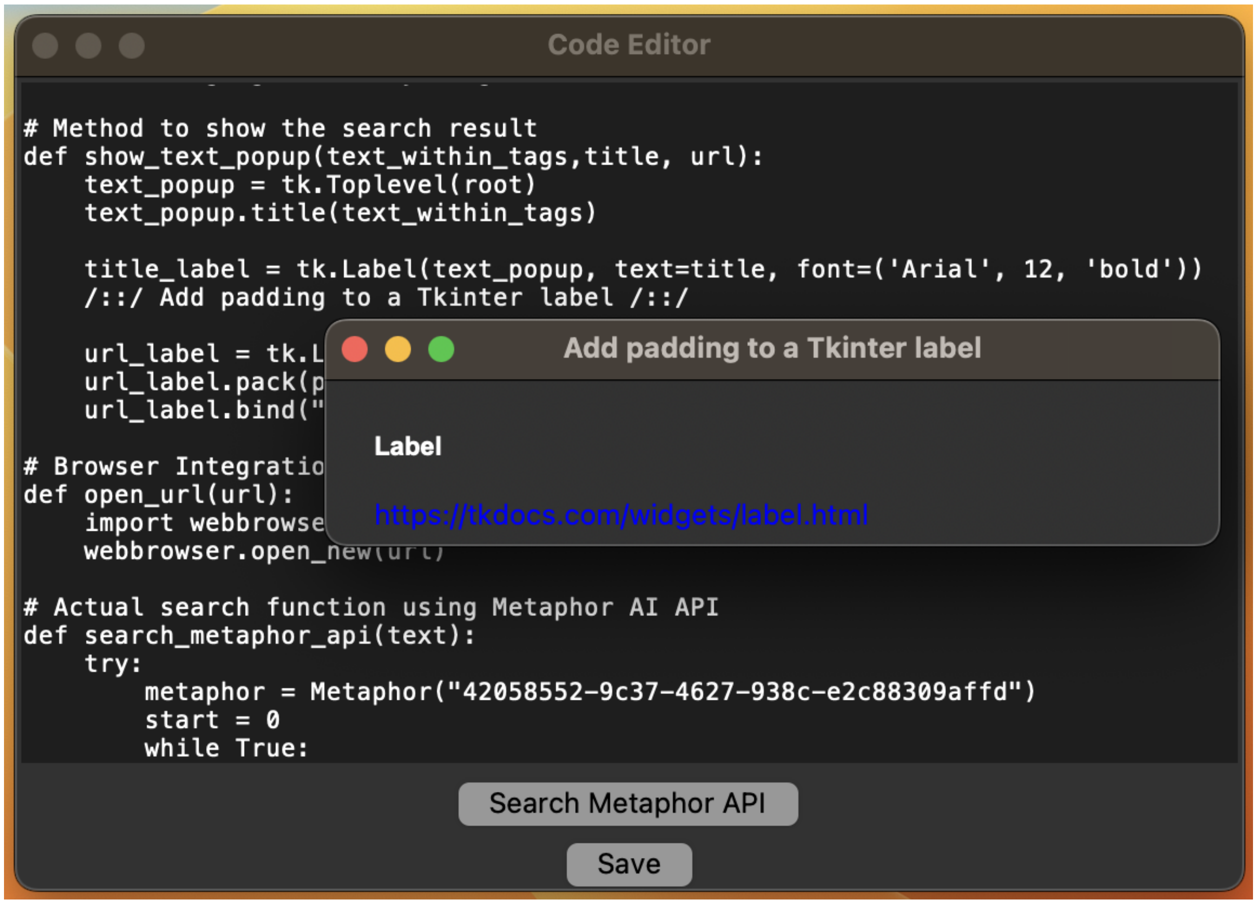
MDE: Metaphor Development Environment

Description

This Python program creates a simple code editor using the Tkinter library and Metaphor AI. It offers several features including code indentation, integration with Metaphor AI for contextual search to assist in writing code, and the ability to save files with specific language extensions.



```
# Method to show the search result
def show_text_popup(text_within_tags,title, url):
    text_popup = tk.Toplevel(root)
    text_popup.title(text_within_tags)

    title_label = tk.Label(text_popup, text=title, font=('Arial', 12, 'bold'))
    /::/ Add padding to a Tkinter label /::/

    url_label = tk.L
    url_label.pack(p
    url_label.bind("

# Browser Integratio
def open_url(url):
    import webbrowse
    webbrowser.open_new(url)

# Actual search function using Metaphor AI API
def search_metaphor_api(text):
    try:
        metaphor = Metaphor("42058552-9c37-4627-938c-e2c88309affd")
        start = 0
        while True:
```

Add padding to a Tkinter label

Label

https://tkdocs.com/widgets/label.html

Key Components

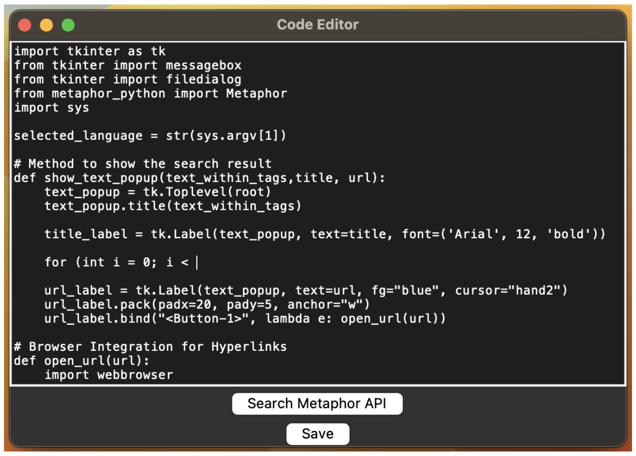
Metaphor AI Integration:

The program integrates with the Metaphor AI for contextual search. It allows users to select a specific programming language via command-line arguments (`sys.argv[1]`).
The `search_metaphor_api` function conducts searches using Metaphor AI, utilizing the selected language in conjunction with the provided text within `/::/` tags.

```
def search_metaphor_api(text):
    try:
        metaphor = Metaphor("42058552-9c37-4627-938c-e2c88309affd")
        start = 0
        while True:
            start = text.find('/::/', start)
            if start == -1:
                break
            end = text.find('/::/', start + 4)
            if end != -1:
                text_within_tags = text[start+4:end]
                start = end + 4

                search_response = metaphor.search(selected_language +
                      text_within_tags, num_results=1, use_autoprompt=True,)

                contents_response = search_response.get_contents()

                for content in contents_response.contents:
                    title = content.title
                    url = content.url

                    show_text_popup(text_within_tags,title, url)
             else:
                 break

    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {str(e)}")
```

Code Editor UI:

The Tkinter library is used to create a user interface for the code editor.
The core of the editor is the `text_widget` object, which is a multi-line text widget where users can input and edit code.



```
import tkinter as tk
from tkinter import messagebox
from tkinter import filedialog
from metaphor_python import Metaphor
import sys

selected_language = str(sys.argv[1])

# Method to show the search result
def show_text_popup(text_within_tags,title, url):
    text_popup = tk.Toplevel(root)
    text_popup.title(text_within_tags)

    title_label = tk.Label(text_popup, text=title, font=('Arial', 12, 'bold'))

    for (int i = 0; i < |

    url_label = tk.Label(text_popup, text=url, fg="blue", cursor="hand2")
    url_label.pack(padx=20, pady=5, anchor="w")
    url_label.bind("<Button-1>", lambda e: open_url(url))

# Browser Integration for Hyperlinks
def open_url(url):
    import webbrowser
```

Auto Indentation:

The `auto_indent` function provides basic auto-indentation behavior. When the user presses Enter or the keypad Enter key, it checks for braces `{}` and inserts an additional tab for indentation.

```
def auto_indent(event):
    index = text_widget.index(tk.INSERT)
    line = text_widget.get(index.split('.')[0] + '.0', index)

    if event.keysym in ["Return", "KP_Enter"]:
        leading_spaces = len(line) - len(line.lstrip())

        if "{" in line:
            text_widget.insert(tk.INSERT, '\t\n}')
            text_widget.mark_set(tk.INSERT,index)
```

Search Result Pop-up:

The `show_text_popup` function displays Metaphor AI's search results in a pop-up bubble. It includes a title, URL (as a clickable link), and an extract.

```
def show_text_popup(text_within_tags,title, url):
    text_popup = tk.Toplevel(root)
    text_popup.title(text_within_tags)
    title_label = tk.Label(text_popup, text=title, font=('Arial', 12, 'bold'))
    title_label.pack(padx=20, pady=(20,5), anchor="w")

    url_label = tk.Label(text_popup, text=url, fg="blue", cursor="hand2")
    url_label.pack(padx=20, pady=5, anchor="w")
    url_label.bind("<Button-1>", lambda e: open_url(url))
```

Hyperlink Integration:

The `open_url` function opens URLs in the default web browser when a user clicks on them.

```
def open_url(url):
    import webbrowser
    webbrowser.open_new(url)
```

Buttons:

The program includes two buttons: "*Search Metaphor API*" triggers the Metaphor API search, while "*Save*" allows the user to save the content.

File Saving Functionality:

The `save_file` function allows users to save the content of the editor. It opens a file dialog to choose the location and name of the file, based on the selected language's extension. This program has integrated Java, Python, and C++ saving, but any language can be saved.

```
def save_file():
    file_path = filedialog.asksaveasfilename(defaultextension="." +
            str(selected_language), filetypes=[("Text files", "*.txt"),
            ("Java Program", "*.java"), ("C++ Program", "*.cpp"),
            ("Python Program", "*.py")])
    if file_path:
        try:
            with open(file_path, 'w') as file:
            file.write(text_widget.get("1.0", 'end-1c'))
        except Exception as e:
            messagebox.showerror("Error", f"An error occurred while saving the
                file: {str(e)}")
```

Usage

The program can be executed from the command line by providing a selected language as an argument (e.g., `python3 code_editor.py cpp`). This language selection impacts the search queries sent to the Metaphor API.

Conclusion

Overall, this program provides a functional code editor with additional features using Metaphor AI integration. It's a versatile tool that can be adapted for various programming languages. In the future, advanced IDE-like features could potentially be seamlessly integrated using advanced Metaphor AI technologies. This could include sophisticated syntax highlighting algorithms, intelligent code completion powered by natural language processing, real-time error detection utilizing advanced pattern recognition, and dynamic code folding based on contextual understanding. Additionally, version control, project management, and other complex functionalities may be streamlined and enhanced through the utilization of cutting-edge AI capabilities. As Metaphor AI continues to evolve and advance, it has the potential to revolutionize the way developers interact with code editors, making them even more powerful and intuitive tools for software development.