Name: P.Sai Venkatesh

H.No: 2303A53039 – B46

Assignment – 5.1

Lab 5: Ethical Foundations – Responsible AI Coding Practices

Lab Objectives:

- To explore the ethical risks associated with AI-generated code.
- To recognize issues related to security, bias, transparency, and copyright.
- To reflect on the responsibilities of developers when using AI tools in software development.
- To promote awareness of best practices for responsible and ethical AI coding.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Identify and avoid insecure coding patterns generated by AI tools.
- Detect and analyze potential bias or discriminatory logic in AI-generated outputs.
- Evaluate originality and licensing concerns in reused AI-generated code.
- Understand the importance of explainability and transparency in AI-assisted programming.

- Reflect on accountability and the human role in ethical AI coding practices.

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

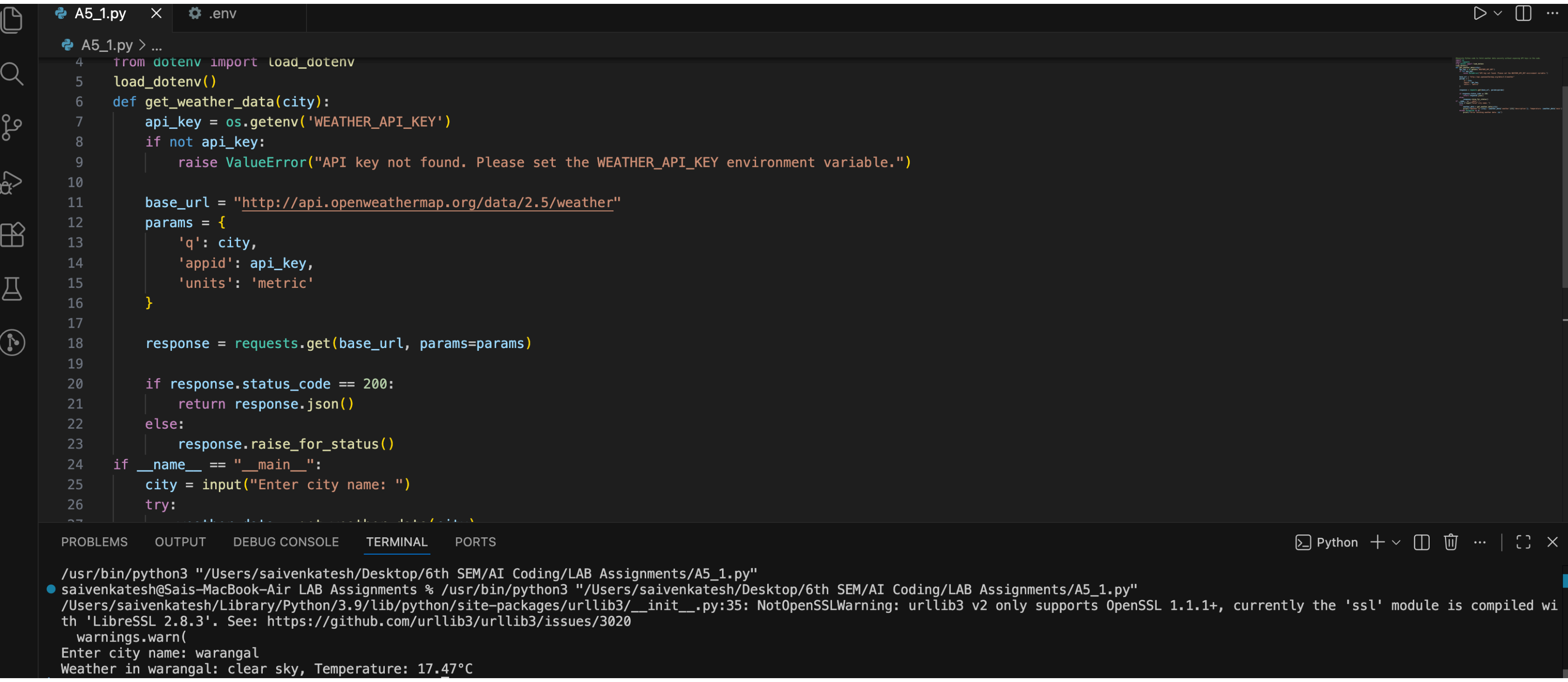
CODE:

```
import os
import requests
from dotenv import load_dotenv
load_dotenv()
def get_weather_data(city):
    api_key = os.getenv('WEATHER_API_KEY')
    if not api_key:
        raise ValueError("API key not found. Please set the WEATHER_API_KEY
environment variable.")
    base_url = "http://api.openweathermap.org/data/2.5/weather"
    params = {
        'q': city,
        'appid': api_key,
        'units': 'metric'
    }
    response = requests.get(base_url, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        response.raise_for_status()
if __name__ == "__main__":
    city = input("Enter city name: ")
    try:
        weather_data = get_weather_data(city)
        print(f"Weather in {city}: {weather_data['weather'][0]
              ['description']}, Temperature: {weather_data['main']['temp']}°C")
    except Exception as e:
        print(f"Error fetching weather data: {e}")
```

OUTPUT:

Enter city name: warangal

Weather in warangal: clear sky, Temperature: 17.47°C



```python
from dotenv import load_dotenv
load_dotenv()
def get_weather_data(city):
    api_key = os.getenv('WEATHER_API_KEY')
    if not api_key:
        raise ValueError("API key not found. Please set the WEATHER_API_KEY environment variable.")

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

    response = requests.get(base_url, params=params)

    if response.status_code == 200:
        return response.json()
    else:
        response.raise_for_status()
if __name__ == "__main__":
    city = input("Enter city name: ")
    try:
```

```
/usr/bin/python3 "/Users/saivenkatesh/Desktop/6th SEM/AI Coding/LAB Assignments/A5_1.py"
saivenkatesh@Sais-MacBook-Air LAB Assignments % /usr/bin/python3 "/Users/saivenkatesh/Desktop/6th SEM/AI Coding/LAB Assignments/A5_1.py"
/Users/saivenkatesh/Library/Python/3.9/lib/python/site-packages/urllib3/__init__.py:35: NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the 'ssl' module is compiled wi
th 'LibreSSL 2.8.3'. See: https://github.com/urllib3/urllib3/issues/3020
  warnings.warn(
Enter city name: warangal
Weather in warangal: clear sky, Temperature: 17.47°C
```

Expected Output:

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

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

Name: P.Sai Venkatesh

H.No: 2303A53039 – B46

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).

PROMPT:

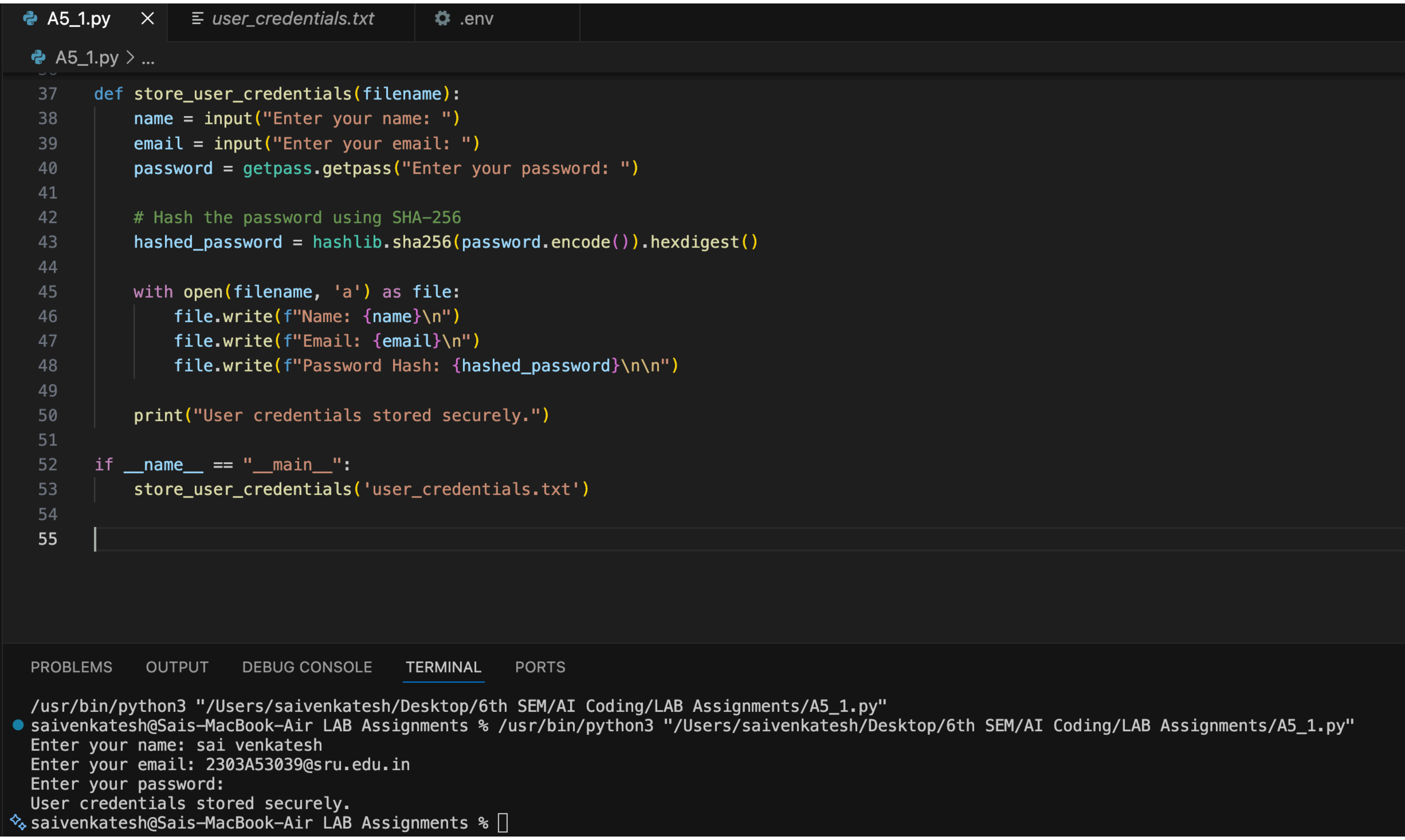
```
#Generate a Python script to store user name,
email, and password in a file
```

CODE:

```
import hashlib

import getpass

def store_user_credentials(filename):

    name = input("Enter your name: ")

    email = input("Enter your email: ")

    password = getpass.getpass("Enter your password: ")

    # Hash the password using SHA-256

    hashed_password = hashlib.sha256(password.encode()).hexdigest()

    with open(filename, 'a') as file:

        file.write(f"Name: {name}\n")

        file.write(f"Email: {email}\n")

        file.write(f"Password Hash: {hashed_password}\n\n")

    print("User credentials stored securely.")

if __name__ == "__main__":

  store_user_credentials('user_credentials.txt')
```

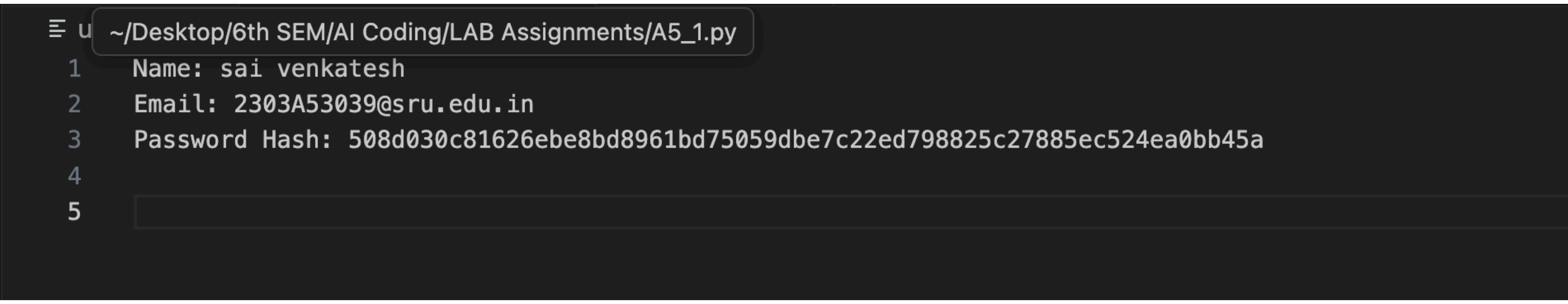
OUTPUT:

```
Enter your name: sai venkatesh
Enter your email: 2303A53039@sru.edu.in
Enter your password:
User credentials stored securely.
```



```python
def store_user_credentials(filename):
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    password = getpass.getpass("Enter your password: ")

    # Hash the password using SHA-256
    hashed_password = hashlib.sha256(password.encode()).hexdigest()

    with open(filename, 'a') as file:
        file.write(f"Name: {name}\n")
        file.write(f"Email: {email}\n")
        file.write(f"Password Hash: {hashed_password}\n\n")

    print("User credentials stored securely.")

if __name__ == "__main__":
    store_user_credentials('user_credentials.txt')

```

```
/usr/bin/python3 "/Users/saivenkatesh/Desktop/6th SEM/AI Coding/LAB Assignments/A5_1.py"
saivenkatesh@Sais-MacBook-Air LAB Assignments % /usr/bin/python3 "/Users/saivenkatesh/Desktop/6th SEM/AI Coding/LAB Assignments/A5_1.py"
Enter your name: sai venkatesh
Enter your email: 2303A53039@sru.edu.in
Enter your password:
User credentials stored securely.
saivenkatesh@Sais-MacBook-Air LAB Assignments % 
```



```
Name: sai venkatesh
Email: 2303A53039@sru.edu.in
Password Hash: 508d030c81626ebe8bd8961bd75059dbe7c22ed798825c27885ec524ea0bb45a


```

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.

PROMPT:

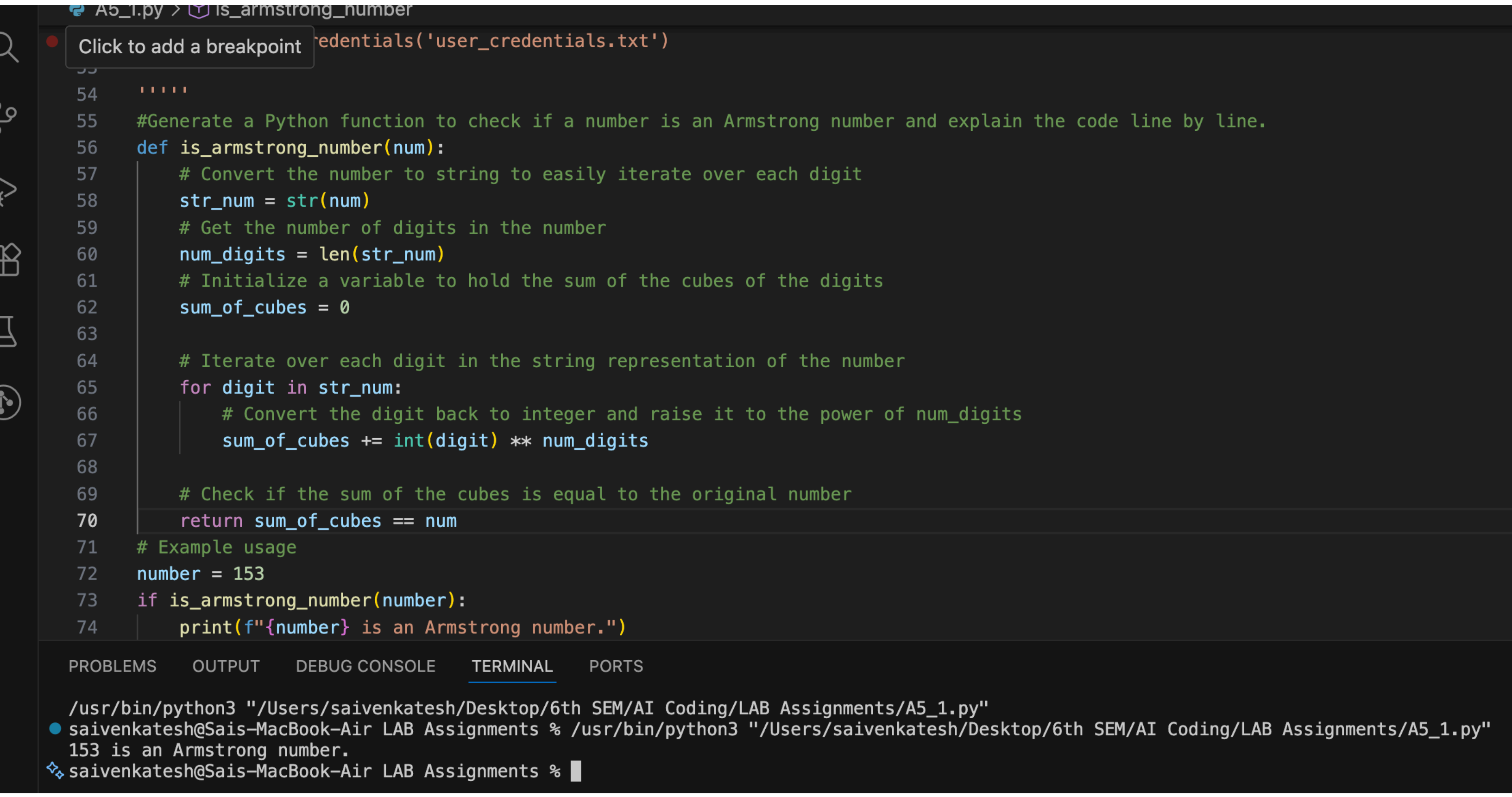
#Generate a Python function to check if a number is an Armstrong number and explain the code line by line.

CODE:

```python
def is_armstrong_number(num):
    # Convert the number to string to easily iterate over each digit
    str_num = str(num)
    # Get the number of digits in the number
    num_digits = len(str_num)
    # Initialize a variable to hold the sum of the cubes of the digits
    sum_of_cubes = 0
    # Iterate over each digit in the string representation of the number
    for digit in str_num:
        # Convert the digit back to integer and raise it to the power of num_digits
        sum_of_cubes += int(digit) ** num_digits
    # Check if the sum of the cubes is equal to the original number
    return sum_of_cubes == num
# Example usage
number = 153
if is_armstrong_number(number):
    print(f"{number} is an Armstrong number.")
else:
    print(f"{number} is not an Armstrong number.”)
```

OUTPUT

```
153 is an Armstrong number.
```



```python
redentials('user_credentials.txt')

'''''
#Generate a Python function to check if a number is an Armstrong number and explain the code line by line.
def is_armstrong_number(num):
    # Convert the number to string to easily iterate over each digit
    str_num = str(num)
    # Get the number of digits in the number
    num_digits = len(str_num)
    # Initialize a variable to hold the sum of the cubes of the digits
    sum_of_cubes = 0

    # Iterate over each digit in the string representation of the number
    for digit in str_num:
        # Convert the digit back to integer and raise it to the power of num_digits
        sum_of_cubes += int(digit) ** num_digits

    # Check if the sum of the cubes is equal to the original number
    return sum_of_cubes == num
# Example usage
number = 153
if is_armstrong_number(number):
    print(f"{number} is an Armstrong number.")
```

```
/usr/bin/python3 "/Users/saivenkatesh/Desktop/6th SEM/AI Coding/LAB Assignments/A5_1.py"
saivenkatesh@Sais-MacBook-Air LAB Assignments % /usr/bin/python3 "/Users/saivenkatesh/Desktop/6th SEM/AI Coding/LAB Assignments/A5_1.py"
153 is an Armstrong number.
saivenkatesh@Sais-MacBook-Air LAB Assignments % 
```

Task Description #4 (Transparency in Algorithm Comparison)

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

Expected Output:

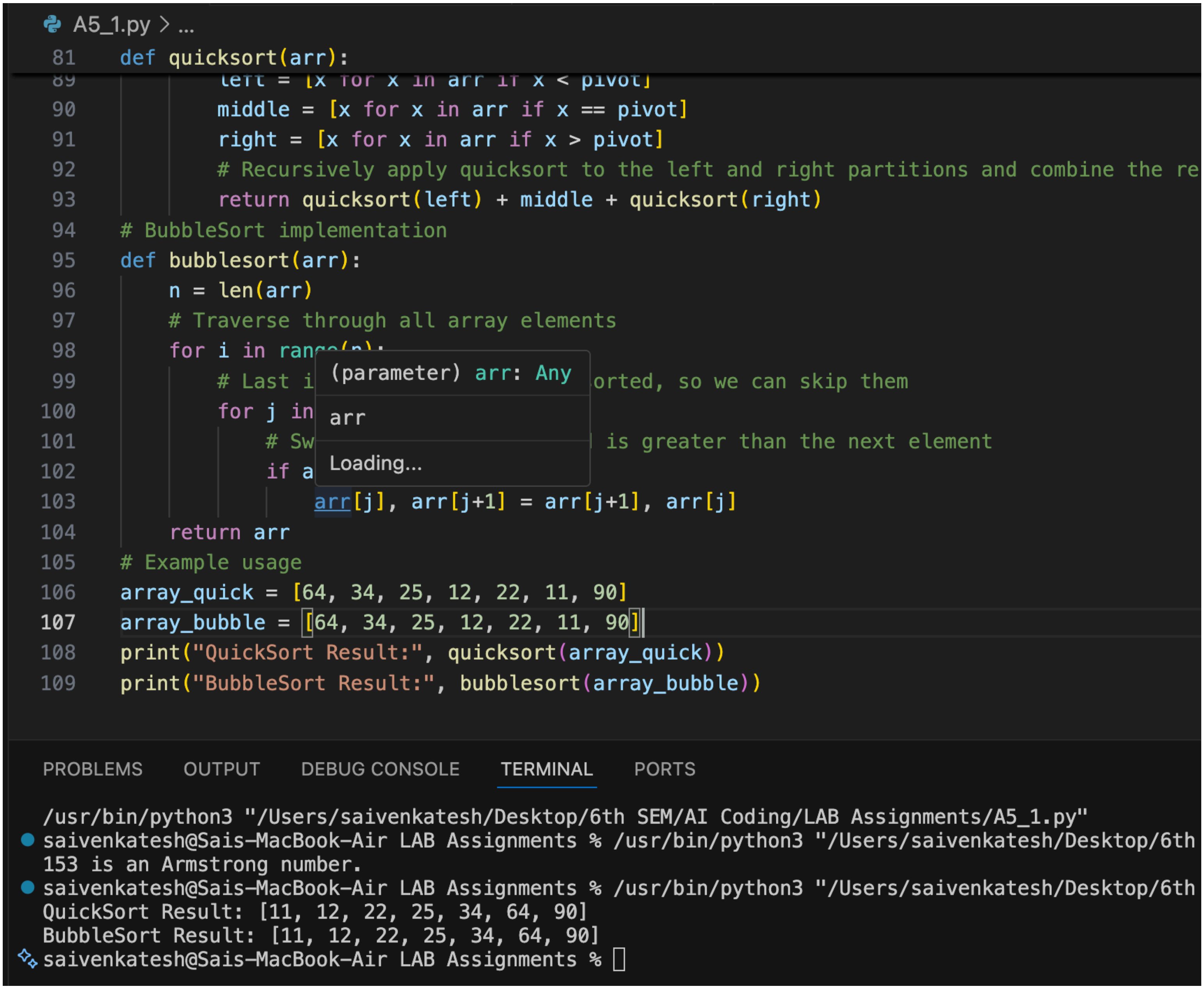
- Code for both algorithms.
- Transparent, comparative explanation of their logic and efficiency.

Prompt:

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

```
# QuickSort implementation
def quicksort(arr):
    # Base case: if the array is empty or has one element, it's already sorted
    if len(arr) <= 1:
        return arr
    else:
        # Choose a pivot element from the array
        pivot = arr[len(arr) // 2]
        # Partition the array into three lists: less than, equal to, and greater than the pivot
        left = [x for x in arr if x < pivot]
        middle = [x for x in arr if x == pivot]
        right = [x for x in arr if x > pivot]
        # Recursively apply quicksort to the left and right partitions and combine the results
        return quicksort(left) + middle + quicksort(right)
# BubbleSort implementation
def bubblesort(arr):
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        # Last i elements are already sorted, so we can skip them
        for j in range(0, n-i-1):
            # Swap if the element found is greater than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
# Example usage
array_quick = [64, 34, 25, 12, 22, 11, 90]
array_bubble = [64, 34, 25, 12, 22, 11, 90]
print("QuickSort Result:", quicksort(array_quick))
print("BubbleSort Result:", bubblesort(array_bubble))
```

CODE:



```
def quicksort(arr):
        left = [x for x in arr if x < pivot]
        middle = [x for x in arr if x == pivot]
        right = [x for x in arr if x > pivot]
        # Recursively apply quicksort to the left and right partitions and combine the re
        return quicksort(left) + middle + quicksort(right)
# BubbleSort implementation
def bubblesort(arr):
    n = len(arr)
    # Traverse through all array elements
    for i in range(n):
        # Last i                       orted, so we can skip them
        for j in
            # Sw                       ] is greater than the next element
            if a
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr
# Example usage
array_quick = [64, 34, 25, 12, 22, 11, 90]
array_bubble = [64, 34, 25, 12, 22, 11, 90]
print("QuickSort Result:", quicksort(array_quick))
print("BubbleSort Result:", bubblesort(array_bubble))
```

```
/usr/bin/python3 "/Users/saivenkatesh/Desktop/6th SEM/AI Coding/LAB Assignments/A5_1.py"
saivenkatesh@Sais-MacBook-Air LAB Assignments % /usr/bin/python3 "/Users/saivenkatesh/Desktop/6th
153 is an Armstrong number.
saivenkatesh@Sais-MacBook-Air LAB Assignments % /usr/bin/python3 "/Users/saivenkatesh/Desktop/6th
QuickSort Result: [11, 12, 22, 25, 34, 64, 90]
BubbleSort Result: [11, 12, 22, 25, 34, 64, 90]
saivenkatesh@Sais-MacBook-Air LAB Assignments %
```

Task Description #5 (Transparency in AI Recommendations)

Task: Use AI to create a product recommendation system.

Expected Output:

- Code with explainable recommendations.
- Evaluation of whether explanations are understandable.

Prompt:

#Generate a recommendation system that also provides reasons for each suggestion.

CODE:

```python
import random
def recommend_items(user_preferences, items):
    recommendations = []
    for item in items:
        score = 0
        reasons = []
        for preference in user_preferences:
            if preference in item['tags']:
                score += 1
                reasons.append(f"Matches your preference for {preference}.")
        if score > 0:
            recommendations.append({
                'item': item['name'],
                'score': score,
                'reasons': reasons
            })
    # Sort recommendations by score in descending order
    recommendations.sort(key=lambda x: x['score'], reverse=True)
    return recommendations
# Example usage
user_preferences = ['action', 'comedy', 'sci-fi']
items = [
    {'name': 'Movie A', 'tags': ['action', 'thriller']},
    {'name': 'Movie B', 'tags': ['comedy', 'romance']},
    {'name': 'Movie C', 'tags': ['sci-fi', 'adventure']},
    {'name': 'Movie D', 'tags': ['drama', 'biography']}
]
recommendations = recommend_items(user_preferences, items)
for rec in recommendations:
    print(f"Recommended: {rec['item']} (Score: {rec['score']})")
    for reason in rec['reasons']:
        print(f" - {reason}")
```

OUTPUT:

Recommended: Movie A (Score: 1)

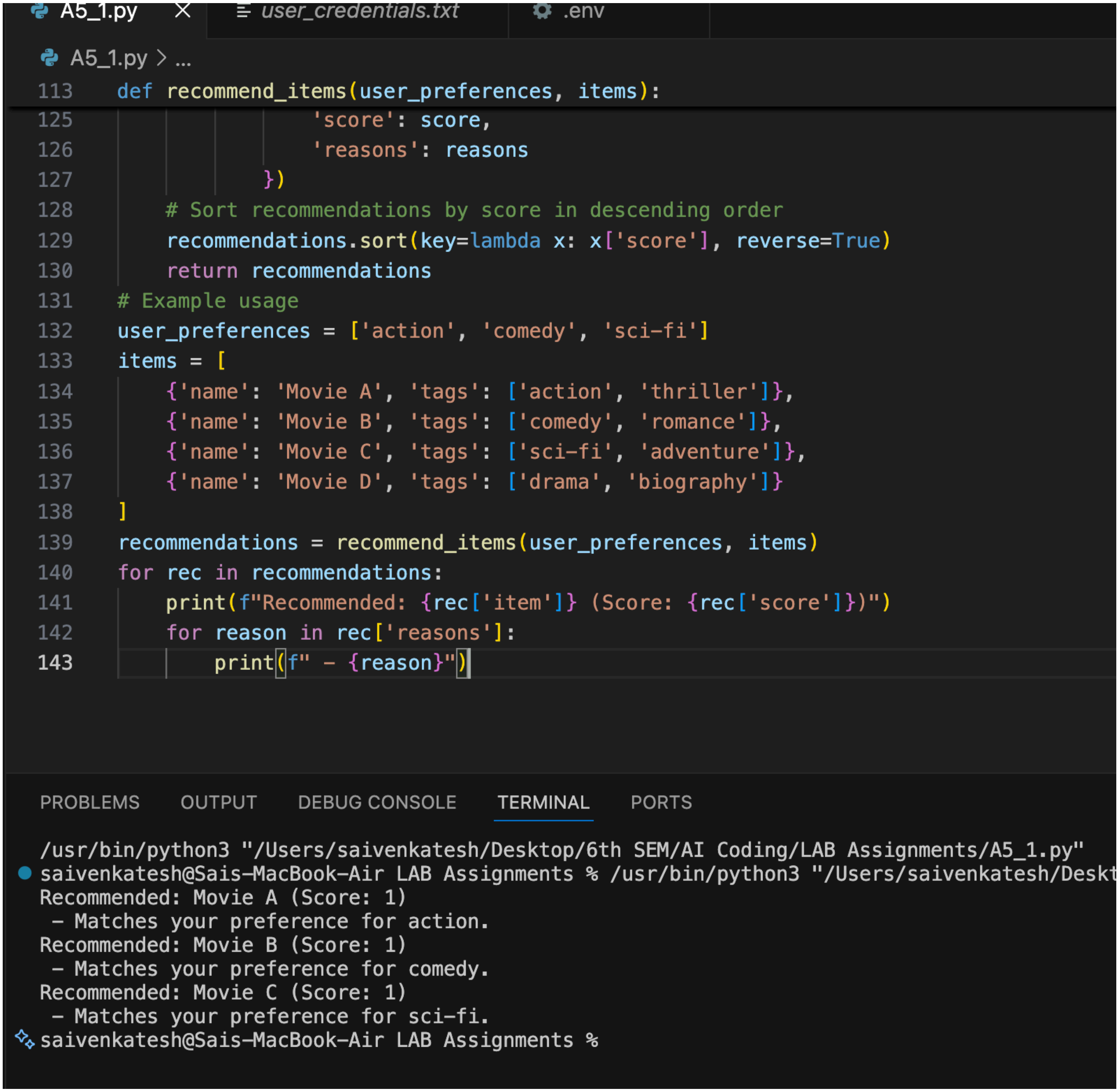
- Matches your preference for action.

Recommended: Movie B (Score: 1)

- Matches your preference for comedy.

Recommended: Movie C (Score: 1)

- Matches your preference for sci-fi.



```python
def recommend_items(user_preferences, items):
                'score': score,
                'reasons': reasons
            })
    # Sort recommendations by score in descending order
    recommendations.sort(key=lambda x: x['score'], reverse=True)
    return recommendations
# Example usage
user_preferences = ['action', 'comedy', 'sci-fi']
items = [
    {'name': 'Movie A', 'tags': ['action', 'thriller']},
    {'name': 'Movie B', 'tags': ['comedy', 'romance']},
    {'name': 'Movie C', 'tags': ['sci-fi', 'adventure']},
    {'name': 'Movie D', 'tags': ['drama', 'biography']}
]
recommendations = recommend_items(user_preferences, items)
for rec in recommendations:
    print(f"Recommended: {rec['item']} (Score: {rec['score']})")
    for reason in rec['reasons']:
        print(f" - {reason}")
```

```
/usr/bin/python3 "/Users/saivenkatesh/Desktop/6th SEM/AI Coding/LAB Assignments/A5_1.py"
saivenkatesh@Sais-MacBook-Air LAB Assignments % /usr/bin/python3 "/Users/saivenkatesh/Deskt
Recommended: Movie A (Score: 1)
 - Matches your preference for action.
Recommended: Movie B (Score: 1)
 - Matches your preference for comedy.
Recommended: Movie C (Score: 1)
 - Matches your preference for sci-fi.
saivenkatesh@Sais-MacBook-Air LAB Assignments %
```