Project 2 – AI Study Buddy (Rule-Based Chat Assistant in Python)

Objective

To **create a conversational AI assistant** using Python's core logic - string matching, functions, dictionaries, and loops.

Concepts Used

Concept	**Purpose**
Strings	To analyze and compare user input
Conditionals (if-elif-else)	To decide chatbot responses
Dictionaries	To store questions and responses
Functions	To organize chatbot logic
Loops	To keep the chat running
User Input (`input()`)	To take input from the user

Description

In this project, we will **build a mini AI chatbot** that can interact with users. The chatbot will understand simple messages like "hello," "motivate me," or "python" and respond with friendly, intelligent answers.

This is a **rule-based chatbot**, it doesn't use real AI models yet, but it behaves intelligently using Python logic.

Sample Output

```
🤖 Hello! I am Saumya's Study Buddy!

You can talk to me. Type 'bye' anytime to exit.


You: Hello

Bot: Hi there! How can I help you today?


You: Who are you?

Bot: I'm your friendly AI Study Buddy created by Saumya Singh.


You: Motivate me

Bot: Keep going! Every bug you fix makes you a better coder 💪


You: Tell me about Python

Bot: Python is powerful — it can do AI, automation, and much more!


You: Bye

Bot: Goodbye! Keep learning and keep smiling 😊
```

Features

- Responds to simple user inputs
- Works continuously until the user types `bye`
- Easy to customize with new keywords and responses
- Demonstrates real-world chatbot logic using Python fundamentals

Future Scope

1. **Add time-based greetings (using `datetime` module)**
2. Add text-to-speech (using `pyttsx3`)
3. Add voice input (using `speech_recognition`)
4. Connect to an **AI API** for real answers (like OpenAI / Hugging Face)
5. Store chat history in a file using **File Handling**

📜 Showcase as Minor Project

Project Title: AI Study Buddy – Rule-Based Chat Assistant in Python

Objective: To **create a conversational assistant** using Python's core logic - string matching, functions, dictionaries, and loops.

Key Features:

- Keyword-based intent recognition
- Dynamic responses using dictionaries
- Modular function design
- Expandable to API-based real AI

Future Scope:
Integration with NLP APIs, Text-to-Speech, and Voice Recognition for full AI assistant behavior.

Showcase in Resume / LinkedIn / Portfolio

AI Study Buddy – Rule-Based Chat Assistant
Designed a smart chatbot using Python fundamentals (loops, conditionals, and dictionaries). Implemented a keyword-matching logic to make it behave like a real AI assistant and built modular functions for reusability.
(Future Plan: connect to OpenAI API for real AI responses.)

Practice Tasks

1. Add at least 3 new questions and responses to the chatbot.
2. Add logic to respond differently if the user says “sad” or “happy.”
3. Add a delay (1 second) between question and answer using `time.sleep()`.

✅ Solution Code

```
# -----------------------------------------------
# Project 2 – AI Study Buddy (Rule-Based Chat Assistant)
# Created by: Saumya Singh
# -----------------------------------------------

import datetime
import time

# --- Step 1: Time-based greeting ---
hour = datetime.datetime.now().hour

if 5 <= hour < 12:
    print("Good morning, Saumya! ☀️")
elif 12 <= hour < 17:
    print("Good afternoon, Saumya! 🌤️")
elif 17 <= hour < 21:
    print("Good evening, Saumya! 🌇")
else:
    print("Good night, Saumya! 🌙")

# ---------------

print("\n🤖 Hello! I am Saumya's Study Buddy!")
print("You can talk to me. Type 'bye' anytime to exit.\n")

# --- Step 2: Chatbot memory (dictionary of responses) -

responses = {
    "hello": "Hi there! How can I help you today?",
    "who are you": "I'm your friendly AI Study Buddy created by
Saumya Singh.",
    "how are you": "I'm just code, but I feel great when you run
me!",
    "motivate me": "Keep going! Every bug you fix makes you a better
coder 💪",
    "python": "Python is powerful — it can do AI, automation, and
much more!",
    "sad": "Don't worry! Even code breaks sometimes, but it always
runs again 😊",
    "happy": "That's great to hear! Keep that positive energy going
🎉",
    "time": f"The current time is
{datetime.datetime.now().strftime('%H:%M:%S')}",
    "bye": "Goodbye! Keep learning and keep smiling 😊"
}


# --- Step 3: Function to find matching response ---
```

```
def get_response(user_input):
    user_input = user_input.lower()
    for key in responses:
        if key in user_input:
            return responses[key]
    return "Hmm... I’m not sure about that yet, but I’ll learn 
soon!"

# --- Step 4: Main chat loop ---

while True:
    user = input("Your Question: ")
    reply = get_response(user)
    time.sleep(0.7)  # slight delay for realistic feel
    print("Bot Answer:", reply)

    if "bye" in user.lower():
        break
```