Python Chatbot Project – Developer Team Guide

Welcome developers!

You are a team of developers working on a chatbot product. I am your client and project manager. Your job is to build a Python chatbot that delivers fun, facts across different topics. This guide walks you through everything step-by-step, explaining WHAT to do, WHY you're doing it, and WHEN each part is use facts across different topics. This guide walks you through everything step-by-step, explaining WHAT to do, WHY you're doing it, and WHEN each part is used.

What Is a Scrum Master?

A **Scrum Master** is like a **team coach** in a software development project.

They **don't tell people what to do**, but instead help the team stay focused, organised, and follow the process of **Scrum** — a way of working that breaks projects into small, manageable chunks called **sprints** (usually 1–2 weeks long).

Key Responsibilities:

- **Helps the team** stay on track and remove any obstacles (like a broken tool or unclear instructions)
- **Leads daily check-ins** (called "stand-ups") where team members say what they did, what they're doing, and if they're stuck
- **Protects the team's focus** by keeping distractions away (for example, stopping last-minute changes from disrupting work)
- **Works closely with the Product Owner** to make sure the team understands what needs to be built

Step 1: Project Setup

WHAT: Create two files in your Python project folder: chatbot.py and main.py.

WHY: Keeping your helper functions in a separate file makes your code cleaner and more reusable.

WHEN: Do this before writing any code.

Step 2: Build chatbot.py

WHAT: Add helper functions to print and input messages with a bot-style format.

Type this into chatbot.py:

```
# These are special chatbot functions that make our messages look like a real bot
def cprint(*args):  18 usages
    print("🤖 :", *args)  # This adds "🤖 :" before anything the bot says

def cinput(prompt):  6 usages
    return input("🤖 : " + prompt + " ")  # This adds "🤖 :" before any input questions
```

WHY: These functions make your chatbot easier to understand and look unique.

WHEN: Use cprint and cinput everywhere in your chatbot instead of regular print/input.

Step 3: Build main.py – Facts and Functions

WHAT: Those are dummy facts and topics lists, and functions to keep your chatbot organised. Replace the facts with your own.

```python
# --- Start of the real chatbot program ---

# We are using tools from the chatbot.py file and a random tool to pick random facts
from chatbot import cprint, cinput
from random import randint  # This helps us pick random numbers later

# These are example facts for each topic
# You can change these to match your own project later!
myth_facts = ["Example fact 1", "Example fact 2", "Example fact 3"]
science_facts = ["Example fact 1", "Example fact 2", "Example fact 3"]
history_facts = ["Example fact 1", "Example fact 2", "Example fact 3"]

# This function says hello to the user
def greeting():
    cprint("Welcome to your custom chatbot!")  # Print welcome message
    name = cinput("What's your name?")  # Ask for their name
    cprint( *args: "Nice to meet you,", name + "!")  # Greet them using their name
    return name  # We will use their name later to say goodbye

# This function says goodbye nicely
def goodbye(name):
    cprint( *args: "Goodbye", name + "!")  # Use their name again
    cprint("Thanks for chatting with me!")  # Friendly ending

# This function helps us understand the user's answer
# For example, if they type "Yes" or "Y", we treat it the same
def clean_input(value):
    lowercase = value.lower()  # Turn everything into lowercase
    return lowercase[0]  # Only look at the first letter

# This function picks one random fact from a list
def random_fact(fact_list):
    index = randint( a: 0, len(fact_list) - 1)  # Choose a random number based on the list size
    return fact_list[index]  # Return the fact at that position
```

WHY: Using functions makes your chatbot modular and easier to test.

WHEN: These are the tools your chatbot will rely on during conversation.

Step 4: Create the Main Chatbot Logic

WHAT: This section controls the chatbot's topic switching, looping, and fact delivery.
Type this under your functions in main.py:

```python
# --- Now the chatbot starts talking ---

# Say hello and get their name
name = greeting()

# This is the main loop. It keeps going until the user types 'exit'
while True:
    # Ask which topic they want to learn about
    topic = cinput("Choose a topic: myth, science, or history (or type 'exit' to quit):")
    topic = topic.lower()  # Just in case they used capital letters

    # If they want to leave the chatbot
    if topic == "exit":
        goodbye(name)
        break  # This stops the main loop

    # If they choose the "myth" topic
    elif topic == "myth":
        cprint("You're now in the 'myth' zone!")  # Let them know the topic

        while True:  # Loop for this topic until they switch or quit
            cprint("Here's a myth for you:")
            cprint(random_fact(myth_facts))  # Show a random myth

            again = cinput("Would you like another myth? (y = yes, s = switch topic, n = exit): ")
            choice = clean_input(again)

            if choice == "y":
                continue  # Give them another fact
            elif choice == "s":
                break  # Go back to topic selection
            elif choice == "n":
                goodbye(name)
                exit()  # End the whole program
            else:
                cprint("Please enter y, s, or n.")  # Help them if they typed something wrong
```

```
# If they choose the "science" topic
elif topic == "science":
    cprint("You're now in the 'science' zone!")

    while True:
        cprint("Here's a science fact:")
        cprint(random_fact(science_facts))

        again = cinput("Would you like another science fact? (y = yes, s = switch topic, n = exit): ")
        choice = clean_input(again)

        if choice == "y":
            continue
        elif choice == "s":
            break
        elif choice == "n":
            goodbye(name)
            exit()
        else:
            cprint("Please enter y, s, or n.")

# If they choose the "history" topic
elif topic == "history":
    cprint("You're now in the 'history' zone!")

    while True:
        cprint("Here's a historical rumour:")
        cprint(random_fact(history_facts))

        again = cinput("Would you like another historical rumour? (y = yes, s = switch topic, n = exit): ")
        choice = clean_input(again)

        if choice == "y":
            continue
        elif choice == "s":
            break
        elif choice == "n":
            goodbye(name)
            exit()
        else:
            cprint("Please enter y, s, or n.")
```

```
# If the topic they typed doesn't match any we know
else:
    cprint("Hmm... I don't know that topic. Try 'myth', 'science', or 'history'.")
```

WHY: This section lets your chatbot behave like a conversation, looping through facts and responding to input.

WHEN: This runs every time the user talks to the bot.

Step 5: Customise Your Bot

WHAT: Change the topics, facts, and prompts to match your own project or idea.

WHY: This makes your project unique and fun to build.

WHEN: Once the chatbot is working, start replacing the example data.

🧭 **Project Milestones –**

MythBot Chatbot

These are the **key stages** of your chatbot development. At each one, check in with your **Project Manager (your tutor)** to demo your progress or ask for support.

✅

Milestone 1: Project Setup

Goal: Create a clean working environment with the required files.

- Create a project folder called my_chatbot
- Create a file named chatbot.py for helper functions
- Create a file named main.py for the main chatbot logic

✅

Milestone 2: Write Helper Functions

Goal: Build and test your cprint() and cinput() functions in chatbot.py.

- Write cprint() to prefix bot messages
- Write cinput() to prefix bot questions
- Test both in a separate file or Python shell

✅

Milestone 3: Add Dummy Data and Utility Functions

Goal: Add sample facts and basic tools to main.py.

- Add three lists: myth_facts, science_facts, history_facts
- Write and test:
    - greeting()
    - goodbye(name)
    - clean_input(value)
    - random_fact(fact_list)

✅

Milestone 4: First Topic Interaction

Goal: Let the user choose a topic and receive a fact.

- Ask the user to choose a topic using input()
- Use if/elif to match the topic and print one fact
- Return to the topic menu after showing a fact

✅

Milestone 5: Loop and Repeat

Goal: Allow the user to keep asking for more facts in a topic.

- Wrap each topic in a while True: loop
- Ask if they want:
    - Another fact (y)

    - To switch topic (s)
    - To exit (n)
- Use clean_input() to handle answers

✅

Milestone 6: Full Chatbot Behaviour

Goal: Finalise all chatbot behaviours.

- Allow full topic switching
- Handle unknown inputs politely
- Use the user's name in greetings and goodbyes
- Exit gracefully when user types exit or chooses n

📋 **Developer Checklist Summary**

✅ Task	Done?
Project folder and files created	☐
chatbot.py helper functions working	☐
Fact lists created with example data	☐
Greeting and goodbye functions tested	☐
User can choose a topic and get a fact	☐
Facts repeat until the user says stop	☐
User can switch topics	☐
Program exits correctly on command	☐
Invalid input handled gracefully	☐
All code is indented and runs with no errors	☐

Final Notes

You are working like a real development team. I will act as your client and project manager. If you get stuck or complete a milestone, check in with me. Test your bot often, and always keep your code tidy!