# Chatbot

By Hatem Elgenedy

December 7, 2025

[1]:
```python
import tkinter as tk # GUI library
from tkinter import scrolledtext # Scrollable text widget

# -----------------------------
# Data: phrases + bot responses
# -----------------------------
USER_PHRASE_LIBRARY = [
    "hello",
    "hello there",
    "how are you",
    "how can I help you",
    "what is your name",
    "what can you do",
    "thank you",
    "bye",
    "good morning",
    "good night",
    "help me with my homework",
    "help me with my code",
    "tell me a joke",
    "tell me something interesting",
]

BOT_RESPONSES = {
    "hello": "Hi! How can I help you today?",
    "hello there": "Hello there!  ",
    "how are you": "I'm just code, but I'm doing great! Thanks for asking.",
    "how can I help you": "You can tell me what you want to do, and I'll try to␣
↪assist.",
    "what is your name": "I'm a simple autocomplete chatbot written in Python.",
    "what can you do": "Right now, I can autocomplete your text and give basic␣
↪replies.",
    "thank you": "You're welcome!",
    "bye": "Goodbye!  ",
    "good morning": "Good morning! Hope you have a great day.",
    "good night": "Good night! Sleep well.",
    "help me with my homework": "Sure, tell me what subject your homework is in.
↪",
    "help me with my code": "Tell me what language you're using and what the␣
↪problem is.",
    "tell me a joke": "Why do programmers prefer dark mode? Because light␣
↪attracts bugs.  ",
    "tell me something interesting": "Did you know? Python was named after␣
↪Monty Python, not the snake.",
}

# -----------------------------
# Autocomplete function
# -----------------------------
# Returns phrases that start with the given prefix
def get_suggestions(prefix, phrases, max_results=5): # -> list:
    """
    Return phrases that start with the given prefix (case-insensitive).
    """
    prefix = prefix.lower().strip()
    if not prefix: # Empty prefix, return no suggestions
        return []

    matches = [] # Collect matching phrases
    for p in phrases: # Check each phrase
        if p.lower().startswith(prefix): # Match found
            matches.append(p) # Add to results
    return matches[:max_results]


class AutocompleteChatbotApp: # Main application class
    def __init__(self, root): # Initialize GUI components
        self.root = root # Main window
        self.root.title("Autocomplete Chatbot") # Window title

        # ------------- Chat display -------------
        # Scrolled text area for chat history
        # Disable editing by user
        # Wrap text by words

        self.chat_area = scrolledtext.ScrolledText(root, wrap=tk.WORD, state=tk.
↪DISABLED, height=15, width=60)
        self.chat_area.grid(row=0, column=0, columnspan=2, padx=10, pady=10)

        # ------------- Entry field --------------
        # Text entry for user input
        self.entry_var = tk.StringVar()
        self.entry = tk.Entry(root, textvariable=self.entry_var, width=40) #␣
↪Entry widget for user input

        self.entry.grid(row=1, column=0, padx=10, pady=(0, 5), sticky="we") #␣
↪Expand horizontally

        # Bind typing event for autocomplete
        self.entry.bind("<KeyRelease>", self.on_key_release) # Update␣
↪suggestions on key release

        # Bind Enter to send
        self.entry.bind("<Return>", self.on_enter_pressed) # Send message on␣
↪Enter key

        # ------------- Send button --------------
        self.send_button = tk.Button(root, text="Send", command=self.
↪send_message) # Send button
        self.send_button.grid(row=1, column=1, padx=10, pady=(0, 5),␣
↪sticky="e") # Align right

        # ------------- Suggestions list ----------
        self.suggestion_box = tk.Listbox(root, height=5) # Listbox for␣
↪suggestions
        self.suggestion_box.grid(row=2, column=0, columnspan=2, padx=10,␣
↪pady=(0, 10), sticky="we") # Expand horizontally

        # When user double-clicks a suggestion
        self.suggestion_box.bind("<Double-Button-1>", self.
↪on_suggestion_selected)

        # Greet user

        self.add_chat_message("Bot", "Hi! Start typing and I'll suggest␣
↪completions.")
# ------------- Methods -----------------
# define methods for chat functionality
# for adding messages, handling input, and suggestions
    def add_chat_message(self, sender, message): # Add message to chat area
        self.chat_area.config(state=tk.NORMAL) # Enable editing
        self.chat_area.insert(tk.END, f"{sender}: {message}\n") # Insert message
        self.chat_area.config(state=tk.DISABLED) #
        self.chat_area.see(tk.END)
# ------------- Event Handlers -------------
# define event handlers for key releases, suggestion selection, and enter key␣
↪press
    def on_key_release(self, event): # Handle key release in entry
```

```
        """
        Called whenever the user types in the entry.
        Updates the suggestion box.
        """
        text = self.entry_var.get() # Current text in entry ,
        suggestions = get_suggestions(text, USER_PHRASE_LIBRARY) # Get␣
↪suggestions

        # Clear and repopulate the listbox
        self.suggestion_box.delete(0, tk.END) # Clear previous suggestions
        for item in suggestions: # Add new suggestions
            self.suggestion_box.insert(tk.END, item) # Insert suggestion

    def on_suggestion_selected(self, event): # Handle suggestion selection
        """
        When a suggestion is double-clicked, fill the entry and send it.
        """
        selection = self.suggestion_box.curselection() # Get selected suggestion
        if not selection:
            return
        chosen_text = self.suggestion_box.get(selection[0]) # Get text of␣
↪selected suggestion
        self.entry_var.set(chosen_text) # Set entry to chosen text
        self.suggestion_box.delete(0, tk.END) # Clear suggestions
        self.send_message() # Send the selected suggestion
# define more methods for sending messages and handling enter key press
    def on_enter_pressed(self, event): # Handle Enter key press
        """
        When user presses Enter in the entry box.
        """
      # s
        self.send_message()
        # Prevent default beep on Enter (Windows)
        return "break"

    def send_message(self): # Send user message and bot reply
        user_text = self.entry_var.get().strip() # Get user input
        if not user_text: # Ignore empty input
            return

        # Show user message
        self.add_chat_message("You", user_text) # Add user message to chat area

        # Clear entry and suggestions
        self.entry_var.set("") # set entry to empty
        self.suggestion_box.delete(0, tk.END) # Clear suggestions
        # Generate bot reply
        reply = BOT_RESPONSES.get( # Get bot response or default reply
            user_text.lower(),
            "I don't have a specific answer for that yet, but I'm learning!"
        )
        self.add_chat_message("Bot", reply) # Add bot reply to chat area

# configure and run the application
if __name__ == "__main__": # Run the application
    root = tk.Tk()
    app = AutocompleteChatbotApp(root) # Create app instance
    root.mainloop() # Start GUI event loop
```