

Submitted By :

Umar Farooq 2022Ag-8043

Sajid Hameed 2022-Ag-8076

Alam Sher 2022-Ag-8081

Submitted To :

Mam Nabeela Ashraf

Subject :

SE – 503

Documentation for "MY Notepad" Python Project

This documentation covers the **MY Notepad** application, which is a simple text editor built using Python's **tkinter** library. It provides core functionalities such as creating a new file, opening an existing file, saving files, and other editing features like undo, redo, cut, copy, paste, delete, and changing text colors.

**Overview:**

The **MY Notepad** application is a GUI-based text editor with the following main features:

* **File Operations**: New, Open, Save, Save As, Exit
* **Edit Operations**: Undo, Redo, Cut, Copy, Paste, Delete
* **Color Customization**: Change the background and foreground colors of the text area
* **Keyboard Shortcuts**: Common shortcuts like Ctrl + N (New), Ctrl + O (Open), Ctrl + S (Save), etc.

The application uses **tkinter**, Python's standard GUI library, for creating the user interface, and it relies on the **file dialog** and **color chooser** modules for file operations and color picking, respectively.

**Detailed Breakdown:**

1. **Class Definition:**

python

class MYNotepad:

The MYNotepad class defines the functionality of the text editor. It initializes the GUI components and handles user interactions.

### 2. ****Constructor (****\_\_init\_\_ ****method)****

python

def \_\_init\_\_(self, master):

The constructor is responsible for setting up the GUI components:

* **master:** The root window passed when creating the tkinter instance.
* **self.master.title("My Notepad"):** Sets the title of the main window to "My Notepad".
* **self.current\_file = None:** Initializes current\_file to None, as there is no file open initially.
* **self.txt:** The main text widget for the text editor.
  + Configures various properties such as padding, word wrapping, and undo functionality.
* Menu creation:
  + **File menu**: Includes commands for creating a new file, opening a file, saving a file, and exiting the application.
  + **Edit menu**: Includes undo, redo, cut, copy, paste, and delete commands.
  + **Color menu**: Includes options for changing the background and foreground color of the text.

### 3. ****File Menu Commands:****

#### **New File (new\_file method):**

python

def new\_file(self):

if self.txt.edit\_modified():

if not self.confirm\_save():

return

self.txt.delete(1.0, END)

self.current\_file = None

self.txt.edit\_modified(False)

* Clears the current text area and resets the current\_file to None.
* If there are unsaved changes, prompts the user to confirm whether they want to save before starting a new file.

#### **Open File (open\_file method):**

python

def open\_file(self):

if self.txt.edit\_modified():

if not self.confirm\_save():

return

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

if file\_path:

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

content = file.read()

self.txt.delete(1.0, END)

self.txt.insert(END, content)

self.current\_file = file\_path

self.txt.edit\_modified(False)

* Prompts the user to select a file using filedialog.askopenfilename().
* If the user selects a file, it reads its contents and loads them into the text widget.

#### **Save File (save\_file method):**

python

def save\_file(self):

if self.current\_file:

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

content = self.txt.get(1.0, END)

file.write(content.strip())

self.txt.edit\_modified(False)

else:

self.saveas\_file()

* If a file is already open (current\_file is set), it saves the content to that file.
* If no file is open, it calls the saveas\_file method to prompt the user to select a location to save.

#### **Save As File (saveas\_file method):**

Python

s

def saveas\_file(self):

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

if file\_path:

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

content = self.txt.get(1.0, END)

file.write(content.strip())

self.current\_file = file\_path

self.txt.edit\_modified(False)

* Prompts the user to select a location to save the file using filedialog.asksaveasfilename().
* Saves the current content to the selected file.

#### **Exit Application (exit\_app method):**

Python

def exit\_app(self):

if self.txt.edit\_modified():

answer = messagebox.askyesnocancel("Save Changes", "Do you want to save changes before exiting?")

if answer: # Yes, save and exit

self.save\_file()

self.master.quit()

elif answer is None: # Cancel, do nothing

return

else: # No, exit without saving

self.master.quit()

else:

self.master.quit() # No changes, exit directly

* Prompts the user to confirm whether they want to save unsaved changes before quitting the application.
* If there are no unsaved changes, the application quits directly.

### 4. ****Edit Menu Commands:****

#### **Undo (undo\_file method):**

python

def undo\_file(self):

try:

self.txt.edit\_undo()

except TclError:

pass

* Attempts to undo the last edit. If an error occurs, it is ignored.

#### **Redo (redo\_file method):**

python

def redo\_file(self):

try:

self.txt.edit\_redo()

except TclError:

pass

* Attempts to redo the last undone action. Errors are ignored.

#### **Cut (cut\_file method):**

python

def cut\_file(self):

self.copy\_file()

self.txt.delete(SEL\_FIRST, SEL\_LAST)

* Copies the selected text to the clipboard and then deletes it from the text widget.

#### **Copy (copy\_file method):**

python

def copy\_file(self):

try:

self.master.clipboard\_clear()

text = self.txt.get(SEL\_FIRST, SEL\_LAST)

self.master.clipboard\_append(text)

except TclError:

pass

* Copies the selected text to the clipboard.

#### **Paste (paste\_file method):**

python

def paste\_file(self):

try:

text = self.master.clipboard\_get()

self.txt.insert(INSERT, text)

except TclError:

pass

* Pastes text from the clipboard at the current cursor position.

#### **Delete (delete\_text method):**

python

def delete\_text(self):

try:

self.txt.delete(SEL\_FIRST, SEL\_LAST)

except TclError:

pass

* Deletes the selected text.

### 5. ****Color Menu Commands:****

#### **Change Background Color (change\_back\_color method):**

python

def change\_back\_color(self):

color = colorchooser.askcolor()[1] # Returns a tuple, we need the color code

if color:

self.txt.config(bg=color)

* Allows the user to choose a background color for the text area using colorchooser.askcolor().

#### **Change Foreground Color (change\_fore\_color method):**

python

def change\_fore\_color(self):

color = colorchooser.askcolor()[1]

if color:

self.txt.config(fg=color)

* Allows the user to choose a text color (foreground color) using colorchooser.askcolor().

## Keyboard Shortcuts:

The application binds several common keyboard shortcuts for ease of use:

* **Ctrl + N**: Create a new file
* **Ctrl + O**: Open an existing file
* **Ctrl + S**: Save the current file
* **Ctrl + Z**: Undo the last action
* **Ctrl + Y**: Redo the last undone action
* **Ctrl + X**: Cut the selected text
* **Ctrl + C**: Copy the selected text
* **Ctrl + V**: Paste the text from the clipboard

## Best Practices and Recommendations:

1. **Error Handling**: Some functions like undo and redo handle errors using try...except blocks to avoid crashes due to missing actions.
2. **Code Modularity**: The separation of concerns (file operations, text editing, color changes) makes the code more maintainable and readable.
3. **User Prompts**: The use of confirmation dialogs (messagebox.askyesnocancel()) ensures that the user is prompted when performing potentially destructive actions, such as closing the application with unsaved changes.

## Conclusion:

This **MY Notepad** project demonstrates a basic text editor with key file operations and editing features built using **tkinter** in Python. It provides a functional interface for users to create, edit, and save text files while supporting color customization and common keyboard shortcuts. The structure of the code ensures it is both extendable and maintainable.

## Developed By:

## Umar Farooq <https://github.com/UmarFarooq9720>

## Alam Sher <https://github.com/AlamSher125>

## Sajid Hameed <https://github.com/SajidHameed223/>

## Resources:

## We use this Youtube video for creating this project :

<https://youtu.be/d1HyXxeCRg8?feature=shared>