AI ASSISTED CODING

NAME:SANIYA
ROLL NO:2403A510E7
ASSIGNMENT:5.1

Task Description #1 (Privacy in API Usage)

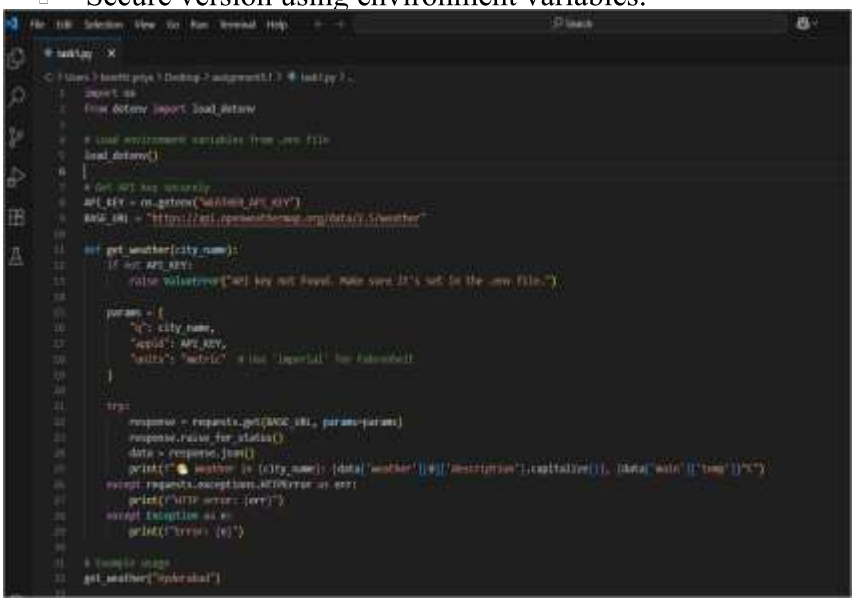
Task: Use an AI tool to generate a Python program that connects to a weather API.

Prompt:

"Generate code to fetch weather data securely without exposing API keys in the code."

Expected Output:

- Original AI code (check if keys are hardcoded).
- Secure version using environment variables.



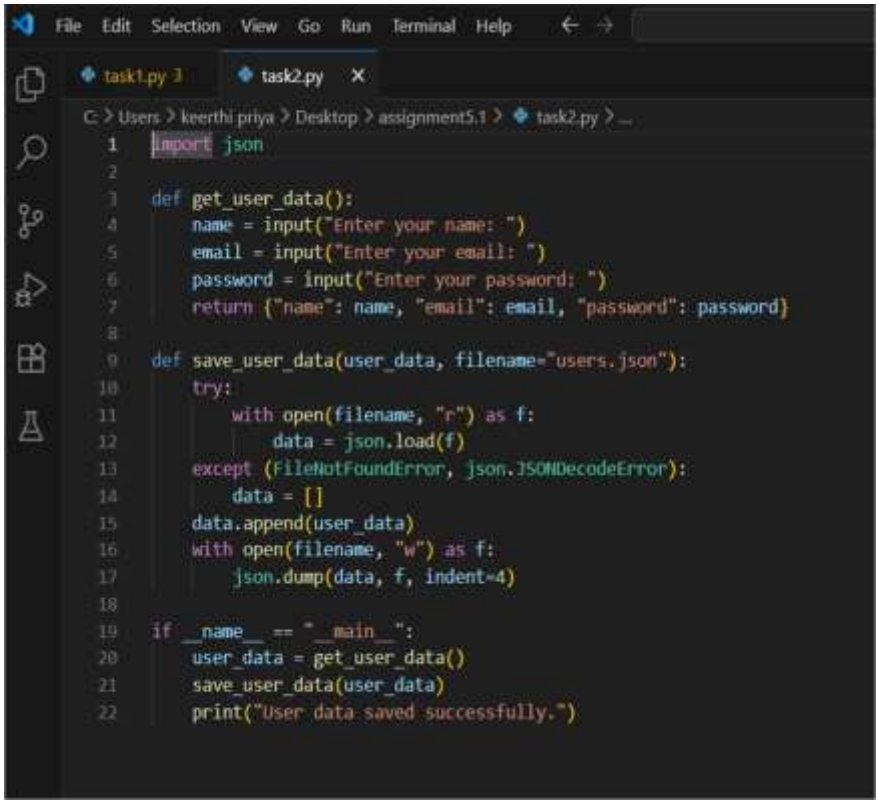
Task Description #2 (Privacy & Security in File Handling)

Task: Use an AI tool to generate a Python script that stores user data (name, email, password) in a file.

Analyze: Check if the AI stores sensitive data in plain text or without encryption.

Expected Output:

- Identified privacy risks.
- Revised version with encrypted password storage (e.g., hashing).



Task Description #3 (Transparency in Algorithm Design)

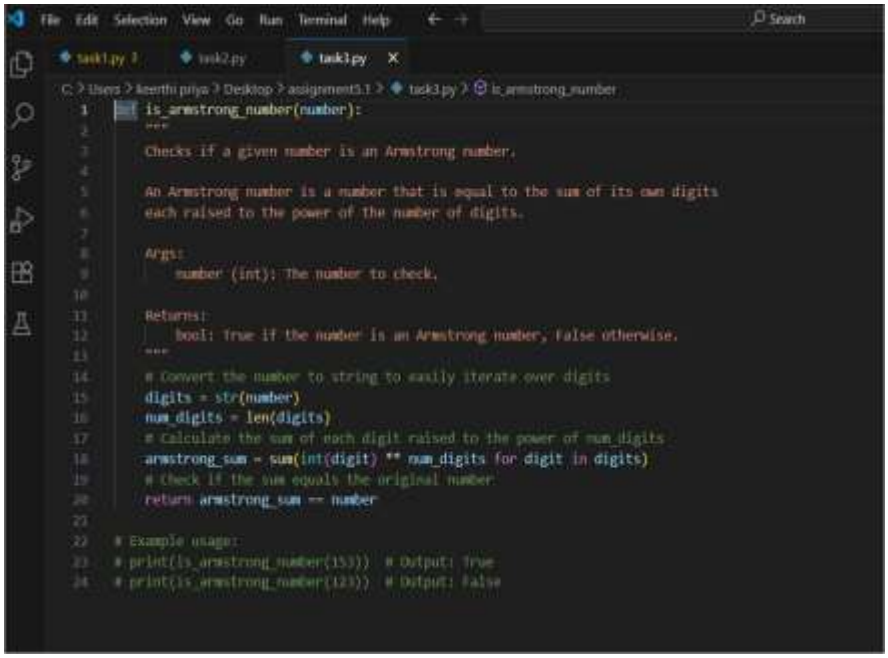
Objective: Use AI to generate an Armstrong number checking function with comments and explanations.

Instructions:

1. Ask AI to explain the code line-by-line.
2. Compare the explanation with code functionality.

Expected Output:

- Transparent, commented code.
- Correct, easy-to-understand explanation.



Task Description #4 (Transparency in Algorithm Comparison)

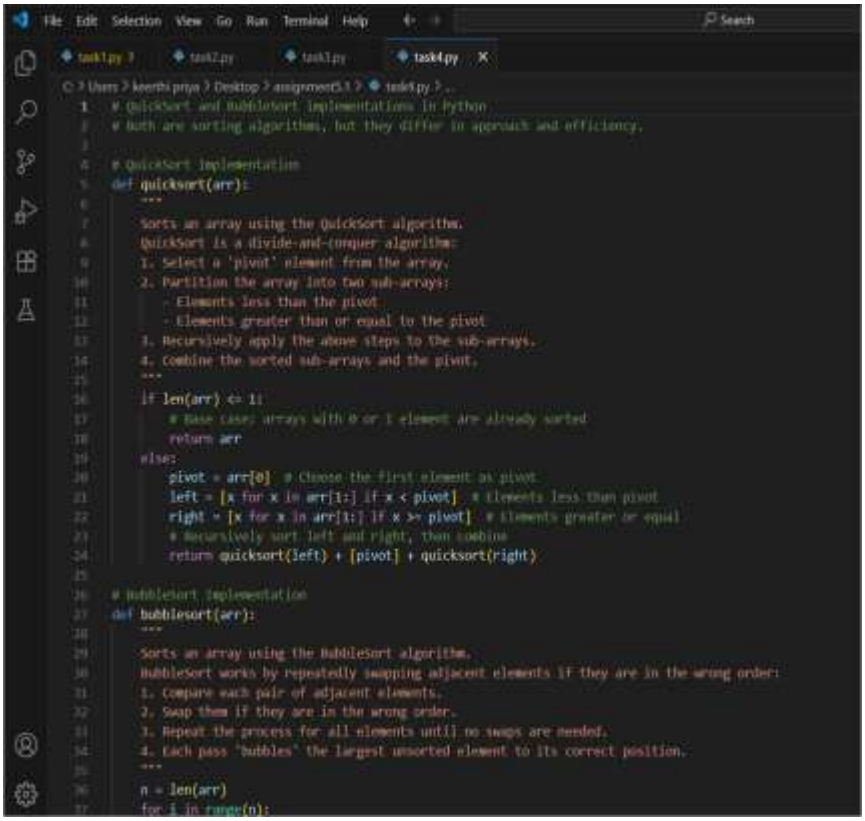
Task: Use AI to implement two sorting algorithms (e.g., QuickSort and BubbleSort).

Prompt:

"Generate Python code for QuickSort and BubbleSort, and include comments explaining step-by-step how each works and where they differ."

Expected Output:

- Code for both algorithms.
- Transparent, comparative explanation of their logic and efficiency.



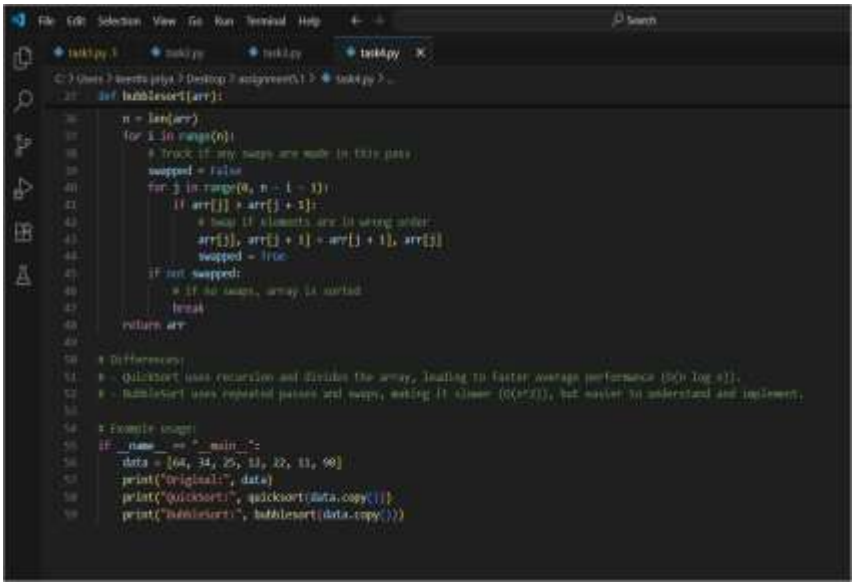
```python
# QuickSort and BubbleSort implementations in Python
# Both are sorting algorithms, but they differ in approach and efficiency.

# QuickSort implementation
def quicksort(arr):
    """
    Sorts an array using the QuickSort algorithm.
    QuickSort is a divide-and-conquer algorithm:
    1. Select a 'pivot' element from the array.
    2. Partition the array into two sub-arrays:
       - Elements less than the pivot
       - Elements greater than or equal to the pivot
    3. Recursively apply the above steps to the sub-arrays.
    4. Combine the sorted sub-arrays and the pivot.
    """
    if len(arr) <= 1:
        # Base case: arrays with 0 or 1 element are already sorted
        return arr
    else:
        pivot = arr[0]  # Choose the first element as pivot
        left = [x for x in arr[1:] if x < pivot]  # Elements less than pivot
        right = [x for x in arr[1:] if x >= pivot]  # Elements greater or equal
        # Recursively sort left and right, then combine
        return quicksort(left) + [pivot] + quicksort(right)

# BubbleSort implementation
def bubblesort(arr):
    """
    Sorts an array using the BubbleSort algorithm.
    BubbleSort works by repeatedly swapping adjacent elements if they are in the wrong order:
    1. Compare each pair of adjacent elements.
    2. Swap them if they are in the wrong order.
    3. Repeat the process for all elements until no swaps are needed.
    4. Each pass 'bubbles' the largest unsorted element to its correct position.
    """
    n = len(arr)
    for i in range(n):
        # Track if any swaps are made in this pass
        swapped = False
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                # Swap if elements are in wrong order
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:
            # If no swaps, array is sorted
            break
    return arr

# Differences:
# - QuickSort uses recursion and divides the array, leading to faster average performance (O(n log n)).
# - BubbleSort uses repeated passes and swaps, making it slower (O(n^2)), but easier to understand and implement.

# Example usage:
if __name__ == "__main__":
    data = [64, 34, 25, 12, 22, 11, 90]
    print("Original:", data)
    print("QuickSort:", quicksort(data.copy()))
    print("BubbleSort:", bubblesort(data.copy()))
```



```python
def bubblesort(arr):
    n = len(arr)
    for i in range(n):
        # Track if any swaps are made in this pass
        swapped = False
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                # Swap if elements are in wrong order
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        if not swapped:
            # If no swaps, array is sorted
            break
    return arr

# Differences:
# - QuickSort uses recursion and divides the array, leading to faster average performance (O(n log n)).
# - BubbleSort uses repeated passes and swaps, making it slower (O(n^2)), but easier to understand and implement.

# Example usage:
if __name__ == "__main__":
    data = [64, 34, 25, 12, 22, 11, 90]
    print("Original:", data)
    print("QuickSort:", quicksort(data.copy()))
    print("BubbleSort:", bubblesort(data.copy()))
```

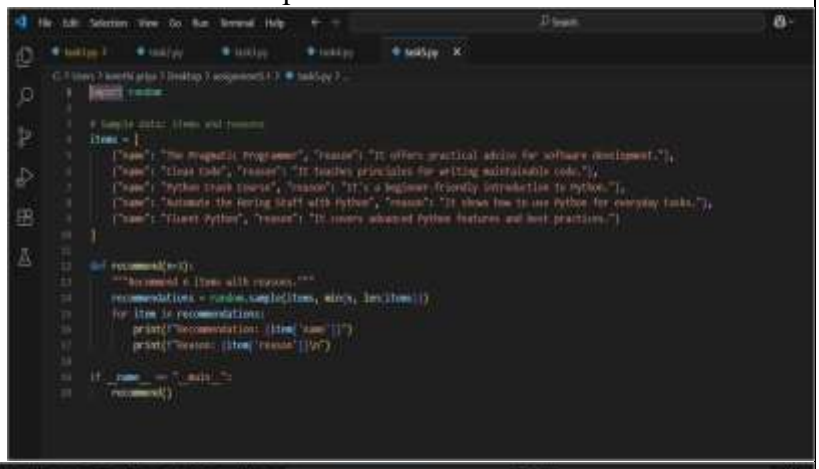
Task Description #5 (Transparency in AI Recommendations) Task: Use AI to create a product recommendation system.

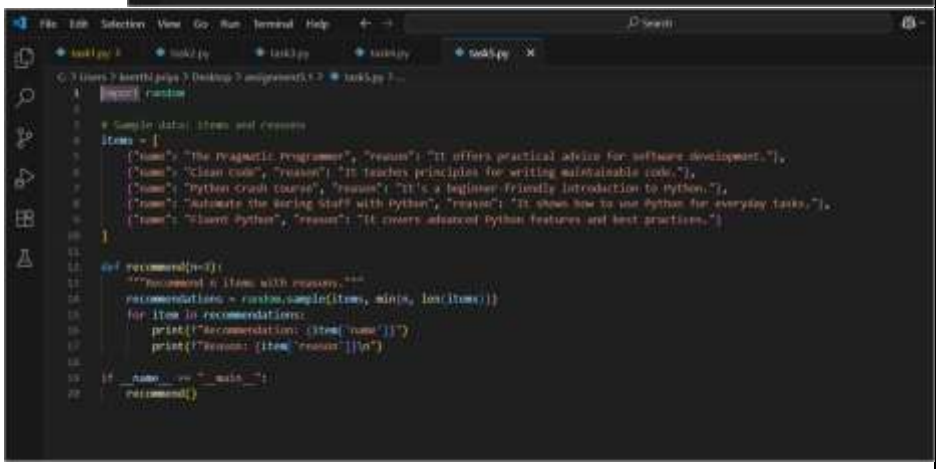
Prompt:

"Generate a recommendation system that also provides reasons for each suggestion." Expected Output:

- Code with explainable recommendations.

- Evaluation of whether explanations are understandable.





Task Description #6 (Transparent Code Generation)

Task: Ask AI to generate a Python function for calculating factorial using recursion.

Prompt:

"Generate a recursive factorial function with comments that explain each line and a final summary of the algorithm's flow." Expected Output:

- Fully commented code.
- Clear documentation of how recursion works. Task Description #7 (Inclusiveness in Customer Support) Code Snippet:

```python
def support_reply(name, gender):
    if gender.lower() == "male":
        prefix = "Mr."
    else:
        prefix = "Mrs."
    return f"Dear {prefix} {name}, we have resolved your i
```

Task:

Regenerate the code so that support messages use neutral language (e.g., “Dear {name}”) and optionally accept preferred titles.

Expected Output:

- Neutral, user-friendly support responses.

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

Evaluation Criteria:

Criteria	Max Marks
Transparency	1
Inclusiveness	0.5
Data security and Privacy	1
Total	2.5 Marks