Assignment Number: 5.1

Name : **v.sushmitha**

Batch: 23CSBTB47B

Hall Ticket No : 2303A54055

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Task Description #1:: (Privacy in API Usage)

Task: An application needs to fetch weather information from an online API. To prevent misuse, the API key must not be exposed directly in the source code.

Scenario:
An application needs to store user details such as name, email, and password in a file. To protect sensitive information, the password must be stored securely instead of plain text.

Prompt:

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

Expected Output:

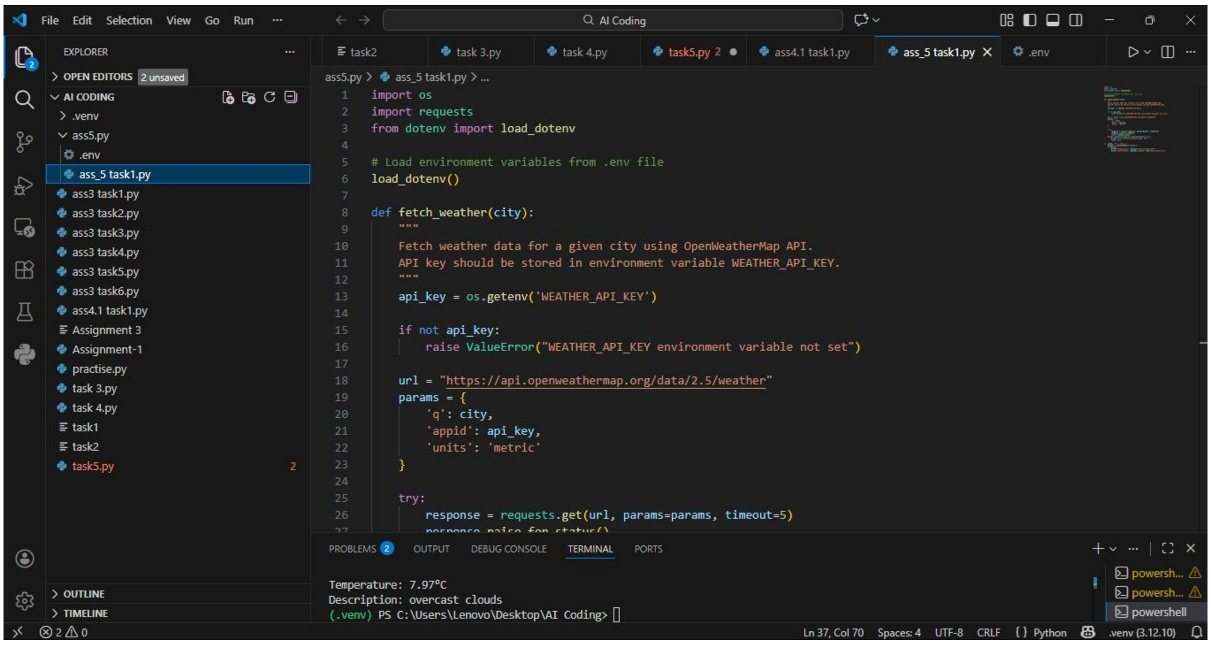
Enter city name: Hyderabad

Sample Output:

```
{'weather': [{'main': 'Clear'}], 'main': {'temp': 303.15}}
```

Explanation : The API key is taken from an environment variable instead of writing it in the code

This helps accidental exposure and improves security while accessing the weather API



```python
import os
import requests
from dotenv import load_dotenv

# Load environment variables from .env file
load_dotenv()

def fetch_weather(city):
    """
    Fetch weather data for a given city using OpenWeatherMap API.
    API key should be stored in environment variable WEATHER_API_KEY.
    """
    api_key = os.getenv('WEATHER_API_KEY')

    if not api_key:
        raise ValueError("WEATHER_API_KEY environment variable not set")

    url = "https://api.openweathermap.org/data/2.5/weather"
    params = {
        'q': city,
        'appid': api_key,
        'units': 'metric'
    }

    try:
        response = requests.get(url, params=params, timeout=5)
```

```
Temperature: 7.97°C
Description: overcast clouds
(.venv) PS C:\Users\Lenovo\Desktop\AI Coding>
```

Task Description #2 – Privacy & Security in File Handling

Task / Scenario

An application needs to store user details such as name, email, and password in a file.
Since passwords are sensitive information, storing them directly can cause security risks.

Prompt Used:

Generate a Python script to store user details and modify it to store passwords securely using hashing instead of plain text.

Sample Input:

Enter name: Sushmitha

Enter email: sushmitha23@gmail.com

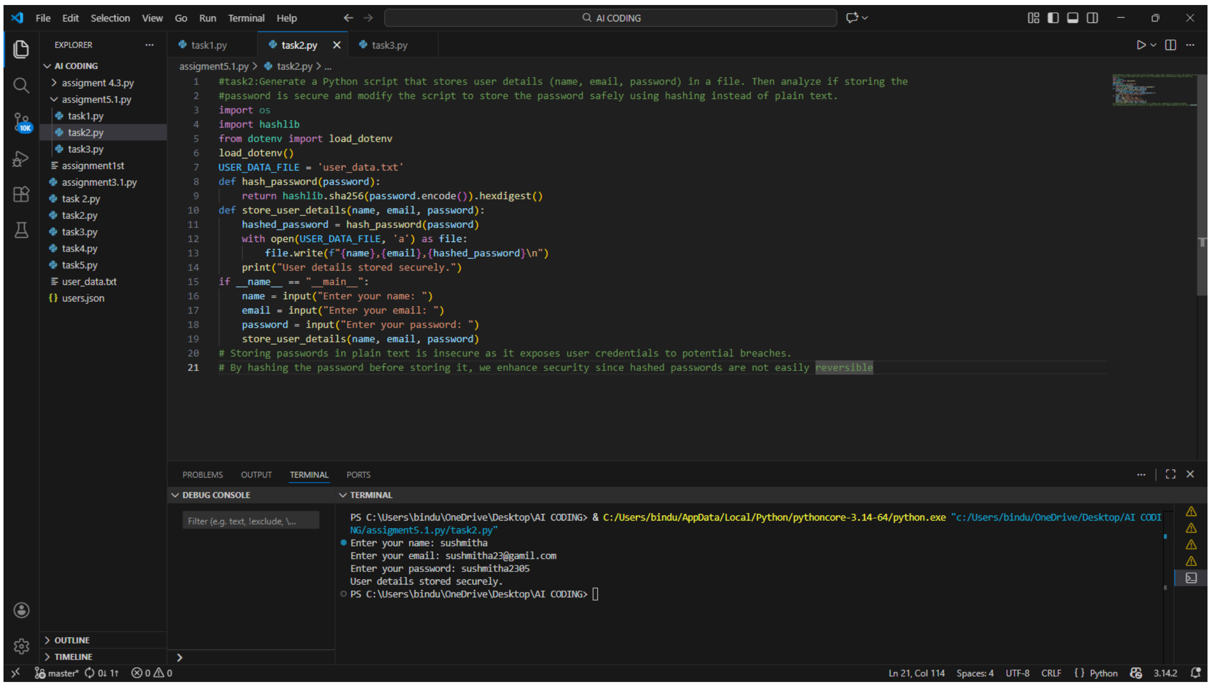
Enter password: sushmitha2305

Sample Output :

Sushmitha,sushmitha@gmail.com,mypass123

Explanation : Instead of saving the password directly, it is coverted into a hash

This ensures the real password cannot be read even if the file is accessed


```python
#task2:Generate a Python script that stores user details (name, email, password) in a file. Then analyze if storing the
#password is secure and modify the script to store the password safely using hashing instead of plain text.
import os
import hashlib
from dotenv import load_dotenv
load_dotenv()
USER_DATA_FILE = 'user_data.txt'
def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()
def store_user_details(name, email, password):
    hashed_password = hash_password(password)
    with open(USER_DATA_FILE, 'a') as file:
        file.write(f"{name},{email},{hashed_password}\n")
    print("User details stored securely.")
if __name__ == "__main__":
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    password = input("Enter your password: ")
    store_user_details(name, email, password)
# Storing passwords in plain text is insecure as it exposes user credentials to potential breaches.
# By hashing the password before storing it, we enhance security since hashed passwords are not easily reversible
```

```
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODI
NG/assigment5.1.py/task2.py"
Enter your name: sushmitha
Enter your email: sushmitha23@gamil.com
Enter your password: sushmitha2305
User details stored securely.
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> 
```

Task Description #3 – Transparency in Algorithm Design

Task / Scenario:

A program is required to check whether a given number is an Armstrong number.

The logic should be clear and understandable to ensure transparency.

Prompt Used:

Generate an Armstrong number checking function with comments and explain the code line by line.

Sample Input

153

370

9474

123

Sample Output

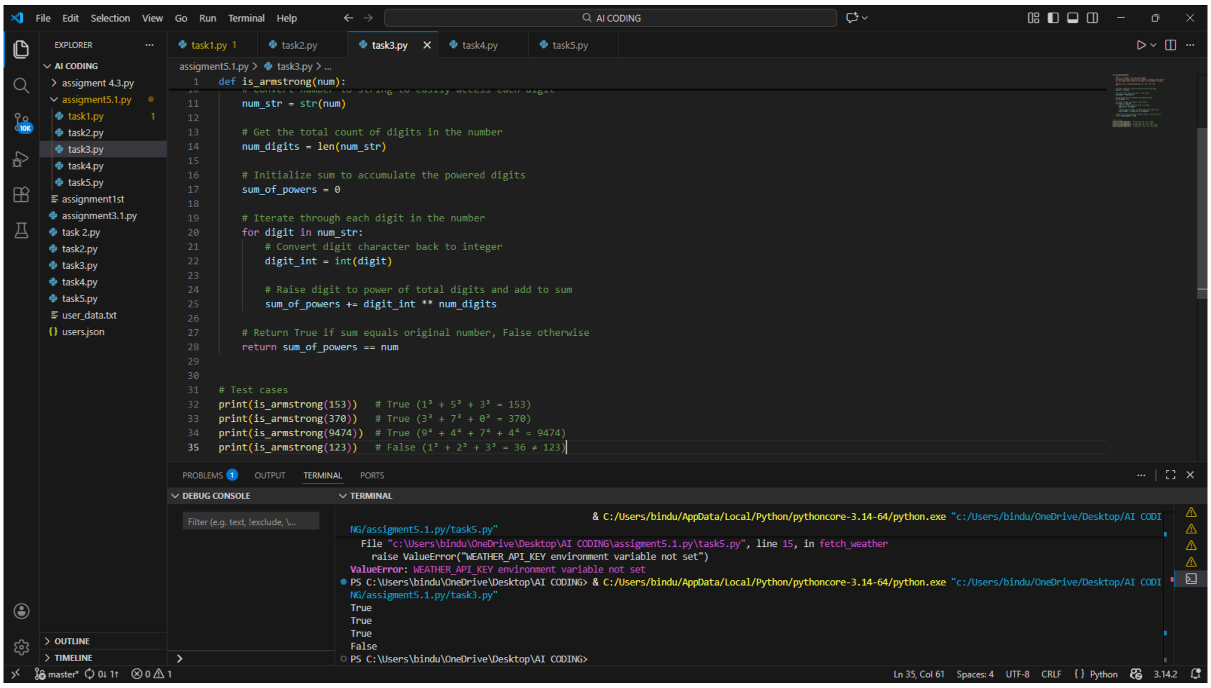
True

True

True

False

Explanation: The program converts the number into digits and counts how many digits it has.
Each digit is raised to the power of the total digit count and added together.
If the final sum matches the original number, it is identified as an Armstrong number.



```python
def is_armstrong(num):
    num_str = str(num)

    # Get the total count of digits in the number
    num_digits = len(num_str)

    # Initialize sum to accumulate the powered digits
    sum_of_powers = 0

    # Iterate through each digit in the number
    for digit in num_str:
        # Convert digit character back to integer
        digit_int = int(digit)

        # Raise digit to power of total digits and add to sum
        sum_of_powers += digit_int ** num_digits

    # Return True if sum equals original number, False otherwise
    return sum_of_powers == num


# Test cases
print(is_armstrong(153))   # True (1³ + 5³ + 3³ = 153)
print(is_armstrong(370))   # True (3³ + 7³ + 0³ = 370)
print(is_armstrong(9474))  # True (9⁴ + 4⁴ + 7⁴ + 4⁴ = 9474)
print(is_armstrong(123))   # False (1³ + 2³ + 3³ = 36 ≠ 123)
```

```
                                   & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODI
NG/assigment5.1.py/task5.py"
  File "c:\Users\bindu\OneDrive\Desktop\AI CODING\assigment5.1.py\task5.py", line 15, in fetch_weather
    raise ValueError("WEATHER_API_KEY environment variable not set")
ValueError: WEATHER_API_KEY environment variable not set
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODI
NG/assigment5.1.py/task3.py"
True
True
True
False
PS C:\Users\bindu\OneDrive\Desktop\AI CODING>
```

Task Description #4 – Transparency in Algorithm Comparison

Task / Scenario

Sorting is a common operation in many applications.
Different sorting algorithms have different performance and efficiency.
This task compares Bubble Sort and Quick Sort to understand their working and differences

Prompt Used

Generate Python code for Bubble Sort and Quick Sort with step-by-step comments and compare their logic and efficiency

Sample Input:

[64, 34, 25, 12, 22, 11, 90]

Sample Output

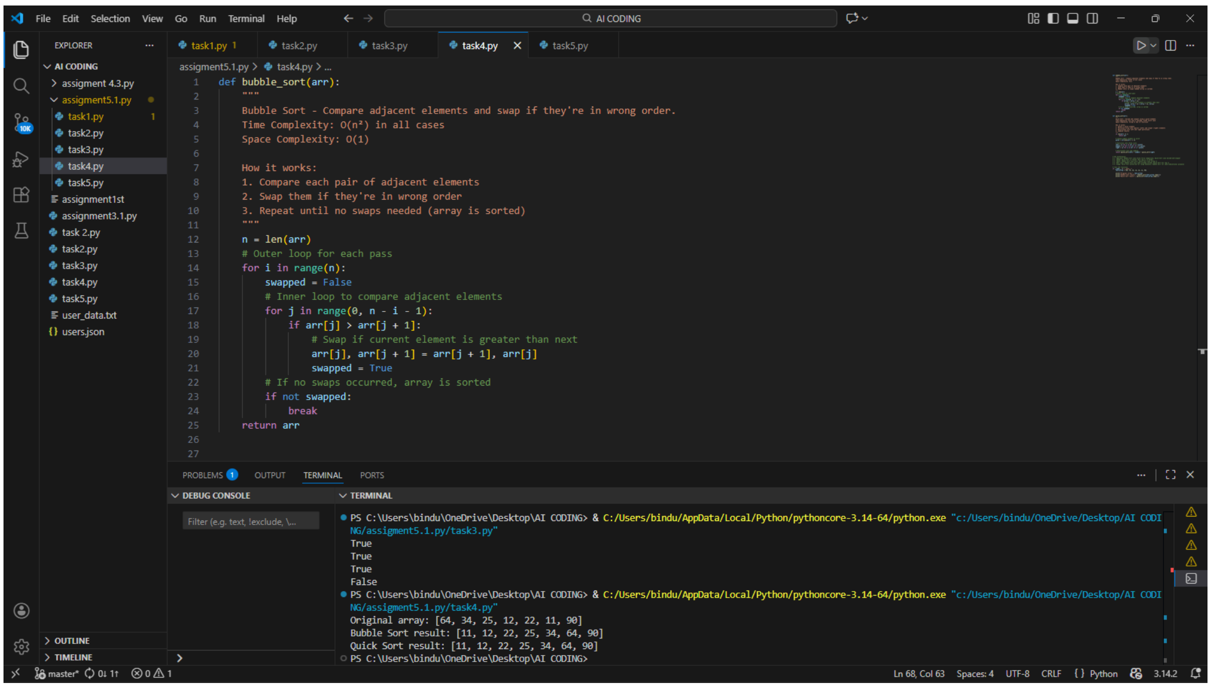
Original array: [64, 34, 25, 12, 22, 11, 90]

Bubble Sort result: [11, 12, 22, 25, 34, 64, 90]

Quick Sort result: [11, 12, 22, 25, 34, 64, 90]

Explanation:

Bubble Sort works by repeatedly comparing adjacent elements and swapping them until the list is sorted, which makes it slow for large data.
Quick Sort works by selecting a pivot element, dividing the list into smaller parts, and sorting them recursively, making it faster and more efficient.



```python
def bubble_sort(arr):
    """
    Bubble Sort - Compare adjacent elements and swap if they're in wrong order.
    Time Complexity: O(n²) in all cases
    Space Complexity: O(1)

    How it works:
    1. Compare each pair of adjacent elements
    2. Swap them if they're in wrong order
    3. Repeat until no swaps needed (array is sorted)
    """
    n = len(arr)
    # Outer loop for each pass
    for i in range(n):
        swapped = False
        # Inner loop to compare adjacent elements
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                # Swap if current element is greater than next
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True
        # If no swaps occurred, array is sorted
        if not swapped:
            break
    return arr
```

```
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODI
NG/assigment5.1.py/task3.py"
True
True
True
False
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODI
NG/assigment5.1.py/task4.py"
Original array: [64, 34, 25, 12, 22, 11, 90]
Bubble Sort result: [11, 12, 22, 25, 34, 64, 90]
Quick Sort result: [11, 12, 22, 25, 34, 64, 90]
PS C:\Users\bindu\OneDrive\Desktop\AI CODING>
```

Task Description #5 – Transparency in AI Recommendations

Task / Scenario

A recommendation system is used to suggest items based on user preferences.
To ensure transparency, the system should also explain why a particular recommendation is made.

Prompt Used

Generate a simple Python-based recommendation system that provides recommendations along with clear reasons for each suggestion.

Sample Input

```
Enter your favorite genre: Action
```

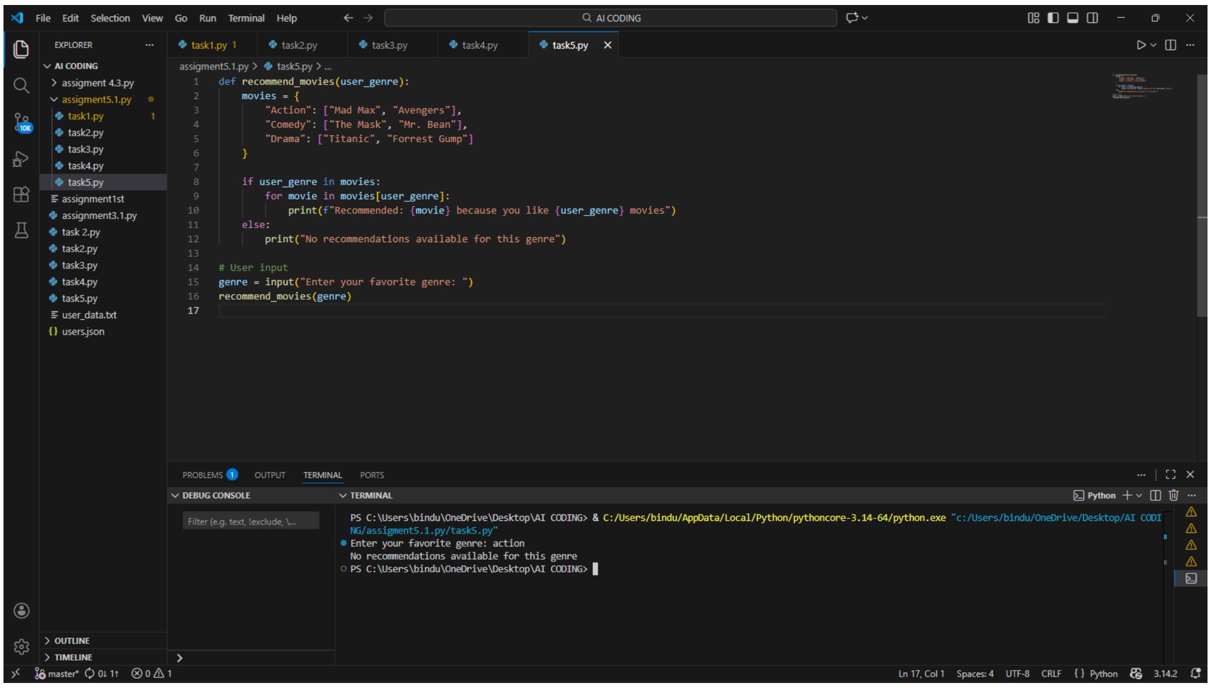
Sample Output

```
Recommended: Mad Max because you like Action movies

Recommended: Avengers because you like Action movies
```

Explanation

The system checks the user's preferred genre and matches it with a predefined list of movies.
Each recommendation includes a reason, making the system transparent and easy to understand.



```python
def recommend_movies(user_genre):
    movies = {
        "Action": ["Mad Max", "Avengers"],
        "Comedy": ["The Mask", "Mr. Bean"],
        "Drama": ["Titanic", "Forrest Gump"]
    }

    if user_genre in movies:
        for movie in movies[user_genre]:
            print(f"Recommended: {movie} because you like {user_genre} movies")
    else:
        print("No recommendations available for this genre")

# User input
genre = input("Enter your favorite genre: ")
recommend_movies(genre)
```

```
PS C:\Users\bindu\OneDrive\Desktop\AI CODING> & C:/Users/bindu/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/bindu/OneDrive/Desktop/AI CODI
NG/assigment5.1.py/task5.py"
Enter your favorite genre: action
No recommendations available for this genre
PS C:\Users\bindu\OneDrive\Desktop\AI CODING>
```