Chatbot

This is a simple chatbot implemented using Python and Tkinter. The chatbot uses a knowledge base in the form of a JSON file to find responses to user messages. If a close match to the user's message is found in the knowledge base, the chatbot will send the first response for that message. Otherwise, it will send a message saying that the message is not in its knowledge base.

Dependencies

- json: Used for loading the knowledge base from a JSON file.
- difflib: Used for finding close matches to user messages in the knowledge base.
- tkinter: Used for creating the user interface.

Class: Chatbot

Properties

- **Brain**: A dictionary containing the knowledge base. It is loaded from the `knowledge.json` file.
- **message_session**: A Text widget for displaying the chat session.
- **overscroll**: A Scrollbar widget for the `message_session` Text widget.
- **send_button**: A Button widget for sending messages.
- **Message_Entry**: An Entry widget for typing messages.
- **message_position**: A float representing the current position in the `message_session` Text widget.

Methods

- **add_chat(message: str)**: Adds a message to the chat session. The message is inserted at the current `message_position` and the `message_position` is incremented by 1.5.
- **reply_to_you(event=None)**: Sends a reply to the user's message. The user's message is taken from the `Message_Entry` Entry widget, and a close match is found in the `Brain` knowledge base using the `get_close_matches` function from the `difflib` library. If a close match is found, the first response for that message is sent. Otherwise, a message saying that the message is not in the knowledge base is sent. The user's message and the chatbot's reply are then added to the chat session using the `add_chat` method.

Functions

- **main()**: Creates the root window and a Chatbot object with the root window as a parameter. The mainloop is then started to display the window and process events.