Lab 5 – Ethical Foundations: Responsible AI Coding Practices

Name: Mohammad Muneer Ahmed

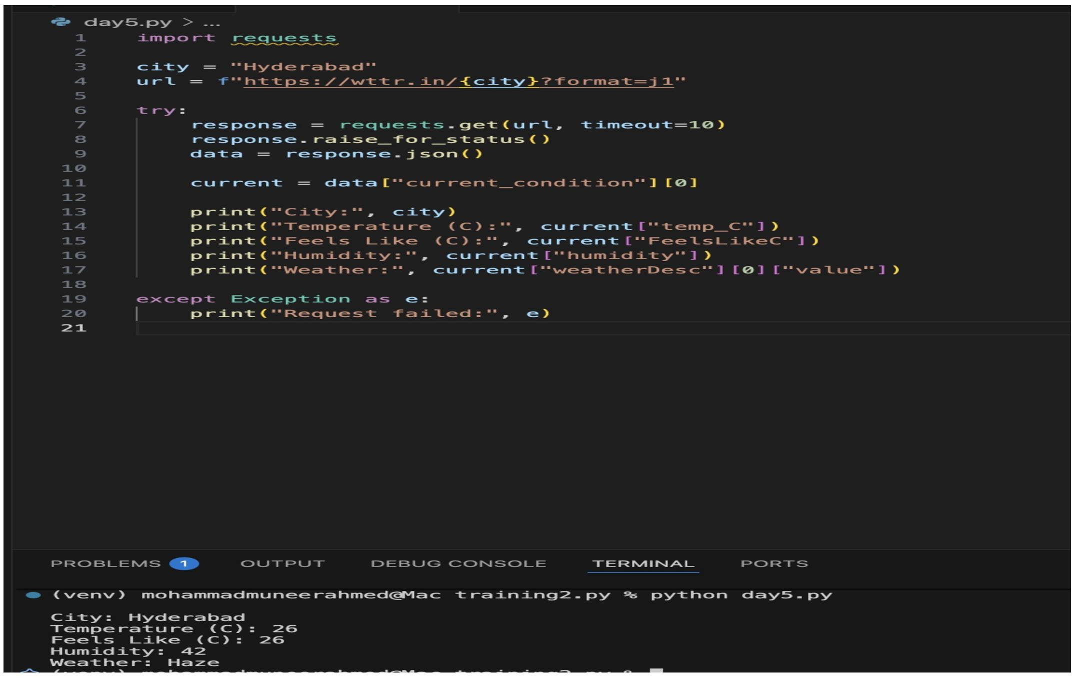
Roll No: 2303A51475

Task 1 – Privacy in API Usage (Weather API)

Question: Generate a Python program to fetch weather data securely without exposing API keys.

Algorithm:
1. Select city name.
2. Build weather URL using public no■key endpoint.
3. Send HTTP GET request.
4. Convert response to JSON.
5. Extract current condition fields.
6. Print temperature, humidity, and description.

Pseudocode:
```
START
SET city
SET url using city
response ← GET(url)
data ← JSON(response)
current ← data.current_condition[0]
PRINT weather fields
END
```



day5.py > ...

```python
import requests

city = "Hyderabad"
url = f"https://wttr.in/{city}?format=j1"

try:
    response = requests.get(url, timeout=10)
    response.raise_for_status()
    data = response.json()

    current = data["current_condition"][0]

    print("City:", city)
    print("Temperature (C):", current["temp_C"])
    print("Feels Like (C):", current["FeelsLikeC"])
    print("Humidity:", current["humidity"])
    print("Weather:", current["weatherDesc"][0]["value"])

except Exception as e:
    print("Request failed:", e)
```

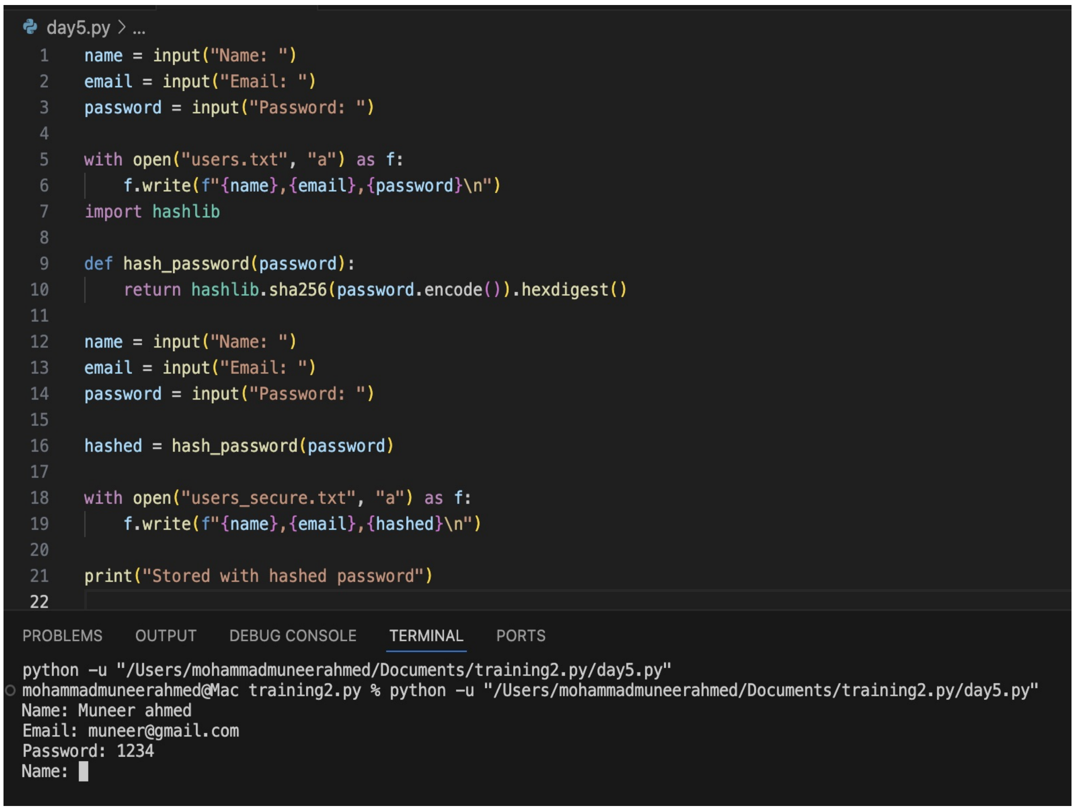
```
(venv) mohammadmuneerahmed@Mac training2.py % python day5.py
City: Hyderabad
Temperature (C): 26
Feels Like (C): 26
Humidity: 42
Weather: Haze
```

Task 2 – Privacy & Security in File Handling

Question: Store user data and avoid plain■text passwords using hashing.

Algorithm:
1. Read name, email, password.
2. Convert password to bytes.
3. Apply SHA■256 hashing.
4. Store name, email, hashed password to file.

Pseudocode:
```
START
INPUT name, email, password
hashed ← SHA256(password)
OPEN file
WRITE name,email,hashed
CLOSE file
END
```



day5.py > ...

```python
name = input("Name: ")
email = input("Email: ")
password = input("Password: ")

with open("users.txt", "a") as f:
    f.write(f"{name},{email},{password}\n")
import hashlib

def hash_password(password):
    return hashlib.sha256(password.encode()).hexdigest()

name = input("Name: ")
email = input("Email: ")
password = input("Password: ")

hashed = hash_password(password)

with open("users_secure.txt", "a") as f:
    f.write(f"{name},{email},{hashed}\n")

print("Stored with hashed password")

```

```
python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
mohammadmuneerahmed@Mac training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Name: Muneer ahmed
Email: muneer@gmail.com
Password: 1234
Name: 
```

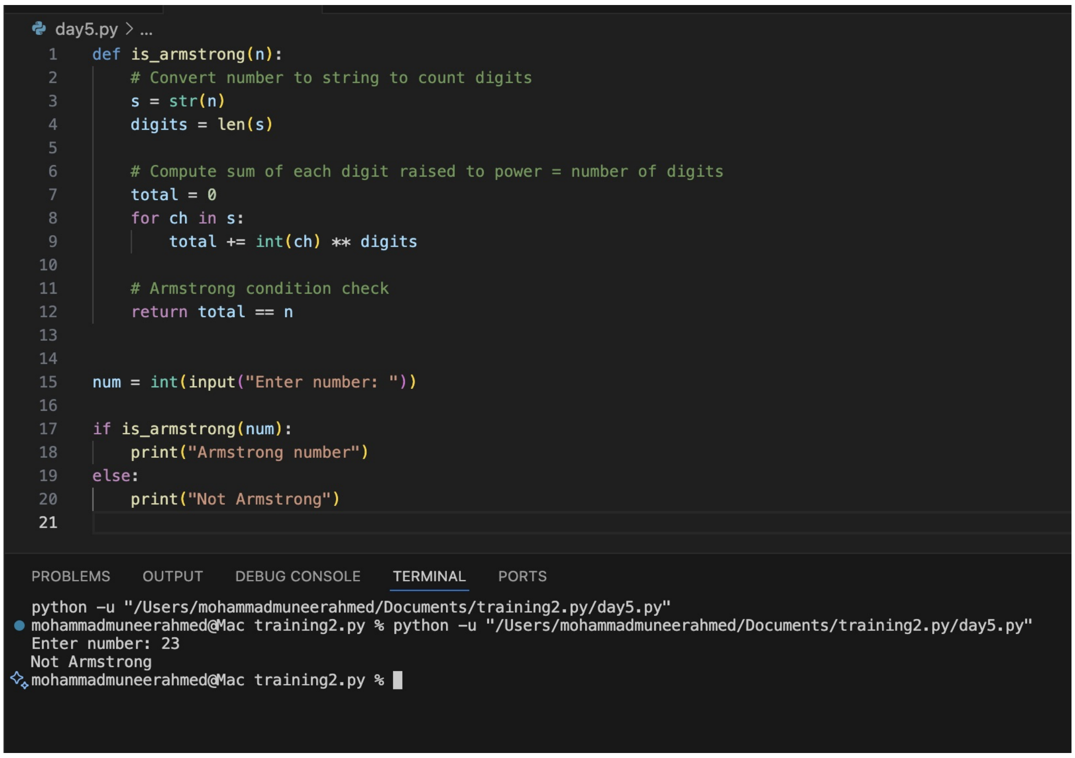
Task 3 – Armstrong Number Transparency

Question: Implement Armstrong number check with clear logic.

Algorithm:
1. Read number.
2. Convert number to string.
3. Count digits.
4. For each digit raise to digit count power.
5. Sum results.
6. Compare with original number.

Pseudocode:

```
START
INPUT n
digits ← length(n)
sum ← 0
FOR each digit d in n
sum ← sum + d^digits
IF sum = n PRINT Armstrong
ELSE PRINT Not Armstrong
END
```



```python
def is_armstrong(n):
    # Convert number to string to count digits
    s = str(n)
    digits = len(s)

    # Compute sum of each digit raised to power = number of digits
    total = 0
    for ch in s:
        total += int(ch) ** digits

    # Armstrong condition check
    return total == n


num = int(input("Enter number: "))

if is_armstrong(num):
    print("Armstrong number")
else:
    print("Not Armstrong")
```

```
python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
mohammadmuneerahmed@Mac training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Enter number: 23
Not Armstrong
mohammadmuneerahmed@Mac training2.py %
```

Task 4 – Sorting Algorithm Comparison

Question: Implement Bubble Sort and Quick Sort with explanation.

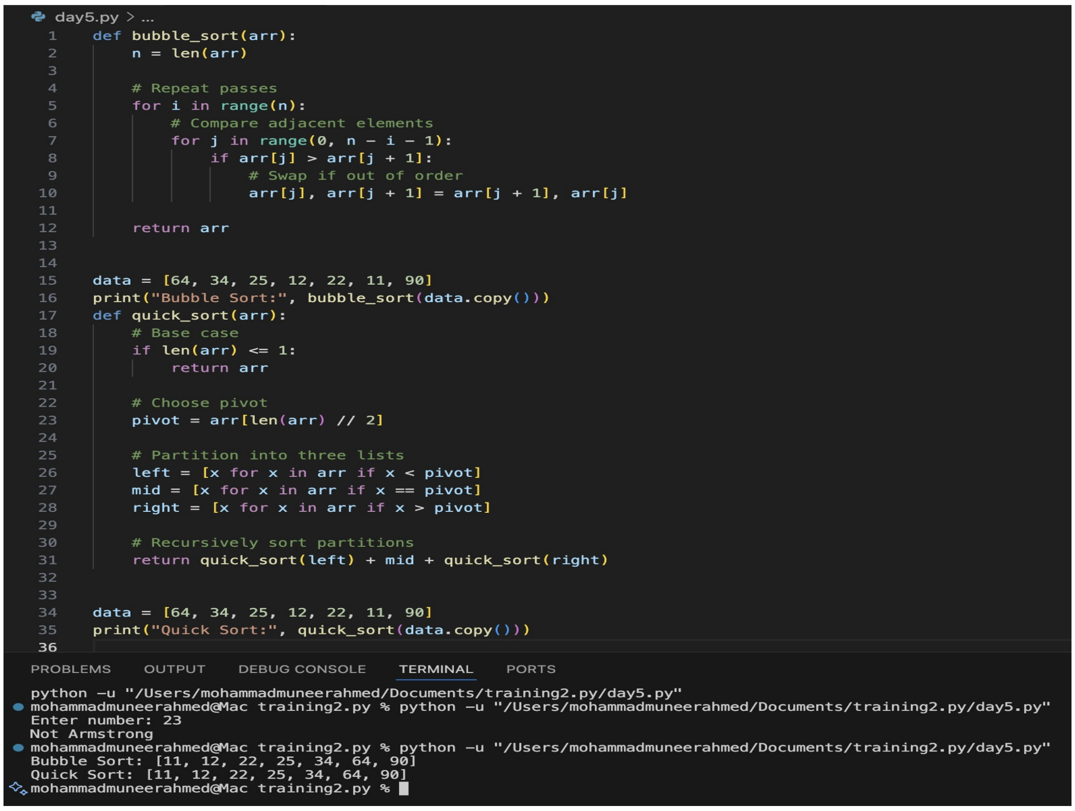
Algorithm:
Bubble Sort:
1. Repeat passes through list.
2. Compare adjacent elements.
3. Swap if out of order.

Quick Sort:
1. Choose pivot.
2. Split into left, equal, right.
3. Recursively sort parts.

Pseudocode:

```
BUBBLE:
FOR i passes
FOR j comparisons
IF a[j] > a[j+1] swap

QUICK:
IF size ≤ 1 return
pivot ← middle
left,right split
RETURN quick(left)+pivot+quick(right)
```


day5.py > ...

```python
def bubble_sort(arr):
    n = len(arr)

    # Repeat passes
    for i in range(n):
        # Compare adjacent elements
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                # Swap if out of order
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

    return arr


data = [64, 34, 25, 12, 22, 11, 90]
print("Bubble Sort:", bubble_sort(data.copy()))
def quick_sort(arr):
    # Base case
    if len(arr) <= 1:
        return arr

    # Choose pivot
    pivot = arr[len(arr) // 2]

    # Partition into three lists
    left = [x for x in arr if x < pivot]
    mid = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]

    # Recursively sort partitions
    return quick_sort(left) + mid + quick_sort(right)


data = [64, 34, 25, 12, 22, 11, 90]
print("Quick Sort:", quick_sort(data.copy()))
```

```
python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
mohammadmuneerahmed@Mac training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Enter number: 23
Not Armstrong
mohammadmuneerahmed@Mac training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Bubble Sort: [11, 12, 22, 25, 34, 64, 90]
Quick Sort: [11, 12, 22, 25, 34, 64, 90]
mohammadmuneerahmed@Mac training2.py %
```

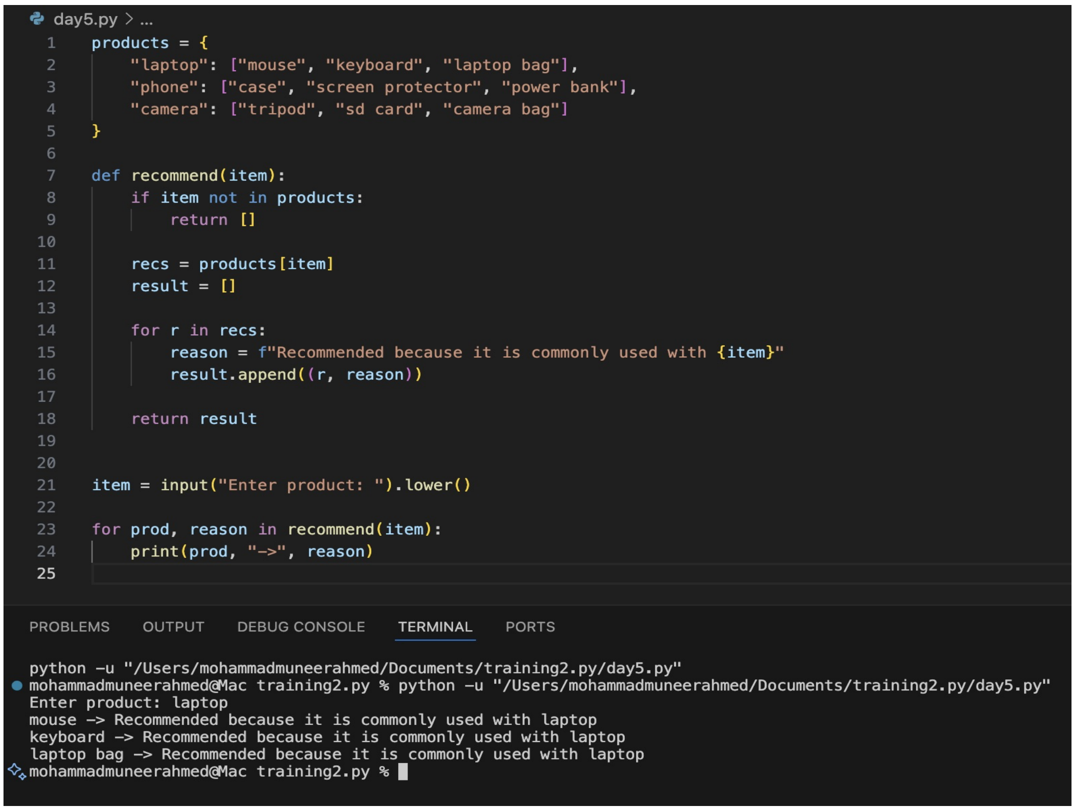
Task 5 – Explainable Recommendation System

Question: Build recommendation system with reasons.

Algorithm:
1. Store product → accessory map.
2. Read user product.
3. Lookup related items.
4. Attach explanation string.
5. Print recommendation with reason.

Pseudocode:

```
START
DEFINE product map
INPUT item
IF item exists
FOR each related product
PRINT product + reason
END
```



day5.py > ...

```python
products = {
    "laptop": ["mouse", "keyboard", "laptop bag"],
    "phone": ["case", "screen protector", "power bank"],
    "camera": ["tripod", "sd card", "camera bag"]
}

def recommend(item):
    if item not in products:
        return []

    recs = products[item]
    result = []

    for r in recs:
        reason = f"Recommended because it is commonly used with {item}"
        result.append((r, reason))

    return result


item = input("Enter product: ").lower()

for prod, reason in recommend(item):
    print(prod, "->", reason)

```

```
python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
mohammadmuneerahmed@Mac training2.py % python -u "/Users/mohammadmuneerahmed/Documents/training2.py/day5.py"
Enter product: laptop
mouse -> Recommended because it is commonly used with laptop
keyboard -> Recommended because it is commonly used with laptop
laptop bag -> Recommended because it is commonly used with laptop
mohammadmuneerahmed@Mac training2.py %
```