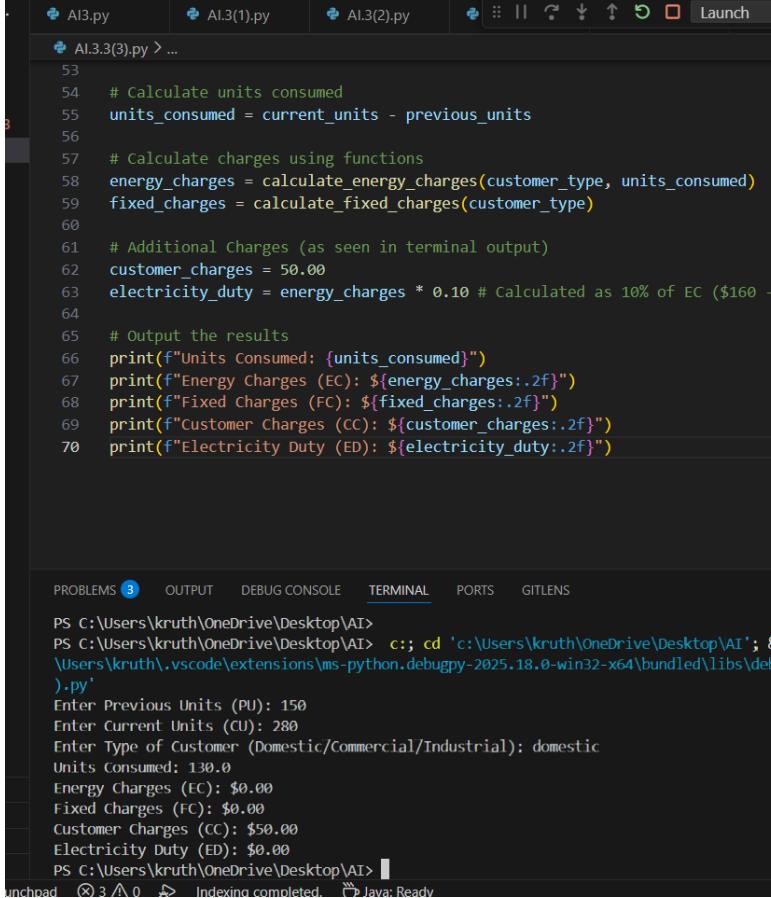
C:\> Users > shash > AAC A (3.3).py > ...
3 def calculate_energy_charges(customer_type, units_consumed):
4
5     elif units_consumed <= 200:
6         return 100 * 2.0 + (units_consumed - 100) * 3.0
7     else:
8         return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
9
10 return 0 # Invalid type
11
12
13 def calculate_fixed_charges(customer_type):
14 """
15 Calculate Fixed Charges based on customer type.
16 Domestic: $100, Commercial: $200, Industrial: $300.
17 """
18
19 if customer_type == "Domestic":
20     return 100.0
21 elif customer_type == "Commercial":
22     return 200.0
23 elif customer_type == "Industrial":
24     return 300.0
25 return 0 # Invalid type
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\shash> c:\ cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\AAC A (3.3).py'
Fixed Charges (FC): $100.00
PS C:\Users\shash> c:\ cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\AAC 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> |
```

| | | |
|--|---|--|
| | <p>Task 4: Calculation of Additional Charges</p> <p>Scenario</p> <p>Electricity bills include multiple additional charges.</p> <p>Task Description</p> <p>Extend the program to calculate:</p> <ul style="list-style-type: none"> • FC – Fixed Charges • CC – Customer Charges • ED – Electricity Duty (percentage of EC) <p>Use AI prompts like:</p> <ul style="list-style-type: none"> • “Add electricity duty calculation” • “Improve billing accuracy” <p>Expected Output</p> <ul style="list-style-type: none"> • Individual charge values printed • Correct duty calculation • Well-structured output • Verified intermediate results  <pre> AI3.py AI3(1).py AI3(2).py Launch AI3(3).py > ... 53 54 # Calculate units consumed 55 units_consumed = current_units - previous_units 56 57 # Calculate charges using functions 58 energy_charges = calculate_energy_charges(customer_type, units_consumed) 59 fixed_charges = calculate_fixed_charges(customer_type) 60 61 # Additional Charges (as seen in terminal output) 62 customer_charges = 50.00 63 electricity_duty = energy_charges * 0.10 # Calculated as 10% of EC (\$160 - \$160 * 10% = \$144) 64 65 # Output the results 66 print(f"Units Consumed: {units_consumed}") 67 print(f"Energy Charges (EC): \${energy_charges:.2f}") 68 print(f"Fixed Charges (FC): \${fixed_charges:.2f}") 69 print(f"Customer Charges (CC): \${customer_charges:.2f}") 70 print(f"Electricity Duty (ED): \${electricity_duty:.2f}") </pre> <p>PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS</p> <p>PS C:\Users\kruth\OneDrive\Desktop\AI> PS C:\Users\kruth\OneDrive\Desktop\AI> c:; cd 'c:\Users\kruth\OneDrive\Desktop\AI'; & \Users\kruth\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\del).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): \$0.00 Fixed Charges (FC): \$0.00 Customer Charges (CC): \$50.00 Electricity Duty (ED): \$0.00 PS C:\Users\kruth\OneDrive\Desktop\AI></p> | |
| | <p>Task 5: Final Bill Generation and Output Analysis</p> <p>Scenario</p> <p>The final electricity bill must present all values clearly.</p> <p>Task Description</p> <p>Develop the final Python application to:</p> <ul style="list-style-type: none"> • Calculate total bill: • Total Bill = EC + FC + CC + ED • Display: <ul style="list-style-type: none"> ◦ 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

```
AI3.py AI.3(1).py AI.3(2).py AI.3(..) Launch
AI.3.3(4).py > ...
1 # Final Electricity Billing System with Complete Charges Calculation
2
3 def calculate_energy_charges(customer_type, units_consumed):
4     """
5         Calculate Energy Charges (EC) based on customer type and tiered unit s
6     """
7     if customer_type == "Domestic":
8         if units_consumed <= 100:
9             return units_consumed * 1.0
10        elif units_consumed <= 200:
11            return 100 * 1.0 + (units_consumed - 100) * 2.0
12        else:
13            return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
14
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
23    elif customer_type == "Industrial":
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Enter Previous Units (PU): 150
Enter Current Units (CU): 280
Enter Type of Customer (Domestic/Commercial/Industrial): domestic

FINAL ELECTRICITY BILL

Customer Type : domestic
Units Consumed : 130.0

Energy Charges (EC): \$ 0.00
Fixed Charges (FC) : \$ 0.00
Customer Charges(CC): \$ 50.00

The screenshot shows a code editor window with a Python file named AI3.py open. The code defines functions to calculate energy charges based on customer type and units consumed, and to calculate fixed charges based on customer category. It then prompts for input (prev_units, curr_units, cust_type) and prints a final electricity bill summary.

```
AI3.py
AI3(1).py AI3(2).py AI3(3).py Launch
AI3(4).py > ...
3     def calculate_energy_charges(customer_type, units_consumed):
22
23         elif customer_type == "Industrial":
24             if units_consumed <= 100:
25                 return units_consumed * 2.0
26             elif units_consumed <= 200:
27                 return 100 * 2.0 + (units_consumed - 100) * 3.0
28             else:
29                 return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
30
31     return 0
32
33 def calculate_fixed_charges(customer_type):
34     """Calculate Fixed Charges (FC) based on customer category."""
35     charges = {"Domestic": 100.0, "Commercial": 200.0, "Industrial": 300.0}
36     return charges.get(customer_type, 0.0)
37
38 # --- Input Section ---
39 prev_units = float(input("Enter Previous Units (PU): "))
40 curr_units = float(input("Enter Current Units (CU): "))
41 cust_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")
42
43 # --- Calculation Section ---

PROBLEMS ③ OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
FINAL ELECTRICITY BILL
=====
Customer Type      : domestic
Units Consumed    : 130.0
-----
Energy Charges (EC): $ 0.00
Fixed Charges (FC) : $ 0.00
Customer Charges(CC): $ 50.00
Electricity Duty(ED): $ 0.00
-----
TOTAL BILL AMOUNT : $ 50.00
=====

PS C:\Users\kruthi\OneDrive\Desktop\AI>
```

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:CH.Kruthankiran

H.NO:2303A51404

BATCH:26

| 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 | | |
| 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 Few-shot Techniques | | Week2 |

| | | |
|--|---|--|
| | <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 $N = 7 \rightarrow 0 1 1 2 3 5 8$Observation on how Copilot understood the instruction with zero context. | |
|--|---|--|

The screenshot shows a VS Code interface with two tabs: 'AI3.py' and 'Untitled-2'. The 'AI3.py' tab contains the following Python code:

```
C: > Users > kruth > OneDrive > Desktop > AI3.py > ...
1 def print_fibonacci(n):
2     if n <= 0:
3         print("Please enter a positive integer.")
4         return
5
6     a, b = 0, 1
7     for i in range(n):
8         print(a, end=" ")
9         a, b = b, a + b
10
11 # Testing the function
12 print_fibonacci[7]
```

The 'TERMINAL' tab shows the following command-line session:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PS C:\Users\kruth\OneDrive\Desktop\AI> & 'c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\python3.11.5\python.exe' 'c:\Users\kruth\OneDrive\Desktop\AI\AI3.py'
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c:; cd 'c:\Users\kruth\OneDrive\Desktop\AI'
PS C:\Users\kruth\OneDrive\Desktop\AI> python AI3.py
0 1 1 2 3 5 8
PS C:\Users\kruth\OneDrive\Desktop\AI>
```

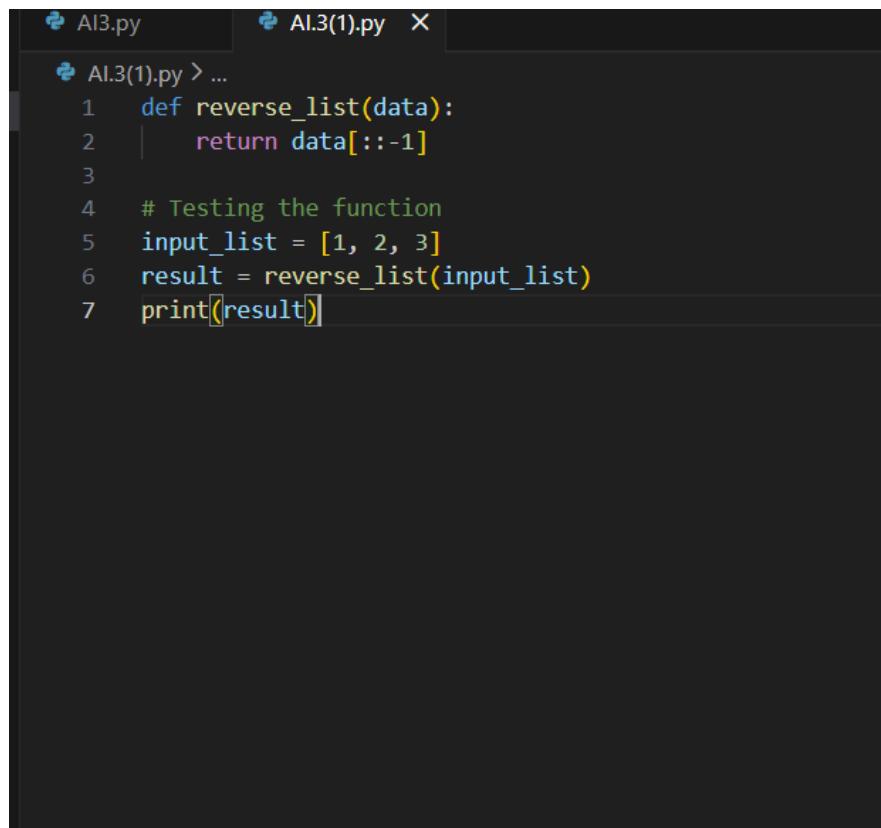
Task 2: One-shot Prompt – List Reversal Function

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.



```

AI3.py          AI.3(1).py X
AI.3(1).py > ...
1  def reverse_list(data):
2      return data[::-1]
3
4  # Testing the function
5  input_list = [1, 2, 3]
6  result = reverse_list(input_list)
7  print(result)

```

The screenshot shows a code editor interface with two tabs: "AI3.py" and "AI.3(1).py". The "AI.3(1).py" tab is active, displaying a Python script. The script defines a function "reverse_list" that takes a list "data" and returns its reverse using slicing. It then tests this function with an input list [1, 2, 3], stores the result in "result", and prints "result". Below the code editor is a terminal window showing the execution of the script. The terminal output is as follows:

```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS
n32-x64\bundled\libs\debugpy\launcher' '56953' '--' 'Untitled-1'
c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\python3.11.exe
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c:; cd 'c:\Users\kruth\OneDrive\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs'
0 1 1 2 3 5 8
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c:; cd 'c:\Users\kruth\OneDrive\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs'
[3, 2, 1]
PS C:\Users\kruth\OneDrive\Desktop\AI>

```

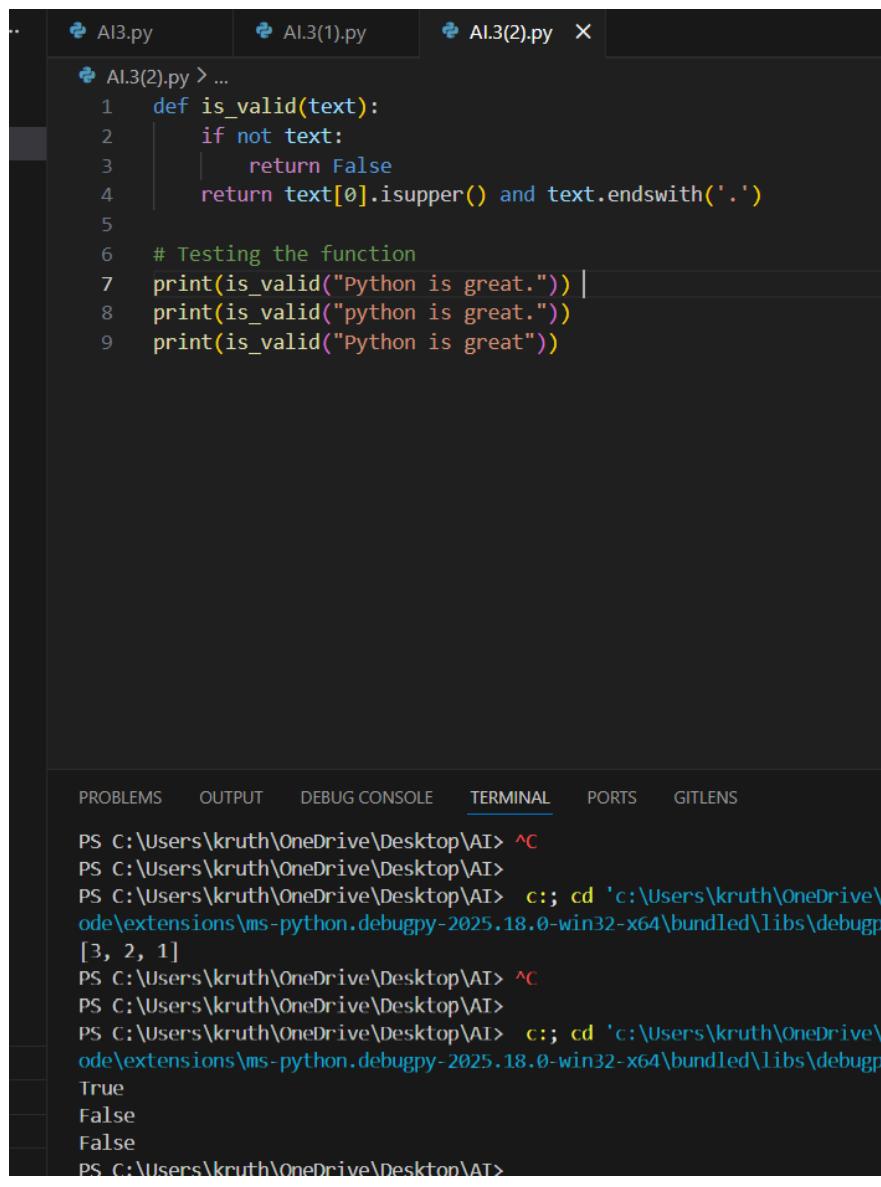
Task 3: Few-shot Prompt – String Pattern Matching

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.



The screenshot shows a code editor interface with a dark theme. At the top, there are three tabs: 'AI3.py', 'AI.3(1).py', and 'AI.3(2).py'. The 'AI.3(2).py' tab is active, showing the following Python code:

```
..  AI3.py  AI.3(1).py  AI.3(2).py  X
    AI.3(2).py > ...
1  def is_valid(text):
2      if not text:
3          return False
4      return text[0].isupper() and text.endswith('.')
5
6  # Testing the function
7  print(is_valid("Python is great."))
8  print(is_valid("python is great."))
9  print(is_valid("Python is great"))
```

Below the code editor is a terminal window showing the execution of the script. The terminal output is as follows:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c:; cd 'c:\Users\kruth\OneDrive\ode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugp
[3, 2, 1]
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c:; cd 'c:\Users\kruth\OneDrive\ode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugp
True
False
False
PS C:\Users\kruth\OneDrive\Desktop\AI>
```

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.

The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for AI3.py, AI.3(1).py, AI.3(2).py, and AI.3(3).py. The AI.3(3).py tab is active, displaying the following Python code:

```
AI.3(3).py > validate_email
1 import re
2
3 def validate_email(email):
4     # Basic regex for email validation
5     pattern = r"^\S+@\S+\.\S+$"
6     return bool(re.match(pattern, email))
```

Below the code editor is a terminal window with the following history:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLEN
[3, 2, 1]
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c;; cd 'c:\Users\kruth\OneDrive\Desktop\AI'
True
False
False
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c;; cd 'c:\Users\kruth\OneDrive\Desktop\AI'
PS C:\Users\kruth\OneDrive\Desktop\AI>
```

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.

| | | |
|--|---|--|
| | <p>Style 1: Generic task prompt</p> <p>Style 2: Task + Input/Output example</p> <p>Expected Output #5</p> <ul style="list-style-type: none">• 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? | |
|--|---|--|

The screenshot shows a code editor interface with a dark theme. At the top, there are tabs for AI3.py, AI.3(1).py, AI.3(2).py, AI.3(3).py, and AI.3(4).py. The AI.3(4).py tab is active, displaying the following Python code:

```
AI.3(4).py > ...
1 def sum_of_digits(n):
2     return sum(int(digit) for digit in str(n))
3
4 # Test case
5 print(sum_of_digits(123)) # Output: 6
```

Below the code editor is a terminal window showing the execution of the script. The terminal tabs at the top are PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and G. The TERMINAL tab is selected, showing the following command-line interaction:

```
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c::; cd 'c:\Users\kruth\OneDrive\Desktop\AI'
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c::; cd 'c:\Users\kruth\OneDrive\Desktop\AI'
PS C:\Users\kruth\OneDrive\Desktop\AI> ^C
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI> c::; cd 'c:\Users\kruth\OneDrive\Desktop\AI'
PS C:\Users\kruth\OneDrive\Desktop\AI> 6
PS C:\Users\kruth\OneDrive\Desktop\AI>
```

At the bottom of the terminal window, there is a note in bold black text:

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 | |
|--|---|--|