

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 + - [] [X]

```
PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant> & "C:/Program Files/Python311/python.exe" "c:/Users/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:/Users/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:/Users/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:

```
lab5.1.py > get_weather_data
1  # Generate code to fetch weather data securely without exposing API keys in the code
2
3  import os
4  import requests
5
6  def get_weather_data(city):
7      # Fetch API key from environment variable
8      api_key = os.getenv('WEATHER_API_KEY')
9
10     if not api_key:
11         raise ValueError(
12             "API key not found. Please set the WEATHER_API_KEY environment variable."
13         )
14
15     base_url = "http://api.openweathermap.org/data/2.5/weather"
16
17     params = {
18         'q': city,
19         'appid': api_key,
20         'units': 'metric'
21     }
22
23     response = requests.get(base_url, params=params)
24
25     if response.status_code == 200:
26         return response.json()
27     else:
28         return {"error": "City not found or API request failed."}
29
```

Task Description #3 (Transparency in Algorithm Design)

Objective: Use AI to generate an Armstrong number checking function

with comments and explanations. **Code:**

```
29
30 # Generate a Python script that collects user details (name, email, and password)
31 # and stores them in a file.
32 # The script writes all details directly to a text file.
33
34 def store_user_details(name, email, password, filename='user_details.txt'):
35     with open(filename, 'a') as file:
36         file.write(f"Name: {name}\n")
37         file.write(f"Email: {email}\n")
38         file.write(f"Password: {password}\n")
39         file.write("-" * 20 + "\n")
40
41
42 if __name__ == "__main__":
43     name = input("Enter your name: ")
44     email = input("Enter your email: ")
45     password = input("Enter your password: ")
46
47     store_user_details(name, email, password)
48     print("User details stored successfully.")
49
50
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] [X]

```
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:

```
49 # Explain the Armstrong number checking code line by line in simple terms
50
51 def is_armstrong(number):
52     """
53     Check if a number is an Armstrong number.
54
55     An Armstrong number is a number that is equal to the sum of its own
56     digits, each raised to the power of the number of digits.
57
58     Parameters:
59     number (int): The number to check.
60
61     Returns:
62     bool: True if the number is an Armstrong number, False otherwise.
63     """
64
65     num_str = str(number)          # Convert number to string
66     num_digits = len(num_str)      # Count number of digits
67
68     # Calculate sum of each digit raised to the power of number of digits
69     sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
70
71     return num_str.isdigit() and sum_of_powers == int(num_str)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + -

```
PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant> & "C:/Program Files/Python311/python.exe"
manikanta/OneDrive/Desktop/ai assistant/lab5.1.py"
Enter a number to check if it's an Armstrong number: 153
153 is an Armstrong number.
PS C:\Users\porika manikanta\OneDrive\Desktop\ai assistant> |
```

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

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

```

112 # Generate a recommendation system that also provides reasons for each suggestion.
113 def recommend_movies(user_preferences, movie_database):
114     """
115     Recommend movies based on user preferences.
116
117     Parameters:
118     user_preferences (dict): A dictionary containing user preferences such as genre, director, and actor.
119     movie_database (list): A list of dictionaries, each representing a movie with its attributes.
120
121     Returns:
122     list: A list of recommended movies with reasons for each suggestion.
123     """
124     recommendations = [] # Initialize an empty list to store recommendations
125
126     for movie in movie_database:
127         score = 0 # Initialize score for each movie
128         reasons = [] # Initialize reasons for recommendation
129
130         # Check genre preference
131         if movie['genre'] in user_preferences.get('genres', []):
132             score += 2 # Increase score for matching genre
133             reasons.append(f"Matches preferred genre: {movie['genre']}")
134
135         # Check director preference
136         if movie['director'] in user_preferences.get('directors', []):
137             score += 1 # Increase score for matching director
138             reasons.append(f"Directed by preferred director: {movie['director']}")
139
140         # Check actor preference
141         for actor in movie['actors']:
142             if actor in user_preferences.get('actors', []):
143                 score += 1 # Increase score for each matching actor
144                 reasons.append(f"Features preferred actor: {actor}")
145
146         # If the movie has a positive score, add it to recommendations
147         if score > 0:
148             recommendations.append({
149                 'movie': movie['title'],
150                 'score': score,
151                 'reasons': reasons
152             })
153

```