AI ASSISTED CODING

NAME:Bayana Arjun
ROLL NO:2403A510A9
ASSIGNMENT:5.1

Task Description #1 (Privacy in API Usage)

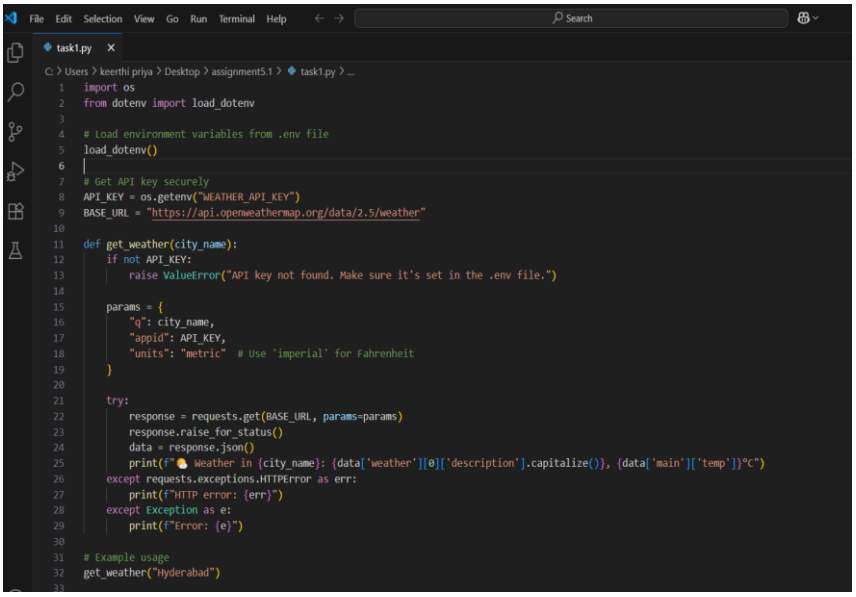
Task: Use an AI tool to generate a Python program that connects to a weather API.

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.



```python
import os
from dotenv import load_dotenv

# Load environment variables from .env file
load_dotenv()

# Get API key securely
API_KEY = os.getenv("WEATHER_API_KEY")
BASE_URL = "https://api.openweathermap.org/data/2.5/weather"

def get_weather(city_name):
    if not API_KEY:
        raise ValueError("API key not found. Make sure it's set in the .env file.")

    params = {
        "q": city_name,
        "appid": API_KEY,
        "units": "metric"  # Use 'imperial' for Fahrenheit
    }

    try:
        response = requests.get(BASE_URL, params=params)
        response.raise_for_status()
        data = response.json()
        print(f"Weather in {city_name}: {data['weather'][0]['description'].capitalize()}, {data['main']['temp']}°C")
    except requests.exceptions.HTTPError as err:
        print(f"HTTP error: {err}")
    except Exception as e:
        print(f"Error: {e}")

# Example usage
get_weather("Hyderabad")
```

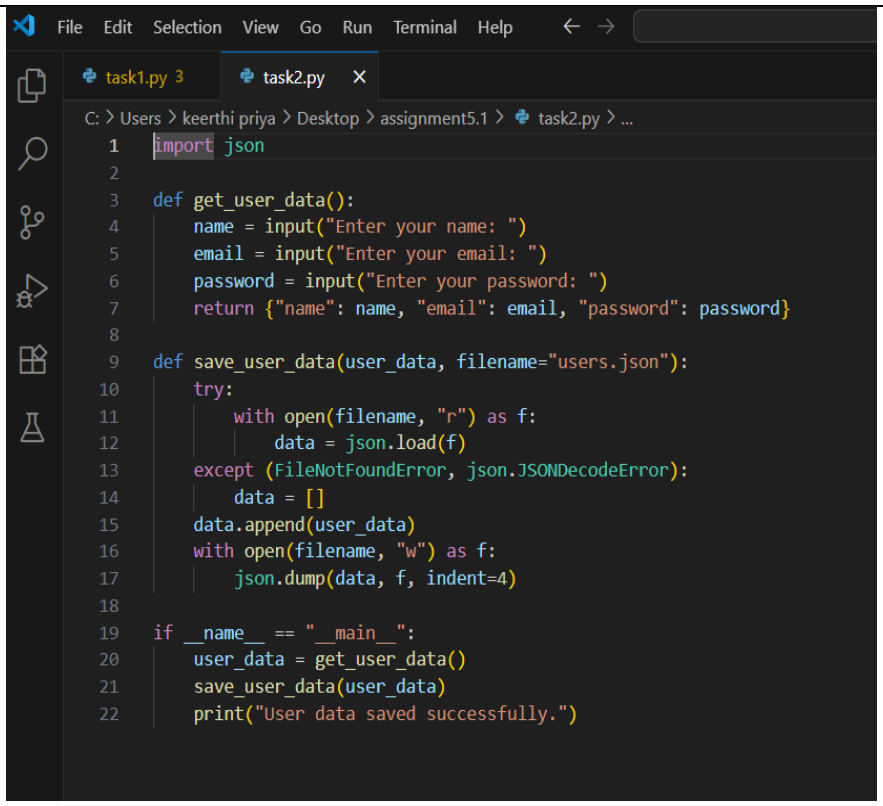
Task Description #2 (Privacy & Security in File Handling)

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)

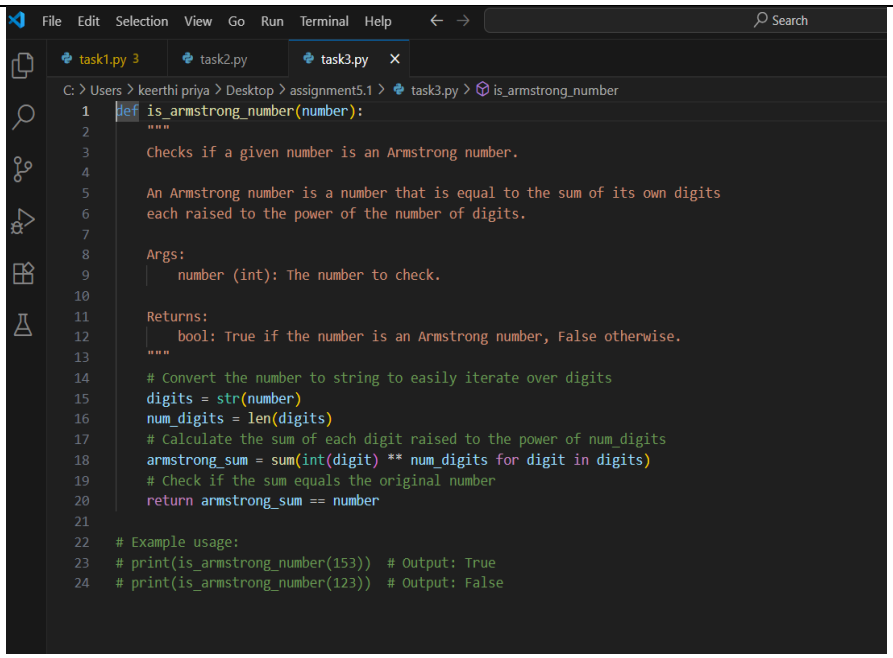
Objective: Use AI to generate an Armstrong number checking function with comments and explanations.

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)

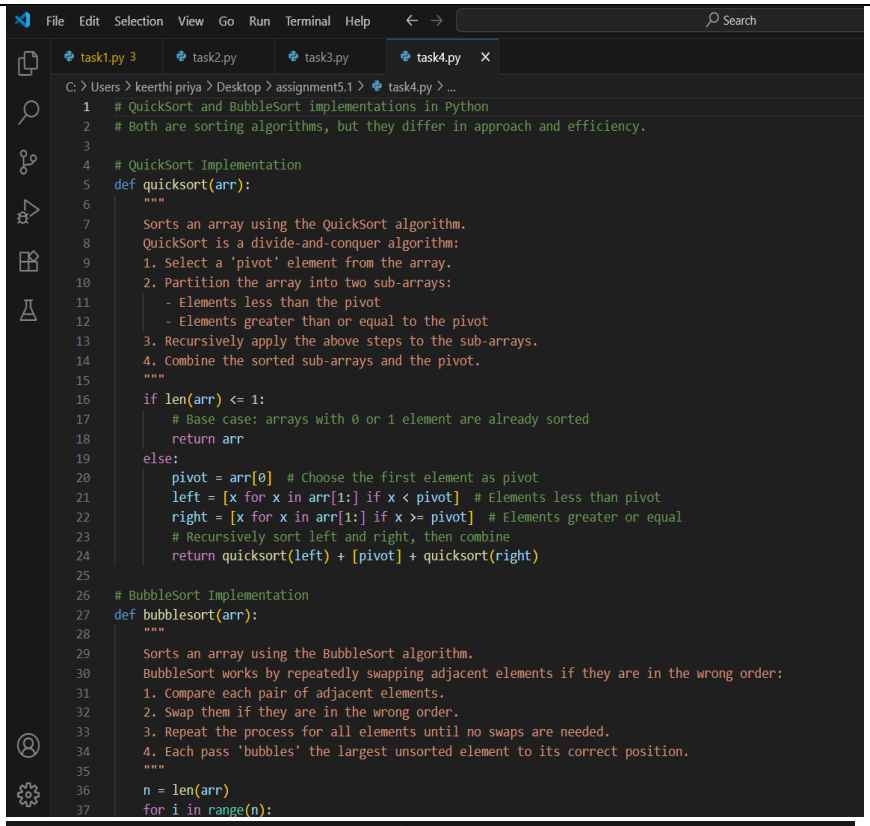
Task: Use AI to implement two sorting algorithms (e.g., QuickSort and BubbleSort).

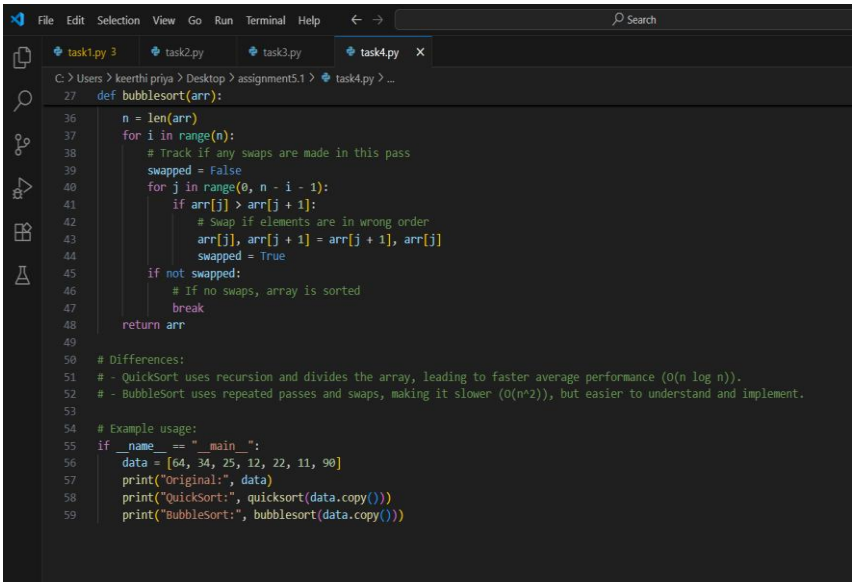
Prompt:

"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.

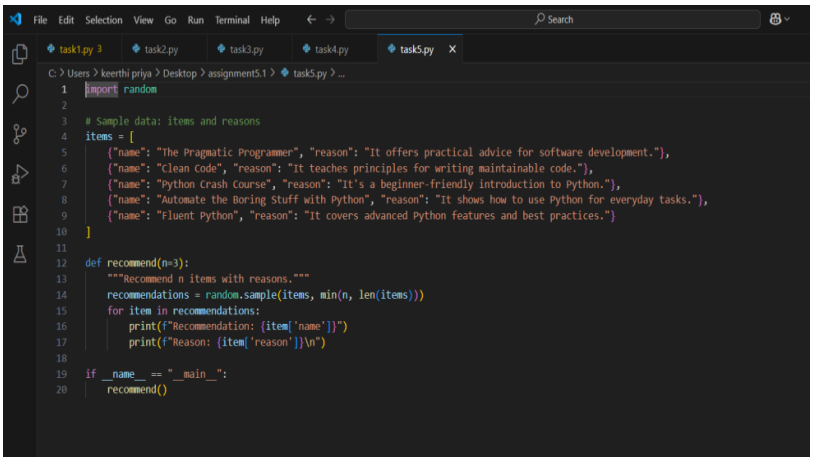


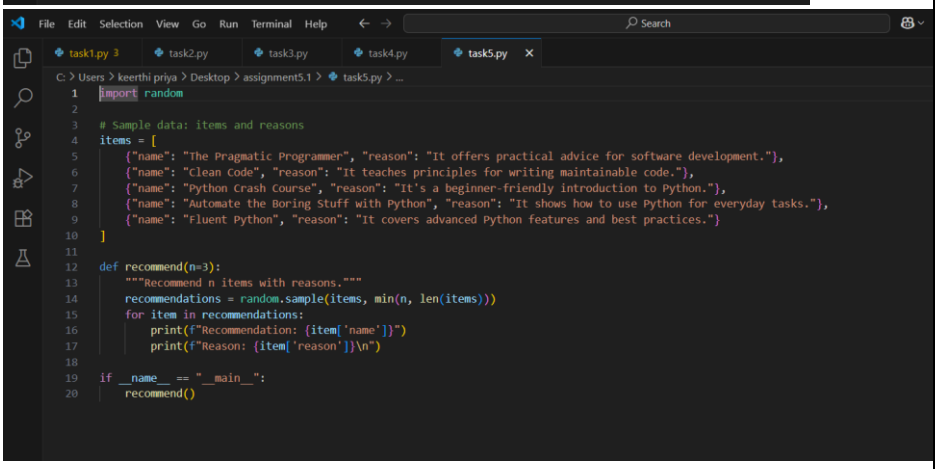


Task Description #5 (Transparency in AI Recommendations)
Task: Use AI to create a product recommendation system.
Prompt:
"Generate a recommendation system that also provides reasons for each suggestion."
Expected Output:

- 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:

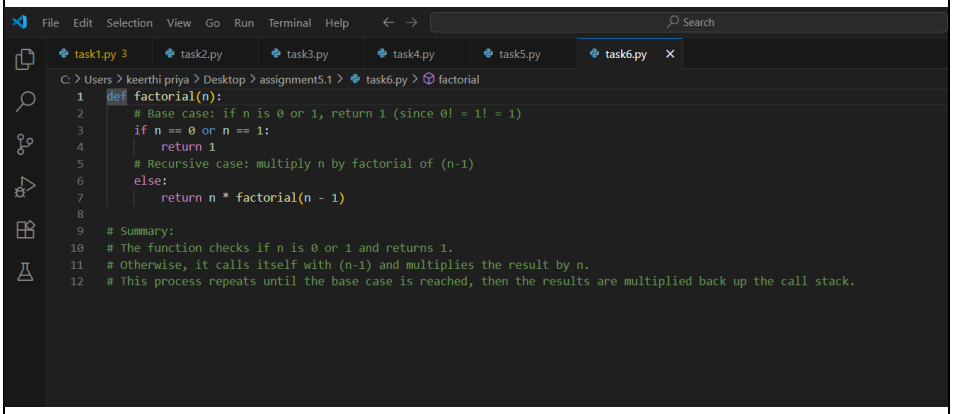
```python
def support_reply(name, gender):
    if gender.lower() == "male":
        prefix = "Mr."
    else:
        prefix = "Mrs."
    return f"Dear {prefix} {name}, we have resolved your i
```

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