import re

import tkinter as tk

from tkinter import scrolledtext

from elements import get\_name\_from\_symbol, get\_symbol\_from\_name, unrecognized\_response, initial\_response, get\_prefixes, get\_elements

# Timer variable

inactivity\_timer = None

prefixes = get\_prefixes()

elements = get\_elements()

import re

def parse\_formula(formula):

    """Parse the chemical formula and return a dictionary of elements and their counts.

    This version ensures that two-character element symbols are prioritized over single-character symbols.

    """

    # Regular expression to find elements with optional numbers (e.g., "H2", "NaCl", "H2O")

    element\_pattern = r"([A-Z][a-z]?)(\d\*)"

    matches = re.findall(element\_pattern, formula)

    elements\_dict = {}

    for match in matches:

        element, count = match

        count = int(count) if count else 1

        # Update the dictionary with the element and its count

        elements\_dict[element] = elements\_dict.get(element, 0) + count

        print(elements\_dict[element])

    return elements\_dict

def generate\_chemical\_name(formula):

    """Generate the chemical name from the given formula."""

    elements\_dict = parse\_formula(formula)

    # Sort the elements by their symbol

    sorted\_elements = sorted(elements\_dict.items())

    names = []

    for element, count in sorted\_elements:

        if element not in elements:

            raise ValueError(f"Element {element} not found in the periodic table")

        element\_name = elements[element]

        if count > 1:

            prefix = prefixes[count - 1] if count <= len(prefixes) else str(count)

            name\_part = f"{prefix}{element\_name.lower()}"

        else:

            name\_part = element\_name.lower()

        names.append(name\_part.capitalize())

    # Combine the names for a full chemical name

    return " ".join(names)

# Function to handle user input and output

def process\_input(event=None):

    global inactivity\_timer

    user\_input = entry.get().strip()

    if user\_input:

        normalized\_input = user\_input.capitalize()

        # Check if input is a chemical symbol (e.g., "H")

        if normalized\_input.isalpha() and len(normalized\_input) <= 2:

            response = get\_name\_from\_symbol(normalized\_input)

        # Check if input is a chemical name (e.g., "Hydrogen")

        elif normalized\_input.istitle() and len(normalized\_input) > 1:

            response = get\_symbol\_from\_name(normalized\_input)

        # Check if input is a chemical formula (e.g., "H2O")

        elif any(char.isdigit() for char in normalized\_input):  # Contains a number, likely a formula

            try:

                response = f"Formula: {normalized\_input} detected. It is {generate\_chemical\_name(normalized\_input)}"

            except ValueError:

                response = "Formula parsing error. Please check the formula."

        else:

            response = unrecognized\_response()

        # Display user input and bot's response

        chat\_box.config(state=tk.NORMAL)

        if user\_input != "":

            chat\_box.insert(tk.END, "You: " + user\_input + '\n')

        chat\_box.insert(tk.END, "Bot: " + response + '\n\n')

        chat\_box.config(state=tk.DISABLED)

        # Clear the entry box

        entry.delete(0, tk.END)

        # Reset inactivity timer after a response

        if inactivity\_timer:

            root.after\_cancel(inactivity\_timer)

        inactivity\_timer = None

# Function to handle inactivity (user stops typing for 30 seconds)

def handle\_inactivity():

    global inactivity\_timer  # Declare inactivity\_timer as global

    if inactivity\_timer is None:

        random\_response = initial\_response()  # This will only trigger once after inactivity

        chat\_box.config(state=tk.NORMAL)

        chat\_box.insert(tk.END, "Bot: " + random\_response + '\n\n')

        chat\_box.config(state=tk.DISABLED)

        inactivity\_timer = None  # Reset the inactivity timer after response

# Set up the main window

root = tk.Tk()

root.title("Chemical Chatbot")

# Apply a futuristic theme

root.config(bg="#121212")  # Dark background

root.geometry("600x500")

root.resizable(False, False)

# Set up the chat box without a scrollbar

chat\_box = scrolledtext.ScrolledText(root, wrap=tk.WORD, width=61, height=18, state=tk.DISABLED,

                   bg="#1e1e1e", fg="white", font=("Segoe UI", 12), padx=1, pady=1)

chat\_box.grid(row=0, column=0, columnspan=2, padx=20, pady=20)

# Set up the entry box with a futuristic look

entry = tk.Entry(root, width=40, font=("Segoe UI", 14), fg="#333333", bg="#1e1e1e", relief="flat",

                 bd=1, insertbackground="white", highlightbackground="#3c3c3c", highlightthickness=2)

entry.grid(row=1, column=0, padx=5, pady=5)

# Set up the send button with a glowing effect

send\_button = tk.Button(root, text="Send", width=10, font=("Segoe UI", 14), fg="white", bg="#4CAF50", relief="flat",

                        activebackground="#45a049", command=process\_input)

send\_button.grid(row=1, column=1, padx=5, pady=5)

# Display initial response

chat\_box.config(state=tk.NORMAL)

chat\_box.insert(tk.END, "Bot: " + initial\_response() + '\n\n')

chat\_box.config(state=tk.DISABLED)

# Function to monitor user input and reset the inactivity timer

def on\_typing(event):

    global inactivity\_timer

    if inactivity\_timer:

        root.after\_cancel(inactivity\_timer)

    inactivity\_timer = root.after(30000, handle\_inactivity)  # 30 seconds timeout

# Bind the typing event to monitor user input

entry.bind("<KeyRelease>", on\_typing)

# Bind Enter key to trigger process\_input

entry.bind("<Return>", process\_input)

# Run the Tkinter main loop

root.mainloop()

import random

elements = {

    "H": "Hydrogen",

    "He": "Helium",

    "Li": "Lithium",

    "Be": "Beryllium",

    "B": "Boron",

    "C": "Carbon",

    "N": "Nitrogen",

    "O": "Oxygen",

    "F": "Fluorine",

    "Ne": "Neon",

    "Na": "Sodium",

    "Mg": "Magnesium",

    "Al": "Aluminum",

    "Si": "Silicon",

    "P": "Phosphorus",

    "S": "Sulfur",

    "Cl": "Chlorine",

    "Ar": "Argon",

    "K": "Potassium",

    "Ca": "Calcium",

    "Sc": "Scandium",

    "Ti": "Titanium",

    "V": "Vanadium",

    "Cr": "Chromium",

    "Mn": "Manganese",

    "Fe": "Iron",

    "Co": "Cobalt",

    "Ni": "Nickel",

    "Cu": "Copper",

    "Zn": "Zinc",

    "Ga": "Gallium",

    "Ge": "Germanium",

    "As": "Arsenic",

    "Se": "Selenium",

    "Br": "Bromine",

    "Kr": "Krypton",

    "Rb": "Rubidium",

    "Sr": "Strontium",

    "Y": "Yttrium",

    "Zr": "Zirconium",

    "Nb": "Niobium",

    "Mo": "Molybdenum",

    "Tc": "Technetium",

    "Ru": "Ruthenium",

    "Rh": "Rhodium",

    "Pd": "Palladium",

    "Ag": "Silver",

    "Cd": "Cadmium",

    "In": "Indium",

    "Sn": "Tin",

    "Sb": "Antimony",

    "I": "Iodine",

    "Te": "Tellurium",

    "Xe": "Xenon",

    "Cs": "Cesium",

    "Ba": "Barium",

    "La": "Lanthanum",

    "Ce": "Cerium",

    "Pr": "Praseodymium",

    "Nd": "Neodymium",

    "Pm": "Promethium",

    "Sm": "Samarium",

    "Eu": "Europium",

    "Gd": "Gadolinium",

    "Tb": "Terbium",

    "Dy": "Dysprosium",

    "Ho": "Holmium",

    "Er": "Erbium",

    "Tm": "Thulium",

    "Yb": "Ytterbium",

    "Lu": "Lutetium",

    "Hf": "Hafnium",

    "Ta": "Tantalum",

    "W": "Tungsten",

    "Re": "Rhenium",

    "Os": "Osmium",

    "Ir": "Iridium",

    "Pt": "Platinum",

    "Au": "Gold",

    "Hg": "Mercury",

    "Tl": "Thallium",

    "Pb": "Lead",

    "Bi": "Bismuth",

    "Po": "Polonium",

    "At": "Astatine",

    "Rn": "Radon",

    "Fr": "Francium",

    "Ra": "Radium",

    "Ac": "Actinium",

    "Th": "Thorium",

    "Pa": "Protactinium",

    "U": "Uranium",

    "Np": "Neptunium",

    "Pu": "Plutonium",

    "Am": "Americium",

    "Cm": "Curium",

    "Bk": "Berkelium",

    "Cf": "Californium",

    "Es": "Einsteinium",

    "Fm": "Fermium",

    "Md": "Mendelevium",

    "No": "Nobelium",

    "Lr": "Lawrencium",

    "Rf": "Rutherfordium",

    "Db": "Dubnium",

    "Sg": "Seaborgium",

    "Bh": "Bohrium",

    "Hs": "Hassium",

    "Mt": "Meitnerium",

    "Ds": "Darmstadtium",

    "Rg": "Roentgenium",

    "Cn": "Copernicium",

    "Nh": "Nihonium",

    "Fl": "Flerovium",

    "Mc": "Moscovium",

    "Lv": "Livermorium",

    "Ts": "Tennessine",

    "Og": "Oganesson"

}

prefixes = ["mono", "di", "tri", "tetra", "penta", "hexa", "hepta", "octa", "nona", "deca"]

unrecognized\_responses = [

    "Hmm, I'm not sure about that. Could you try again?",

    "Sorry, I didn't quite catch that. Please enter a chemical name, symbol, or formula.",

    "I couldn't understand that. Could you rephrase it?",

    "Oops, I couldn't recognize that input. Try a chemical name, symbol, or formula.",

    "I'm not familiar with that. Could you please give me a chemical name or symbol?",

    "I didn't quite get that. Please check your input.",

    "Can you try again with a valid chemical name or symbol?",

    "I wasn't able to recognize that. Please provide a chemical name or formula.",

    "Sorry, I didn't understand your input. Please enter a chemical name or formula.",

    "Hmm, that doesn't seem to be a valid input. Please try again.",

    "I couldn't recognize that. Could you enter a valid chemical symbol or name?",

    "Sorry, I don't know that. Try using a chemical name or symbol.",

    "Not sure what that is. Could you enter a valid chemical name or formula?",

    "I didn't catch that. Could you give me a valid chemical name or symbol?",

    "I don't quite understand that. Try entering a chemical name or formula.",

    "Sorry, that input doesn't match anything I know. Try again.",

    "I couldn't process that. Can you give me a chemical name or formula?",

    "That doesn't seem right. Please enter a valid chemical name or formula.",

    "I'm afraid I don't understand that. Could you provide a valid chemical name or symbol?",

    "I didn't quite understand. Could you try again with a chemical name or symbol?"

]

initial\_responses = [

    "Hello! I'm your chemical chatbot. How can I assist you today?",

    "Hi there! Ask me anything about chemical symbols, formulas, or names.",

    "Welcome to the Chemical Chatbot! Enter a chemical name, symbol, or formula.",

    "Hello! I can help you find chemical names, symbols, or formulas. What would you like to know?",

]

# Function to get name from symbol

def get\_name\_from\_symbol(symbol):

    """Get the name of the element from its symbol."""

    symbol = symbol.capitalize()  # Capitalize to handle case insensitivity

    return elements.get(symbol, "Chemical symbol not found")

# Function to get symbol from name

def get\_symbol\_from\_name(name):

    """Get the symbol of the element from its name."""

    name = name.lower()  # Convert name to lowercase for case insensitivity

    for symbol, element\_name in elements.items():

        if element\_name.lower() == name:

            return symbol

    return "Chemical name not found"

def unrecognized\_response():

    return random.choice(unrecognized\_responses)

def initial\_response():

    return random.choice(initial\_responses)

def get\_elements():

    return elements

def get\_prefixes():

    return prefixes