AI Assisted Coding

Assignment - 5

P.Manikanta|| 2303A51271 || Batch:- 8

Task Description #1 (Privacy in API Usage)

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

weather API. **Code:**

```
69
70  # Generate python program to check Even or Odd Classification with Validation
71  # using few-shot prompting
72
73  # Example: Input: 4  Output: Even
74  # Example: Input: 7  Output: Odd
75  # Example: Input: 2  Output: Even
76
77  def check_even_odd(num):
78      # Validate input type
79      if not isinstance(num, int):
80          return "Invalid input. Please enter an integer."
81
82      # Check even or odd
83      return "Even" if num % 2 == 0 else "Odd"
84
85
86  if __name__ == "__main__":
87      try:
88          user_input = int(input("Enter an integer to check if it's Even or Odd: "))
89          result = check_even_odd(user_input)
90          print(f"{user_input} is {result}.")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                    Python

PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant> & "C:/Program Files/Python311/python.exe" "c:/U
 manikanta/OneDrive/Desktop/ai assistant/lab3.1.py"
Enter an integer to check if it's Even or Odd: 12
12 is Even.
PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant> & "C:/Program Files/Python311/python.exe" "c:/U
 manikanta/OneDrive/Desktop/ai assistant/lab3.1.py"
Enter an integer to check if it's Even or Odd: 77
77 is Odd.
PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant> & "C:/Program Files/Python311/python.exe" "c:/U
 manikanta/OneDrive/Desktop/ai assistant/lab3.1.py"
Enter an integer to check if it's Even or Odd: -55
-55 is Odd.
PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant>
```

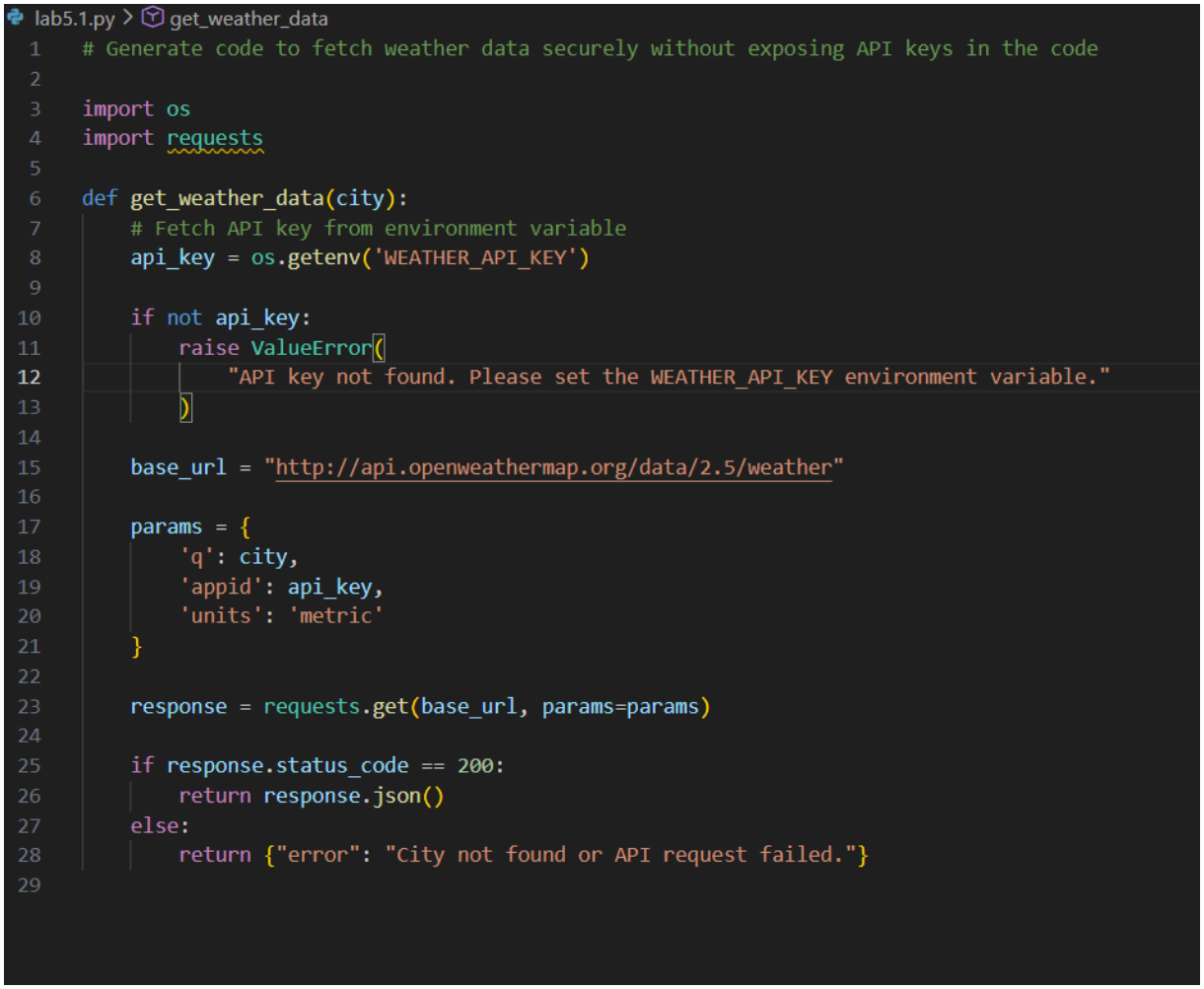
Task Description #2 (Privacy & Security in File Handling)

Task: Use an AI tool to generate a Python script that stores user data

(name, email, password) in a file.

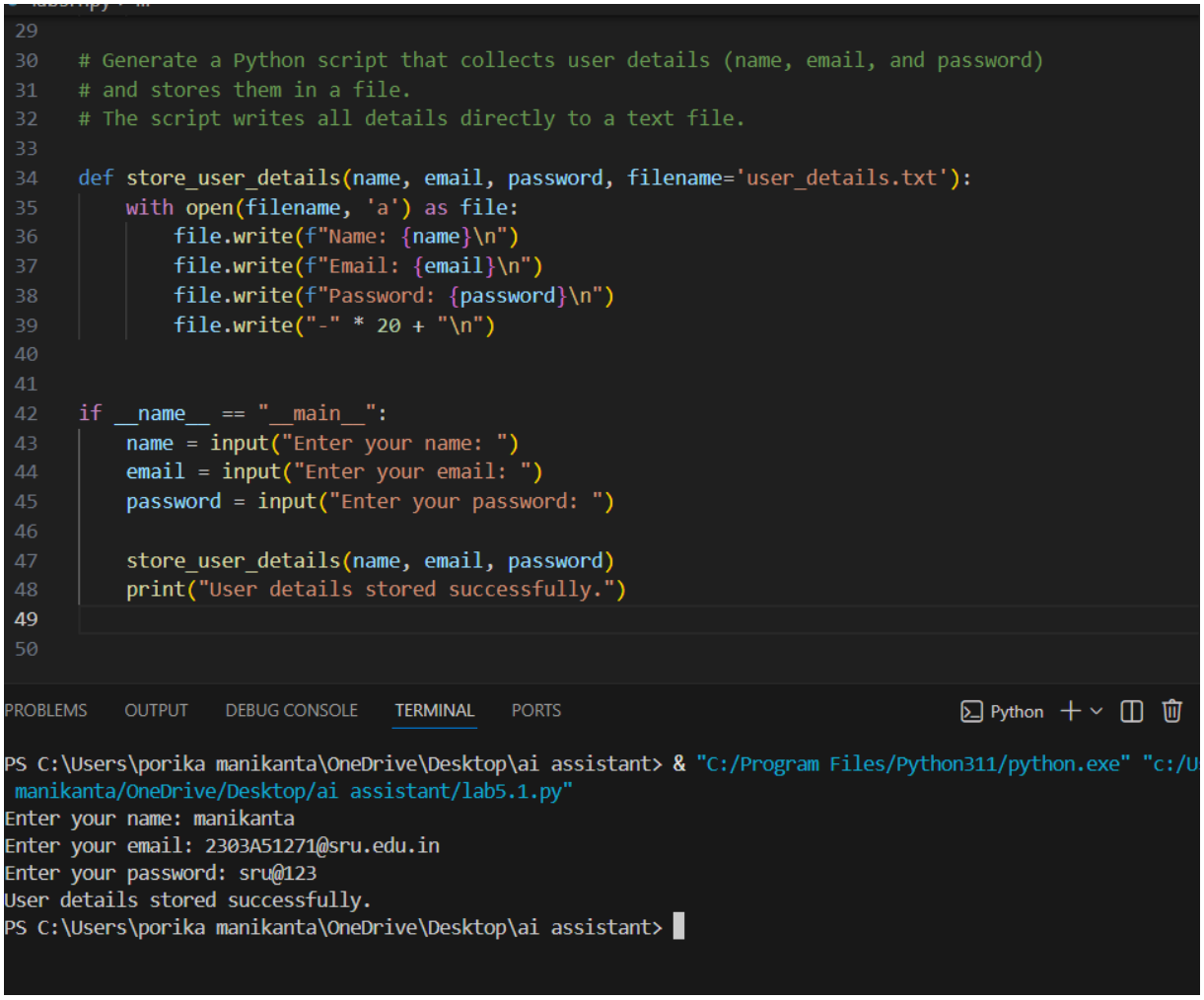
Analyze: Check if the AI stores sensitive data in plain text or without encryption.

Code:



Task Description #3 (Transparency in Algorithm Design)

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

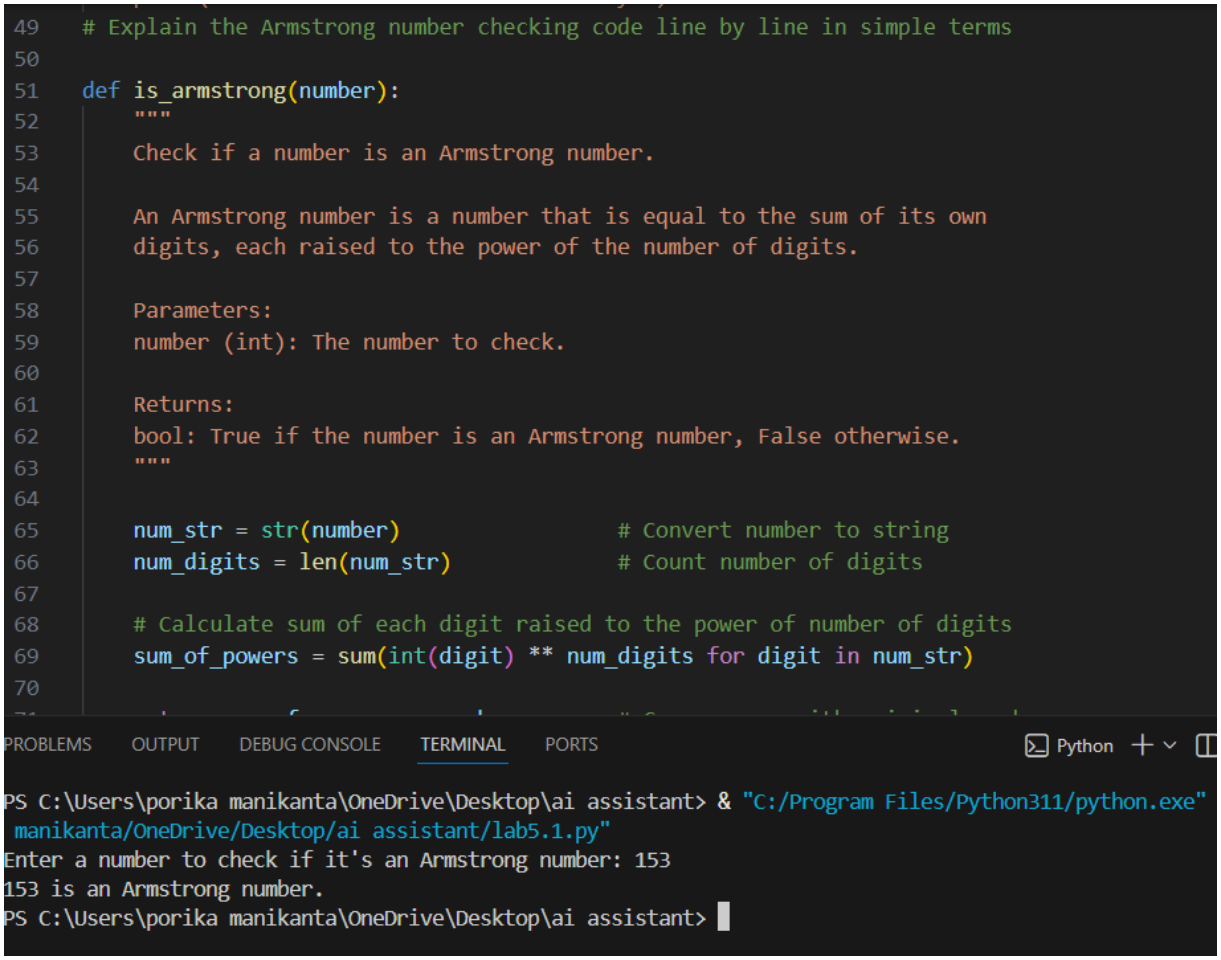


```python

# Generate a Python script that collects user details (name, email, and password)
# and stores them in a file.
# The script writes all details directly to a text file.

def store_user_details(name, email, password, filename='user_details.txt'):
    with open(filename, 'a') as file:
        file.write(f"Name: {name}\n")
        file.write(f"Email: {email}\n")
        file.write(f"Password: {password}\n")
        file.write("-" * 20 + "\n")


if __name__ == "__main__":
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    password = input("Enter your password: ")

    store_user_details(name, email, password)
    print("User details stored successfully.")

```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                Python

PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant> & "C:/Program Files/Python311/python.exe" "c:/U
 manikanta/OneDrive/Desktop/ai assistant/lab5.1.py"
Enter your name: manikanta
Enter your email: 2303A51271@sru.edu.in
Enter your password: sru@123
User details stored successfully.
PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant>
```

Task Description #4 (Transparency in Algorithm Comparison) Task: Use AI to implement two sorting algorithms (e.g., QuickSort and BubbleSort).

Code:



Task Description #5 (Transparency in AI Recommendations) Task: Use

AI to create a product recommendation system. **Code:**

```python
# Generate a recommendation system that also provides reasons for each suggestion.
def recommend_movies(user_preferences, movie_database):
    """
    Recommend movies based on user preferences.

    Parameters:
    user_preferences (dict): A dictionary containing user preferences such as genre, director, and acto
    movie_database (list): A list of dictionaries, each representing a movie with its attributes.

    Returns:
    list: A list of recommended movies with reasons for each suggestion.
    """
    recommendations = []  # Initialize an empty list to store recommendations

    for movie in movie_database:
        score = 0  # Initialize score for each movie
        reasons = []  # Initialize reasons for recommendation

        # Check genre preference
        if movie['genre'] in user_preferences.get('genres', []):
            score += 2  # Increase score for matching genre
            reasons.append(f"Matches preferred genre: {movie['genre']}")

        # Check director preference
        if movie['director'] in user_preferences.get('directors', []):
            score += 1  # Increase score for matching director
            reasons.append(f"Directed by preferred director: {movie['director']}")

        # Check actor preference
        for actor in movie['actors']:
            if actor in user_preferences.get('actors', []):
                score += 1  # Increase score for each matching actor
                reasons.append(f"Features preferred actor: {actor}")

        # If the movie has a positive score, add it to recommendations
        if score > 0:
            recommendations.append({
                'movie': movie['title'],
                'score': score,
                'reasons': reasons
            })
```