import datetime

class Chatbot:

def \_\_init\_\_(self):

self.history = []

self.commands\_used = {}

self.greet()

def greet(self):

print("Chatbot: Hello! I’m your assistant! How can I help you today?")

self.history.append("Chatbot: Hello! I’m your assistant! How can I help you today?")

self.listen()

def listen(self):

user\_input = input("User: ")

self.history.append(f"User: {user\_input}")

self.process\_input(user\_input)

def process\_input(self, user\_input):

if user\_input.lower() in ["hello", "hi"]:

self.respond("Hi there! How can I help you today?")

elif user\_input.lower() == "todays date/time":

now = datetime.datetime.now().strftime("%d %B %Y, %I:%M %p")

self.respond(now)

self.respond("How else can I assist you?")

elif user\_input.lower() == "list operations":

self.respond("Please enter a list of integers (comma-separated, integer):")

list\_input = input("User: ")

self.history.append(f"User: {list\_input}")

try:

num\_list = [int(x.strip()) for x in list\_input.split(",")]

self.list\_operations(num\_list)

except ValueError:

self.respond("The list must contain integers only.")

elif user\_input.lower() == "generate prime":

self.respond("Enter the range (start and end):")

range\_input = input("User: ")

self.history.append(f"User: {range\_input}")

try:

start, end = [int(x.strip()) for x in range\_input.split(",")]

primes = self.generate\_primes(start, end)

self.respond(f"Prime Numbers: {primes}")

except ValueError:

self.respond("The range must contain integers only.")

elif user\_input.lower() == "search history":

self.respond("Enter the keyword to search in chat history:")

keyword = input("User: ")

self.history.append(f"User: {keyword}")

found\_lines = [line for line in self.history if keyword.lower() in line.lower()]

if found\_lines:

self.respond(f"Found the following lines:\n- " + "\n- ".join(found\_lines))

else:

self.respond("No matching lines found.")

elif user\_input.lower() == "bye":

summary = f"Here’s a summary of your session:\n- Commands Used: {len(self.commands\_used)}\n- Most Frequent Command: {max(self.commands\_used, key=self.commands\_used.get)}"

self.respond(summary)

self.respond("Do you want to save this summary? (yes/no)")

save\_summary = input("User: ")

if save\_summary.lower() == "yes":

filename = f"summary\_{datetime.datetime.now().strftime('%d%m%Y')}.txt"

with open(filename, 'w') as f:

f.write(summary)

self.respond(f"Summary saved to {filename}")

self.respond("Goodbye! Have a great day!")

return

else:

self.respond("Goodbye! Have a great day!")

return

else:

self.respond("Enter correct keyword")

# Track command usage

command = user\_input.split()[0].lower()

if command in self.commands\_used:

self.commands\_used[command] += 1

else:

self.commands\_used[command] = 1

# Continue listening

self.listen()

def respond(self, response):

print(f"Chatbot: {response}")

self.history.append(f"Chatbot: {response}")

def list\_operations(self, num\_list):

sum\_list = sum(num\_list)

max\_list = max(num\_list)

reversed\_list = num\_list[::-1]

self.respond(f"Sum: {sum\_list}")

self.respond(f"Maximum: {max\_list}")

self.respond(f"Reversed List: {reversed\_list}")

self.respond("Would you like to remove duplicates? (yes/no)")

remove\_duplicates = input("User: ")

if remove\_duplicates.lower() == "yes":

updated\_list = list(set(num\_list))

updated\_list.sort()

sum\_updated\_list = sum(updated\_list)

max\_updated\_list = max(updated\_list)

self.respond(f"Updated List: {updated\_list}")

self.respond(f"Sum: {sum\_updated\_list}")

self.respond(f"Maximum: {max\_updated\_list}")

# Continue listening

self.listen()

def generate\_primes(self, start, end):

primes = []

for num in range(start, end + 1):

if num > 1:

for i in range(2, num):

if (num % i) == 0:

break

else:

primes.append(num)

return primes

# Create an instance of the Chatbot

chatbot = Chatbot()