**Assignment**

**CSA0805 – Python Programming**

|  |  |
| --- | --- |
| **Register Number** | **192325122** |
| **Name** | **P. Lohith Krishna** |

**Title: Text File Editor**

**Problem Statement: Implement a Python program that provides a text editor interface for editing plain text files, supporting features such as syntax highlighting, line numbering, search and replace, and file saving**

**Code:**

**import tkinter as tk**

**from tkinter import filedialog, messagebox**

**from pygments import lex**

**from pygments.lexers import PythonLexer**

**from pygments.styles import get\_style\_by\_name**

**class TextEditor:**

**def \_\_init\_\_(self, root):**

**self.root = root**

**self.root.title("Simple Text Editor")**

**self.current\_file = None**

**# Frame to hold text area and line numbers**

**self.text\_frame = tk.Frame(self.root)**

**self.text\_frame.pack(fill=tk.BOTH, expand=1)**

**# Line numbers**

**self.line\_numbers = tk.Text(self.text\_frame, width=4, padx=3, takefocus=0, border=0, background='lightgrey', state='disabled', wrap='none')**

**self.line\_numbers.pack(side=tk.LEFT, fill=tk.Y)**

**# Creating a Text widget**

**self.text\_area = tk.Text(self.text\_frame, wrap='word', undo=True)**

**self.text\_area.pack(side=tk.RIGHT, fill=tk.BOTH, expand=1)**

**self.text\_area.bind("<KeyRelease>", self.on\_key\_release)**

**self.text\_area.bind("<MouseWheel>", self.update\_line\_numbers)**

**self.text\_area.bind("<Button-1>", self.update\_line\_numbers)**

**# Adding a menu bar**

**self.menu\_bar = tk.Menu(self.root)**

**self.root.config(menu=self.menu\_bar)**

**# File menu**

**self.file\_menu = tk.Menu(self.menu\_bar, tearoff=0)**

**self.file\_menu.add\_command(label="Open", command=self.open\_file)**

**self.file\_menu.add\_command(label="Save", command=self.save\_file)**

**self.file\_menu.add\_command(label="Save As", command=self.save\_file\_as)**

**self.file\_menu.add\_separator()**

**self.file\_menu.add\_command(label="Exit", command=self.root.quit)**

**self.menu\_bar.add\_cascade(label="File", menu=self.file\_menu)**

**# Edit menu**

**self.edit\_menu = tk.Menu(self.menu\_bar, tearoff=0)**

**self.edit\_menu.add\_command(label="Find", command=self.find\_text)**

**self.edit\_menu.add\_command(label="Replace", command=self.replace\_text)**

**self.menu\_bar.add\_cascade(label="Edit", menu=self.edit\_menu)**

**self.update\_line\_numbers()**

**def open\_file(self):**

**file\_path = filedialog.askopenfilename(filetypes=[("Text Files", "\*.txt"), ("All Files", "\*.\*")])**

**if file\_path:**

**with open(file\_path, "r") as file:**

**self.text\_area.delete(1.0, tk.END)**

**self.text\_area.insert(tk.END, file.read())**

**self.root.title(f"Simple Text Editor - {file\_path}")**

**self.current\_file = file\_path**

**self.update\_line\_numbers()**

**self.highlight\_syntax()**

**def save\_file(self):**

**if self.current\_file:**

**with open(self.current\_file, "w") as file:**

**file.write(self.text\_area.get(1.0, tk.END))**

**messagebox.showinfo("Save", "File saved successfully!")**

**else:**

**self.save\_file\_as()**

**def save\_file\_as(self):**

**file\_path = filedialog.asksaveasfilename(defaultextension=".txt", filetypes=[("Text Files", "\*.txt"), ("All Files", "\*.\*")])**

**if file\_path:**

**with open(file\_path, "w") as file:**

**file.write(self.text\_area.get(1.0, tk.END))**

**self.root.title(f"Simple Text Editor - {file\_path}")**

**self.current\_file = file\_path**

**messagebox.showinfo("Save As", "File saved successfully!")**

**def update\_line\_numbers(self, event=None):**

**line\_numbers = "\n".join(str(i) for i in range(1, int(self.text\_area.index('end').split('.')[0])))**

**self.line\_numbers.config(state='normal')**

**self.line\_numbers.delete(1.0, tk.END)**

**self.line\_numbers.insert(1.0, line\_numbers)**

**self.line\_numbers.config(state='disabled')**

**def on\_key\_release(self, event=None):**

**self.update\_line\_numbers()**

**self.highlight\_syntax()**

**def find\_text(self):**

**self.find\_dialog = tk.Toplevel(self.root)**

**self.find\_dialog.title("Find")**

**self.find\_dialog.geometry("300x100")**

**tk.Label(self.find\_dialog, text="Find:").pack(side=tk.LEFT)**

**self.find\_entry = tk.Entry(self.find\_dialog, width=25)**

**self.find\_entry.pack(side=tk.LEFT, padx=10)**

**tk.Button(self.find\_dialog, text="Find", command=self.find).pack(side=tk.LEFT)**

**def find(self):**

**start = "1.0"**

**word = self.find\_entry.get()**

**self.text\_area.tag\_remove('found', '1.0', tk.END)**

**if word:**

**while True:**

**start = self.text\_area.search(word, start, stopindex=tk.END)**

**if not start:**

**break**

**end = f"{start}+{len(word)}c"**

**self.text\_area.tag\_add('found', start, end)**

**self.text\_area.tag\_config('found', background='yellow')**

**start = end**

**def replace\_text(self):**

**self.replace\_dialog = tk.Toplevel(self.root)**

**self.replace\_dialog.title("Replace")**

**self.replace\_dialog.geometry("300x100")**

**tk.Label(self.replace\_dialog, text="Find:").pack(side=tk.LEFT)**

**self.find\_entry = tk.Entry(self.replace\_dialog, width=25)**

**self.find\_entry.pack(side=tk.LEFT, padx=10)**

**tk.Label(self.replace\_dialog, text="Replace:").pack(side=tk.LEFT)**

**self.replace\_entry = tk.Entry(self.replace\_dialog, width=25)**

**self.replace\_entry.pack(side=tk.LEFT, padx=10)**

**tk.Button(self.replace\_dialog, text="Replace", command=self.replace).pack(side=tk.LEFT)**

**def replace(self):**

**word = self.find\_entry.get()**

**replacement = self.replace\_entry.get()**

**content = self.text\_area.get(1.0, tk.END)**

**new\_content = content.replace(word, replacement)**

**self.text\_area.delete(1.0, tk.END)**

**self.text\_area.insert(1.0, new\_content)**

**self.highlight\_syntax()**

**def highlight\_syntax(self):**

**code = self.text\_area.get("1.0", tk.END)**

**tokens = lex(code, PythonLexer())**

**self.text\_area.mark\_set("range\_start", "1.0")**

**for token\_type, token in tokens:**

**self.text\_area.mark\_set("range\_end", "range\_start + %dc" % len(token))**

**self.text\_area.tag\_add(str(token\_type), "range\_start", "range\_end")**

**self.text\_area.mark\_set("range\_start", "range\_end")**

**# Example: Configuring colors (you can define more for different token types)**

**self.text\_area.tag\_configure('Token.Keyword', foreground='blue')**

**self.text\_area.tag\_configure('Token.Name.Function', foreground='green')**

**self.text\_area.tag\_configure('Token.Comment', foreground='grey', slant='italic')**

**if \_\_name\_\_ == "\_\_main\_\_":**

**root = tk.Tk()**

**editor = TextEditor(root)**

**root.mainloop()**

**Output Screen Shots:**

**Conclusion:**

**The Python-based text editor we've implemented provides a functional and user-friendly interface for editing plain text files. It includes essential features such as**:

* **Syntax Highlighting**: **Enhanced code readability, especially for Python files.**
* **Line Numbering**: **A helpful feature for keeping track of code lines, especially in larger files.**
* **Search and Replace**: **Simplifies text manipulation within documents**.
* **File Operations**: **Supports opening, saving, and saving files with a new name.**

**This project demonstrates how you can create a powerful text editor using Python's tkinter for the GUI and Pygments for syntax highlighting. While it's a basic implementation, it serves as a strong foundation for further enhancements, such as adding support for more file types, advanced text editing features, or even more sophisticated syntax highlighting.**