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** • 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

- To generate structured billing output similar to utility bills

Lab Outcomes (LOs)

After completing this lab, students will be able to:

- 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.

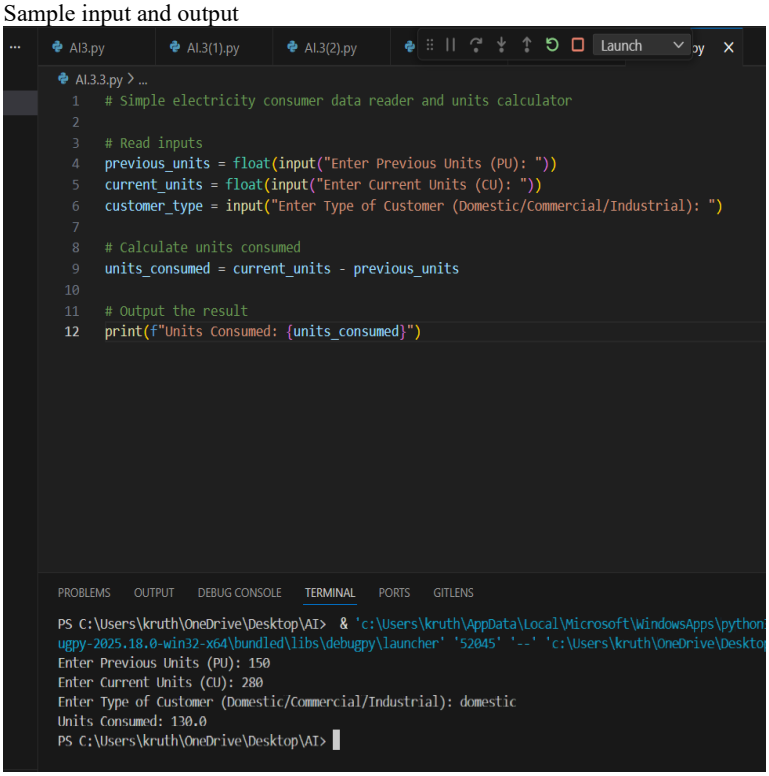
Task Description

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



Task 2: Energy Charges Calculation Based on Units Consumed

Scenario

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

Task Description

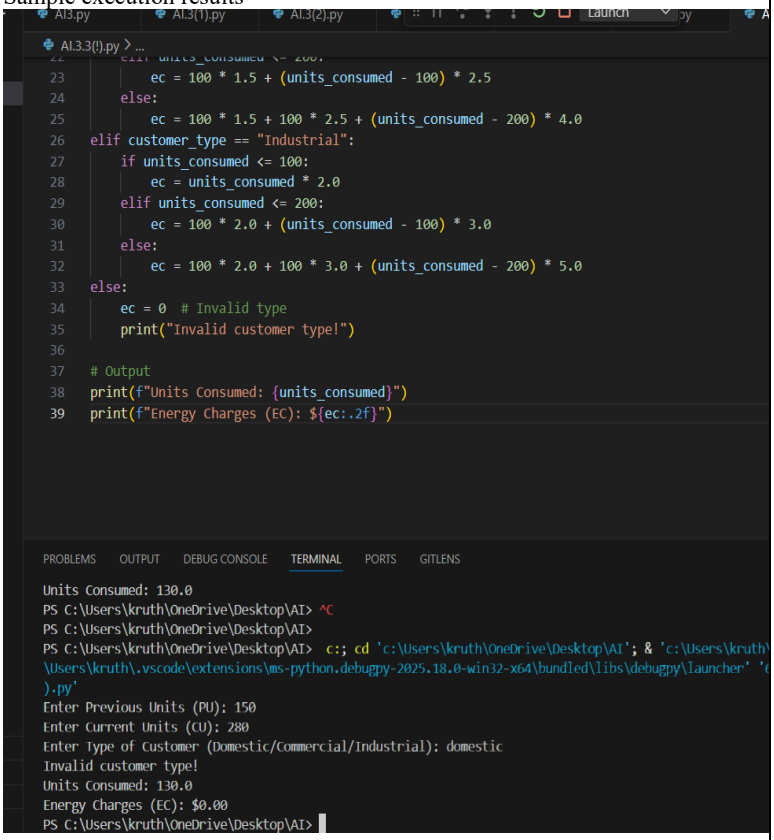
Review the AI-generated code from Task 1 and extend it to:

- Calculate **Energy Charges (EC)**

- Use conditional statements based on:
  - Domestic
  - Commercial
  - Industrial consumers
- Improve readability using AI prompts such as:
  - *"Simplify energy charge calculation logic"*
  - *"Optimize conditional statements"*

Expected Output

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



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

```
C: > Users > shash > AAC A (3.3).py > ...
# Modular Electricity Billing System

def calculate_energy_charges(customer_type, units_consumed):
    """
    Calculate Energy Charges based on customer type and units consumed.
    Slabs: Domestic (1/2/3), Commercial (1.5/2.5/4), Industrial (2/3/5) per unit tiers.
    """
    if customer_type == "Domestic":
        if units_consumed <= 100:
            return units_consumed * 1.0
        elif units_consumed <= 200:
            return 100 * 1.0 + (units_consumed - 100) * 2.0
        else:
            return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0
    elif customer_type == "Commercial":
        if units_consumed <= 100:
            return units_consumed * 1.5
        elif units_consumed <= 200:
            return 100 * 1.5 + (units_consumed - 100) * 2.5
        else:
            return 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0
    elif customer_type == "Industrial":
        if units_consumed <= 100:
            return units_consumed * 2.0
        elif units_consumed <= 200:
            return 100 * 2.0 + (units_consumed - 100) * 3.0
        else:
            return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
    return 0  # Invalid type

def calculate_fixed_charges(customer_type):
    """
    Calculate Fixed Charges based on customer type.
    Domestic: $100, Commercial: $200, Industrial: $300.
    """
    if customer_type == "Domestic":
        return 100.0
    elif customer_type == "Commercial":
        return 200.0
    elif customer_type == "Industrial":
        return 300.0
    return 0  # Invalid type
```

```
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:\User
sh\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
Fixed Charges (FC): $100.00
PS C:\Users\shash>
```

-

```
Welcome | SubSetSum.java | lab - 2.html | lab - 2.css | lab - 2.js | resume dev
C: > Users > shash > AAC A (3.3).py > ...
def calculate_energy_charges(customer_type, units_consumed):
        elif units_consumed <= 200:
            return 100 * 2.0 + (units_consumed - 100) * 3.0
        else:
            return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0
    return 0  # Invalid type

def calculate_fixed_charges(customer_type):
    """
    Calculate Fixed Charges based on customer type.
    Domestic: $100, Commercial: $200, Industrial: $300.
    """
    if customer_type == "Domestic":
        return 100.0
    elif customer_type == "Commercial":
        return 200.0
    elif customer_type == "Industrial":
        return 300.0
    return 0  # Invalid type

# Main program
previous_units = float(input("Enter Previous Units (PU): "))
current_units = float(input("Enter Current Units (CU): "))
customer_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")

units_consumed = current_units - previous_units
ec = calculate_energy_charges(customer_type, units_consumed)
fc = calculate_fixed_charges(customer_type)

print(f"Units Consumed: {units_consumed}")
print(f"Energy Charges (EC): ${ec:.2f}")
print(f"Fixed Charges (FC): ${fc:.2f}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\shash>  c:; cd 'c:\Users\shash'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:
sh\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:
sh\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>
```

Task 4: Calculation of Additional Charges

Scenario

Electricity bills include multiple additional charges.

Task Description

Extend the program to calculate:

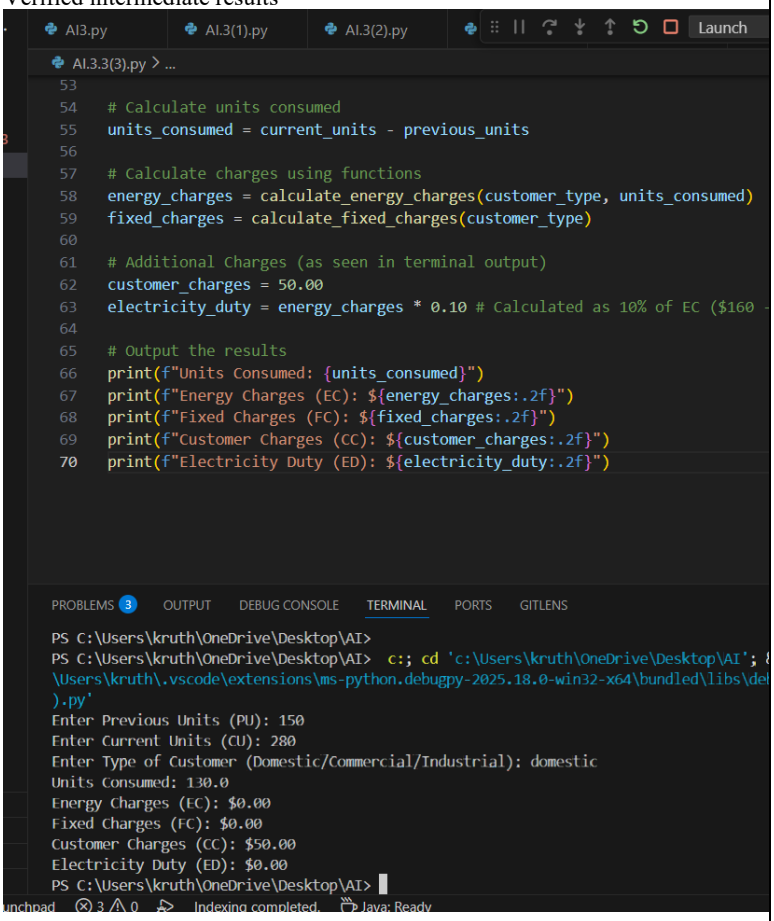
- **FC –** Fixed Charges
- **CC –** Customer Charges
- **ED –** Electricity Duty (percentage of EC)

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



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

AI.3.3(4).py > ...

```python
# Final Electricity Billing System with Complete Charges Calculation

def calculate_energy_charges(customer_type, units_consumed):
    """
    Calculate Energy Charges (EC) based on customer type and tiered unit s
    """
    if customer_type == "Domestic":
        if units_consumed <= 100:
            return units_consumed * 1.0
        elif units_consumed <= 200:
            return 100 * 1.0 + (units_consumed - 100) * 2.0
        else:
            return 100 * 1.0 + 100 * 2.0 + (units_consumed - 200) * 3.0

    elif customer_type == "Commercial":
        if units_consumed <= 100:
            return units_consumed * 1.5
        elif units_consumed <= 200:
            return 100 * 1.5 + (units_consumed - 100) * 2.5
        else:
            return 100 * 1.5 + 100 * 2.5 + (units_consumed - 200) * 4.0

    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
```

```python
def calculate_energy_charges(customer_type, units_consumed):

    elif customer_type == "Industrial":
        if units_consumed <= 100:
            return units_consumed * 2.0
        elif units_consumed <= 200:
            return 100 * 2.0 + (units_consumed - 100) * 3.0
        else:
            return 100 * 2.0 + 100 * 3.0 + (units_consumed - 200) * 5.0

    return 0

def calculate_fixed_charges(customer_type):
    """Calculate Fixed Charges (FC) based on customer category."""
    charges = {"Domestic": 100.0, "Commercial": 200.0, "Industrial": 300.0}
    return charges.get(customer_type, 0.0)

# --- Input Section ---
prev_units = float(input("Enter Previous Units (PU): "))
curr_units = float(input("Enter Current Units (CU): "))
cust_type = input("Enter Type of Customer (Domestic/Commercial/Industrial): ")

# --- Calculation Section ---
```

```
   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\kruth\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

	Task 1: Zero-shot Prompt – Fibonacci Series Generator **Task Description #1** • Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers. **Expected Output #1** • 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.	

AI3.py Untitled-2

C: > Users > kruth > OneDrive > Desktop > AI3.py > ...

```python
def print_fibonacci(n):
    if n <= 0:
        print("Please enter a positive integer.")
        return

    a, b = 0, 1
    for i in range(n):
        print(a, end=" ")
        a, b = b, a + b

# Testing the function
print_fibonacci(7)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\kruth\OneDrive\Desktop\AI>  & 'c:\Users\kruth\AppD
n32-x64\bundled\libs\debugpy\launcher' '56953' '--' 'Untitled-
c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\python3.11.
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
ode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\l
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

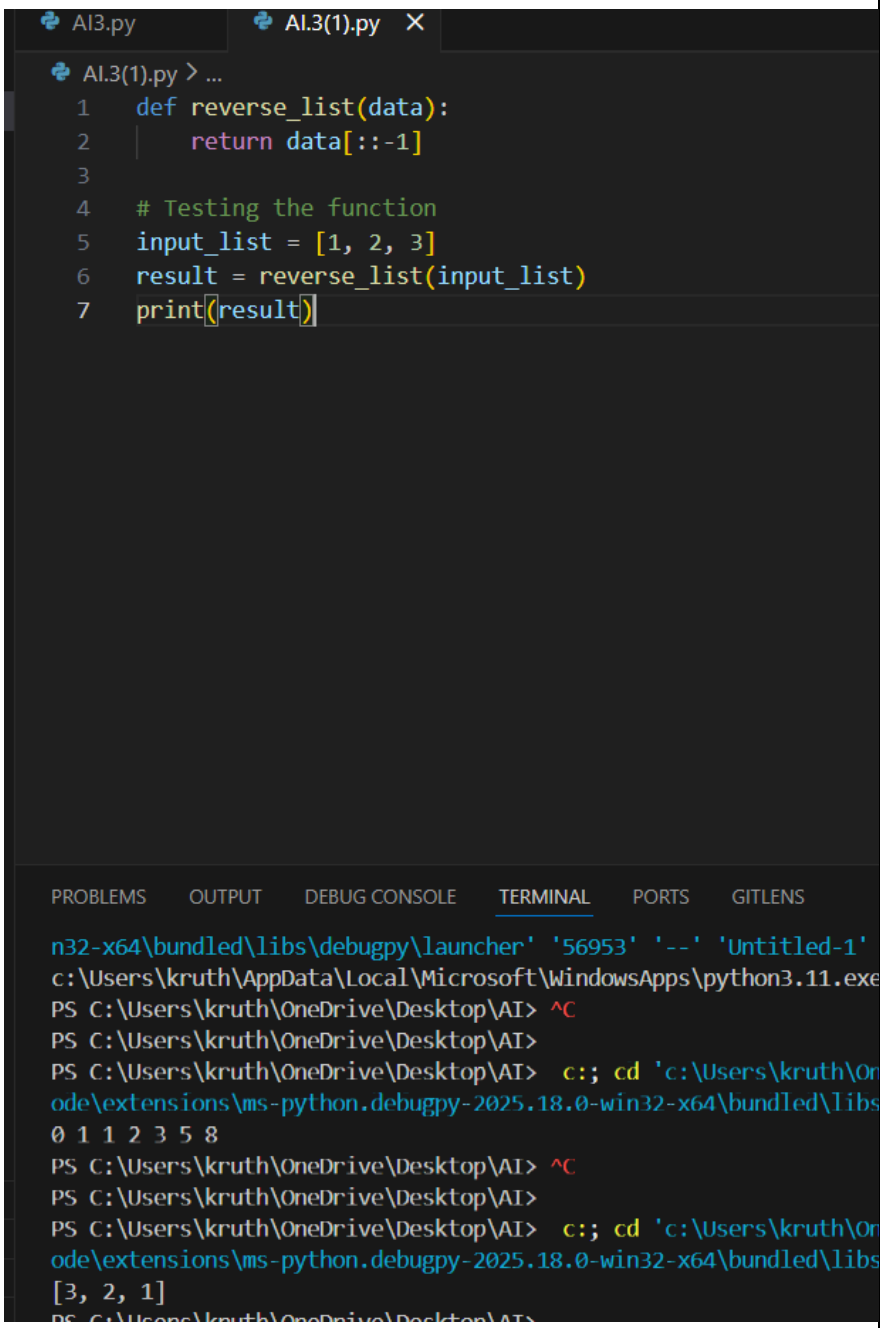
• Write a comment prompt to reverse a list and provide one example below the comment to guide Copilot.

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.



Task 3: Few-shot Prompt – String Pattern Matching

Task Description #3

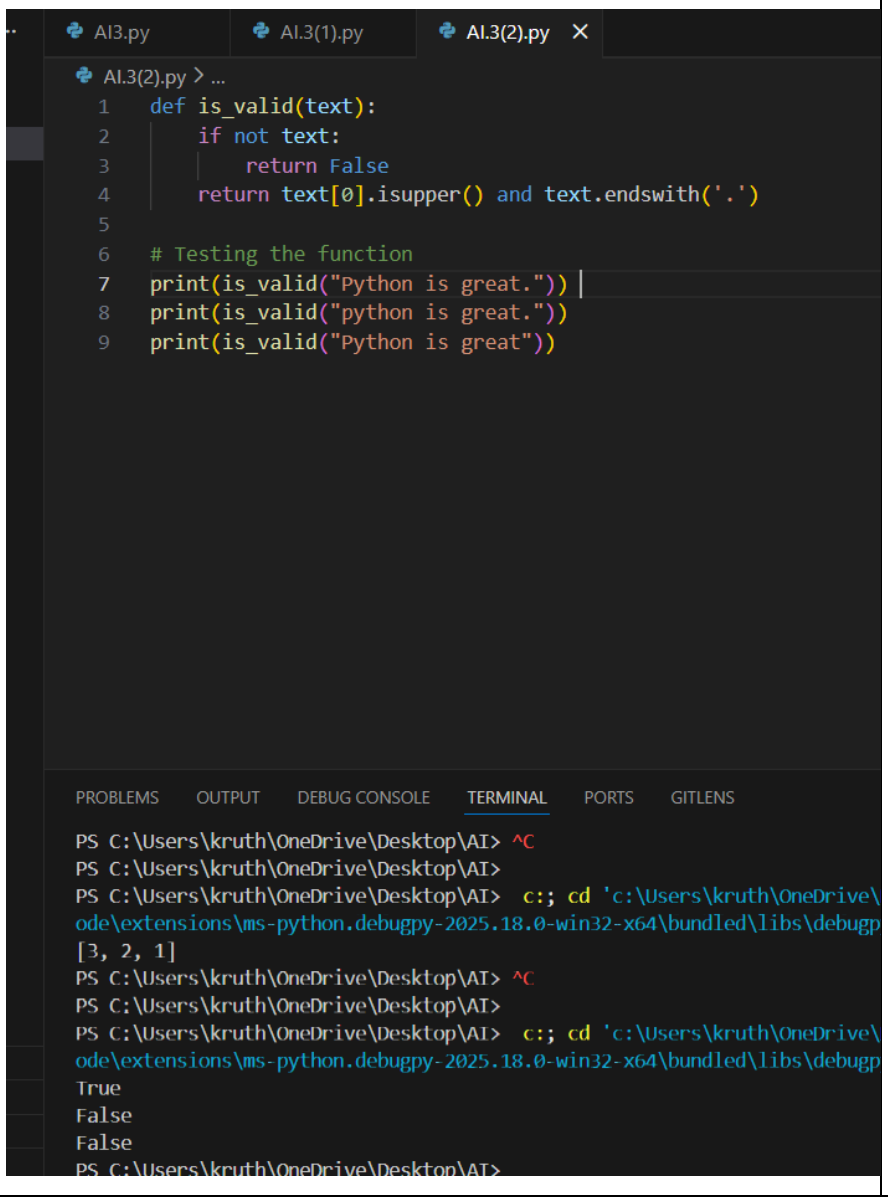
• Write a comment with 2–3 examples to help Copilot understand how to check if a string starts with a capital letter and ends with a period.

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.



	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.	

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?	

AI3.py | AI.3(1).py | AI.3(2).py | AI.3

AI.3(4).py > ...

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS G

```
PS C:\Users\kruth\OneDrive\Desktop\AI>
PS C:\Users\kruth\OneDrive\Desktop\AI>  c:; cd 'c:\User
ode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bu
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:\User
ode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bu
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:\User
ode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bu
6
PS C:\Users\kruth\OneDrive\Desktop\AI>
```

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	