**Filename: conversation1.txt**

User:

Can you help me build a Python weather assistant that fetches live weather using wttr.in and shows a menu?

AI:

Sure! Here's a modular Python script that:

- Uses requests to call wttr.in

- Has a text-based menu (with pyinputplus)

- Fetches weather forecasts

- Can plot temperature and precipitation charts

...

[Code Provided: weather assistant with menu, get\_weather\_data, welcome message]

User:

Add temperature and precipitation visualisation using matplotlib.

AI:

Done! I’ve added:

- create\_temperature\_visualisation()

- create\_precipitation\_visualisation()

They both plot data from the forecast, using color-coded lines and bars.

User:

Make sure it catches errors when data is incomplete.

AI:

I added:

- Try/except block in get\_weather\_data()

- Check for keys like 'weather' and 'current\_condition'

...

**Conversation2.txt – Smart AI Q&A Functionality**

vbnet

Copy

Edit

Filename: conversation2.txt

User:

I want to ask questions like “Will it rain tomorrow in Port Louis?” and get an answer.

AI:

Great idea! I added:

- parse\_weather\_question() → detects type (rain/temp), time (today/tomorrow), location

- generate\_weather\_response() → gives a smart answer based on the question

User:

Can you give examples?

AI:

Example:

Question: "Will it rain today in Curepipe?"

→ Response: "☁️ Curepipe - Today rain chance: 60%"

...

User:

Show how the AI decides between temperature and rain?

AI:

It uses keyword detection:

- Words like "hot", "cold", "temperature" → type = 'temperature'

- Words like "rain", "wet", "precipitation" → type = 'rain'

…

**Conversation3.txt – Final Assembly and Explanation**

vbnet

Copy

Edit

Filename: conversation3.txt

User:

Now combine everything into one final working code.

AI:

Here’s your full program, all functions connected to main\_menu():

- Welcome banner

- Menu options

- Weather fetching

- Smart Q&A

- Charts

User:

Explain this whole code clearly, I need to show it to my lecturer.

AI:

Here’s the explanation section-by-section:

- Imports

- Welcome message

- Data fetching

- Chart plotting

- Natural language parsing

- Menu logic

...

User:

Can you structure it well and prepare it like a doc or .txt file?

AI:

Yes, I’ll give you:

- A detailed explanation

- This AI exchange log

- If you want, I can also format it in Markdown or Word

**Filename: conversation4.txt**

User:

What if the user enters more than 5 forecast days?

AI:

Good question. I added a limit using:

`forecast = data.get('weather', [])[:forecast\_days]`

It slices the forecast list to only show up to 5 days.

User:

What if the API is down or the location is invalid?

AI:

I updated get\_weather\_data() to:

- Use `try...except` for connection issues

- Check if 'weather' and 'current\_condition' keys exist

- Print a warning if data is incomplete or invalid

User:

Okay, make the app say "AI Notice: Incomplete data received" when that happens.

AI:

Done! Inside the function:

```python

if 'weather' not in data or 'current\_condition' not in data:

print("⚠️ AI Notice: Incomplete data received. Try another city.")

return None

**Conversation5.txt` – Polishing and User Experience**

Filename: conversation5.txt

User:

Make the app more engaging. Use emojis and cleaner titles.

AI:

Done! I added emojis in:

Welcome screen

Chart titles (📈, ☔)

AI responses (🌡️, ☁️)

Also updated:

plt.title(f"📈 Temperature Trend: {location}")

Main menu with icons for each option

User:

Add color to the temperature line and rain bars.

AI:

Sure! Added:

'orangered' for max temp

'skyblue' for min temp

'dodgerblue' for rain chance bars

User:

Make sure grid lines are visible.

AI:

Grid enabled using:

plt.grid(True) for temperature

plt.grid(True, axis='y') for precipitation

Now the charts are clean, colorful, and informative.