PD LAB

ASSIGNMENT - 5

Name: Raunak Thanawala

Registration Number: 231070051

Branch: Computer Engineering

Batch: 3

**Aim:-**

Create text editor using tkinter and file handling functions in Python

**Theory:-**

Python provides alot of in built functions for us to handle files.Some of these functions are:

* Opening a file:
  + To open a file we use the open() function which returns a file object.
  + We also need to specify which mode we want to open the file in
  + Eg. file = open(“filename.txt”, “mode”)
* Closing a file:
  + We close files so that we can free up system resources
  + We can do this with file.close()
  + Or we can use with statement as:
    - with open(“filename.txt”, “mode”) as file:
* Reading a file:
  + We can read files with multiple functions:
    - read(size) reads a specified number of bytes or if size is not given the whole file.
    - readline() reads one line at a time
    - readlines() returns a list of all available lines
* Writing a file:
  + There are 2 functions we can use for writing:
    - write(string) writes the given string to the file
    - writelines(list\_of\_strings) writes the given list of strings to the file
* Appending to a file:
  + To append onto files we have to open the file in mode - ‘a’ which keeps existing content the same.

**Code and Output:**

import tkinter as tk

from tkinter import filedialog, messagebox, simpledialog, colorchooser, font

import random

*# Define the main application class*

class TextEditor:

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

self.root = root

self.root.title("Text Editor")

self.root.geometry("1000x500")

*# Initialize default font settings*

self.current\_font\_name = "Consolas" *# Corrected font name*

self.current\_font\_size = 12

self.current\_font\_color = "black"

self.current\_theme = "light"

*# Create a Text widget*

self.text\_area = tk.Text(root, wrap='word', undo=True, font=(self.current\_font\_name, self.current\_font\_size), fg=self.current\_font\_color)

self.text\_area.pack(expand=True, fill='both')

*# Create a Menu bar*

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

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

*# Create File menu*

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

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

self.file\_menu.add\_command(label="New", command=self.new\_file)

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\_as\_file)

self.file\_menu.add\_separator()

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

*# Create View menu*

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

self.menu\_bar.add\_cascade(label="View", menu=self.view\_menu)

self.view\_menu.add\_command(label="Find", command=self.find\_word)

self.view\_menu.add\_command(label="Replace", command=self.replace\_word)

*# Create Format menu*

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

self.menu\_bar.add\_cascade(label="Format", menu=self.format\_menu)

self.format\_menu.add\_command(label="Font", command=self.show\_font\_name\_menu)

self.format\_menu.add\_command(label="Font Size", command=self.show\_font\_size\_menu)

self.format\_menu.add\_command(label="Font Color", command=self.change\_font\_color)

*# Create Themes menu*

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

self.menu\_bar.add\_cascade(label="Themes", menu=self.theme\_menu)

self.theme\_menu.add\_command(label="Light", command=self.light\_theme)

self.theme\_menu.add\_command(label="Dark", command=self.dark\_theme)

self.theme\_menu.add\_command(label="Rainbow", command=self.rainbow\_theme)

*# Initialize file path*

self.file\_path = None

*# Define font options*

self.font\_names = sorted(list(font.families()))

self.font\_sizes = list(range(8, 73, 2)) *# Example sizes: 8, 10, 12, ..., 72*

*# Create dropdown menus for font type and size*

self.font\_name\_var = tk.StringVar(value=self.current\_font\_name)

self.font\_size\_var = tk.IntVar(value=self.current\_font\_size)

*# Create a Frame for font options*

self.font\_frame = tk.Frame(root)

self.font\_frame.pack(fill='x')

*# Create Font Name dropdown*

self.font\_name\_button = tk.Button(self.font\_frame, text=self.current\_font\_name, command=self.show\_font\_name\_menu)

self.font\_name\_button.pack(side='left')

*# Create Font Size dropdown*

self.font\_size\_button = tk.Button(self.font\_frame, text=str(self.current\_font\_size), command=self.show\_font\_size\_menu)

self.font\_size\_button.pack(side='left')

*# Bind keyboard shortcuts*

self.bind\_shortcuts()

def new\_file(self):

if self.text\_area.get('1.0', tk.END+'-1c'):

if messagebox.askyesno("Save File", "Do you want to save changes?"):

self.save\_file()

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

self.file\_path = None

def open\_file(self):

file\_path = filedialog.askopenfilename(defaultextension=".txt",

filetypes=[("Text Files", "\*.txt"),

("All Files", "\*.\*")])

if file\_path:

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

content = file.read()

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

self.text\_area.insert(tk.END, content)

self.file\_path = file\_path

def save\_file(self):

if self.file\_path:

with open(self.file\_path, 'w') as file:

content = self.text\_area.get('1.0', tk.END+'-1c')

file.write(content)

else:

self.save\_as\_file()

def save\_as\_file(self):

file\_path = filedialog.asksaveasfilename(defaultextension=".txt",

filetypes=[("Text Files", "\*.txt"),

("All Files", "\*.\*")])

if file\_path:

self.file\_path = file\_path

self.save\_file()

def copy\_text(self, event=None):

self.text\_area.event\_generate("<<Copy>>")

def paste\_text(self, event=None):

self.text\_area.event\_generate("<<Paste>>")

def select\_all\_text(self, event=None):

self.text\_area.tag\_add(tk.SEL, "1.0", tk.END)

self.text\_area.mark\_set(tk.INSERT, "1.0")

self.text\_area.see(tk.INSERT)

return "break" *# Prevent default handling of this key event*

def find\_word(self):

*# Prompt the user for the word to find*

word = simpledialog.askstring("Find", "Enter the word to find:")

if word:

self.highlight\_word(word)

def replace\_word(self):

*# Prompt the user for the word to find and replace*

find\_word = simpledialog.askstring("Find", "Enter the word to find:")

if find\_word:

replace\_word = simpledialog.askstring("Replace", "Enter the replacement word:")

if replace\_word is not None:

self.replace\_text(find\_word, replace\_word)

def highlight\_word(self, word):

*# Remove previous highlight*

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

start = '1.0'

while True:

*# Find the next occurrence of the word*

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

if not pos:

break

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

*# Highlight the found word*

self.text\_area.tag\_add('highlight', pos, end)

start = end

*# Configure the highlight tag*

self.text\_area.tag\_configure('highlight', background='yellow')

def replace\_text(self, find\_word, replace\_word):

*# Replace all occurrences of find\_word with replace\_word*

start = '1.0'

while True:

*# Find the next occurrence of the word*

pos = self.text\_area.search(find\_word, start, stopindex=tk.END)

if not pos:

break

end = f"{pos}+{len(find\_word)}c"

*# Replace the found word*

self.text\_area.delete(pos, end)

self.text\_area.insert(pos, replace\_word)

start = pos + f"+{len(replace\_word)}c"

def show\_font\_name\_menu(self):

*# Create a window for font name selection*

font\_name\_menu = tk.Toplevel(self.root)

font\_name\_menu.title("Select Font")

font\_name\_menu.geometry("300x400") *# Adjust size to fit the number of fonts*

*# Create a canvas with scrollbar*

canvas = tk.Canvas(font\_name\_menu)

scrollbar = tk.Scrollbar(font\_name\_menu, orient="vertical", command=canvas.yview)

frame = tk.Frame(canvas)

*# Add fonts to the frame*

for font\_name in self.font\_names:

btn = tk.Button(frame, text=font\_name, command=lambda fn=font\_name: self.set\_font\_name(fn))

btn.pack(fill='x')

*# Pack canvas and scrollbar*

canvas.create\_window((0, 0), window=frame, anchor='nw')

canvas.configure(yscrollcommand=scrollbar.set)

scrollbar.pack(side='right', fill='y')

canvas.pack(side='left', fill='both', expand=True)

*# Update canvas scrollregion*

frame.update\_idletasks()

canvas.config(scrollregion=canvas.bbox('all'))

def show\_font\_size\_menu(self):

*# Create a window for font size selection*

font\_size\_menu = tk.Toplevel(self.root)

font\_size\_menu.title("Select Font Size")

font\_size\_menu.geometry("200x300") *# Adjust size to fit the number of sizes*

*# Create a canvas with scrollbar*

canvas = tk.Canvas(font\_size\_menu)

scrollbar = tk.Scrollbar(font\_size\_menu, orient="vertical", command=canvas.yview)

frame = tk.Frame(canvas)

*# Add sizes to the frame*

for size in self.font\_sizes:

btn = tk.Button(frame, text=str(size), command=lambda sz=size: self.set\_font\_size(sz))

btn.pack(fill='x')

*# Pack canvas and scrollbar*

canvas.create\_window((0, 0), window=frame, anchor='nw')

canvas.configure(yscrollcommand=scrollbar.set)

scrollbar.pack(side='right', fill='y')

canvas.pack(side='left', fill='both', expand=True)

*# Update canvas scrollregion*

frame.update\_idletasks()

canvas.config(scrollregion=canvas.bbox('all'))

def set\_font\_name(self, font\_name):

self.font\_name\_var.set(font\_name)

self.current\_font\_name = font\_name

self.apply\_font()

def set\_font\_size(self, font\_size):

self.font\_size\_var.set(font\_size)

self.current\_font\_size = font\_size

self.apply\_font()

def apply\_font(self):

*# Apply the selected font and size to the text area*

self.text\_area.config(font=(self.current\_font\_name, self.current\_font\_size))

def change\_font\_color(self):

*# Create a color chooser dialog*

color = colorchooser.askcolor(title="Choose Font Color")[1]

if color:

self.current\_font\_color = color

self.text\_area.config(fg=self.current\_font\_color)

def light\_theme(self):

self.text\_area.config(bg="white", fg="black")

self.current\_theme = "light"

def dark\_theme(self):

self.text\_area.config(bg="#301B3F", fg="white")

self.current\_theme = "dark"

def rainbow\_theme(self , event=None):

self.text\_area.config(bg=self.generate\_rgb(), fg=self.generate\_rgb())

self.current\_theme = "rainbow"

def bind\_shortcuts(self):

self.text\_area.bind("<Control-c>", self.copy\_text)

self.text\_area.bind("<Control-v>", self.paste\_text)

self.text\_area.bind("<Control-a>", self.select\_all\_text)

self.text\_area.bind("<Control-r>", self.rainbow\_theme)

def generate\_rgb(self):

r = random.randint(0, 255)

g = random.randint(0, 255)

b = random.randint(0, 255)

return f"#{r:02x}{g:02x}{b:02x}"

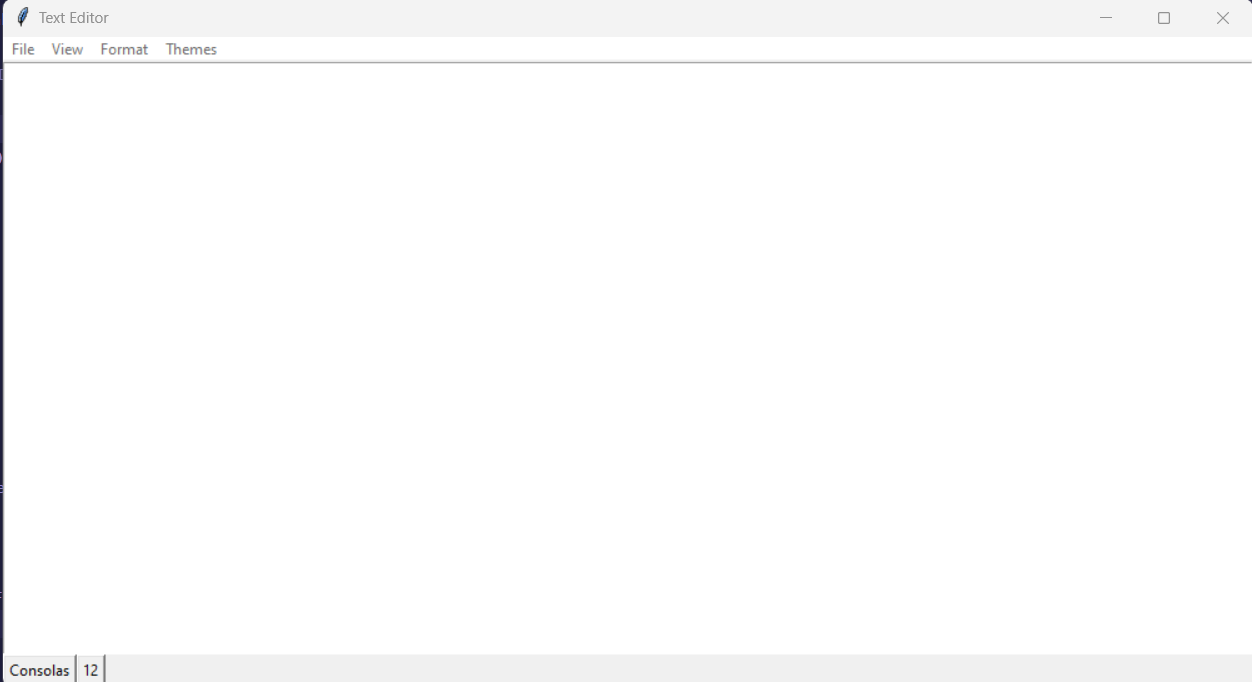
*# Create the main window*

root = tk.Tk()

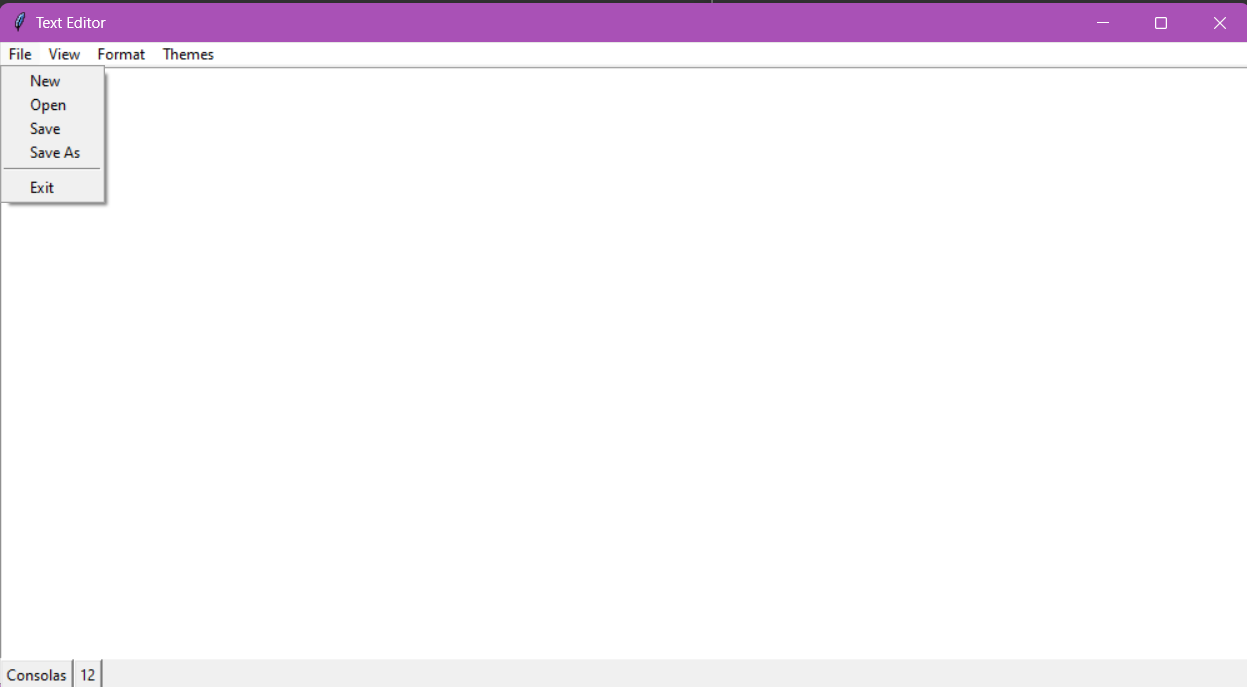
app = TextEditor(root)

root.mainloop()

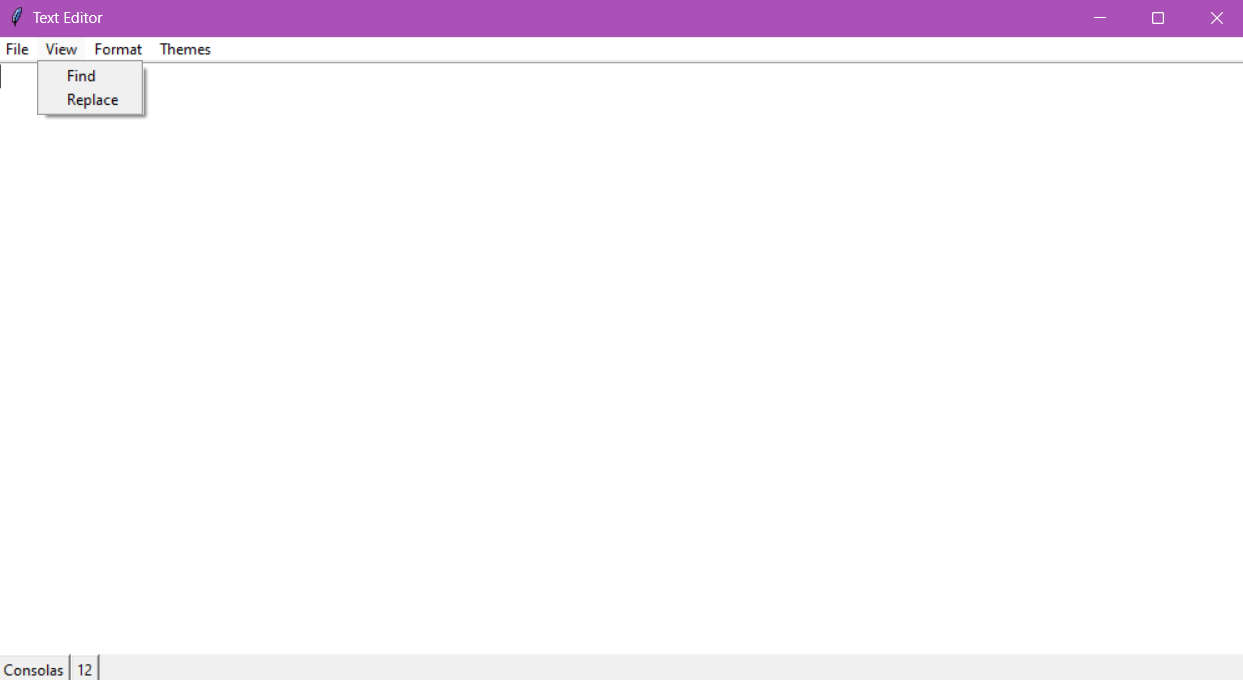
OUTPUT:



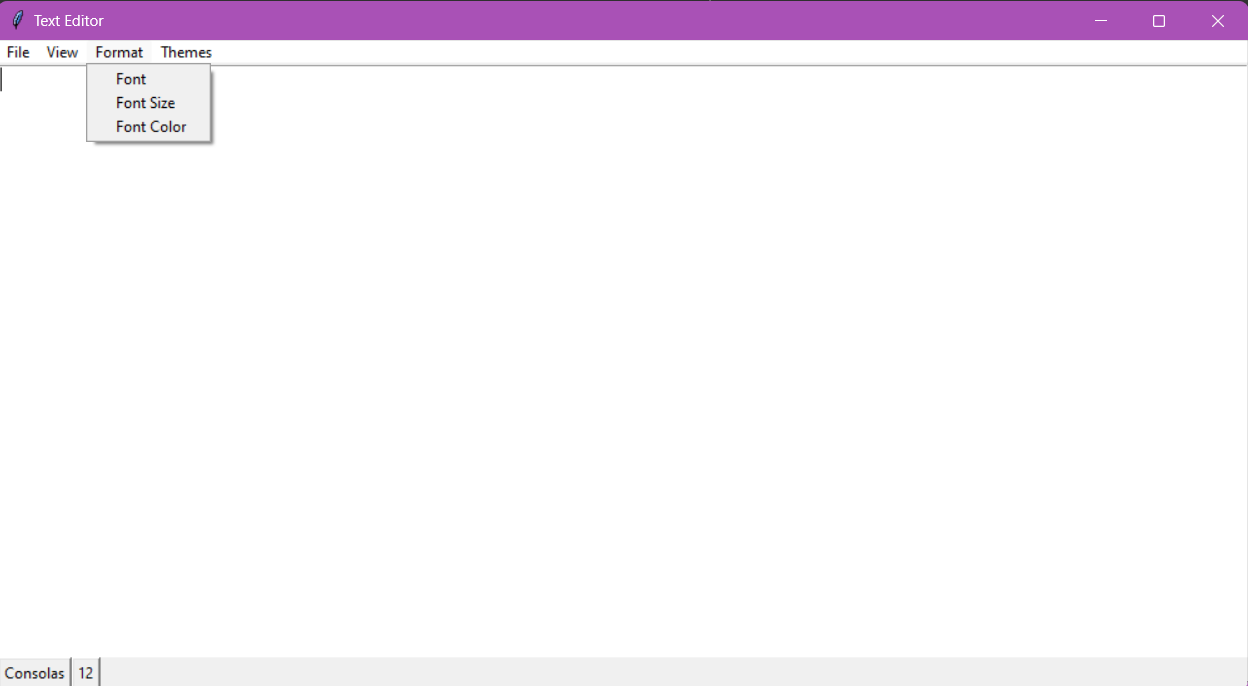
What you see when you run the code.



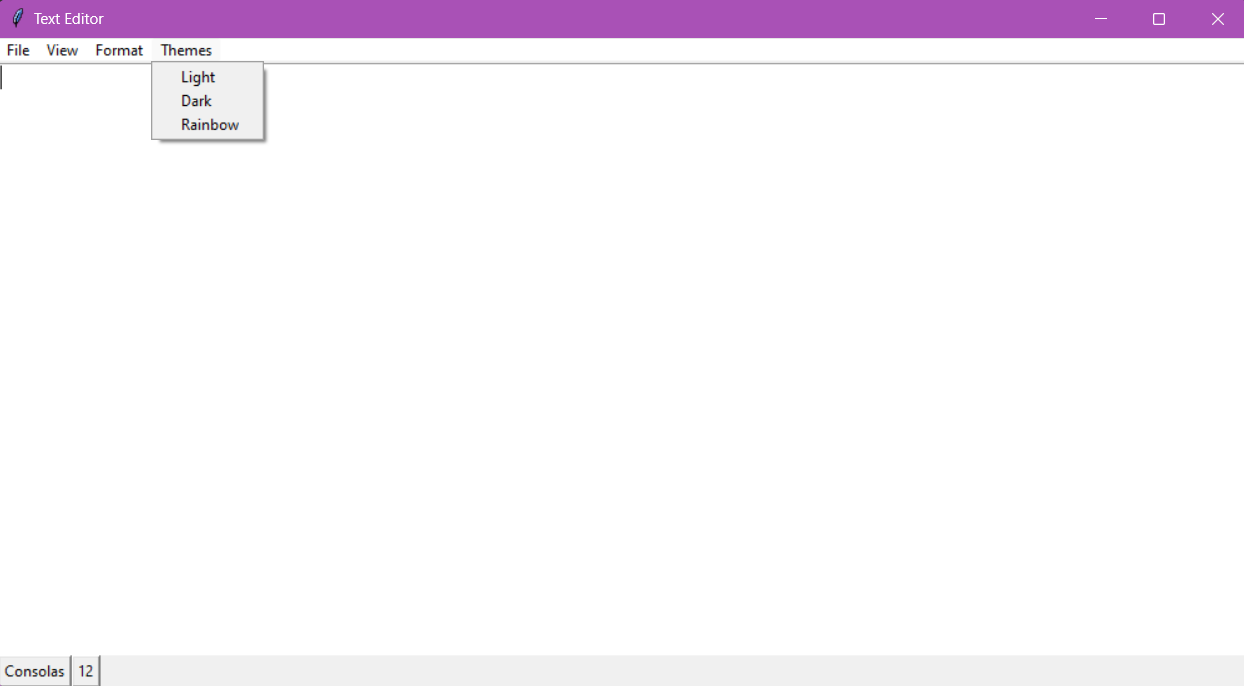
When you click the File button



When you click the View Button

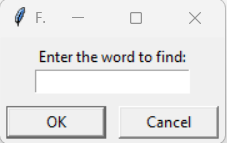


What you see when you click Format



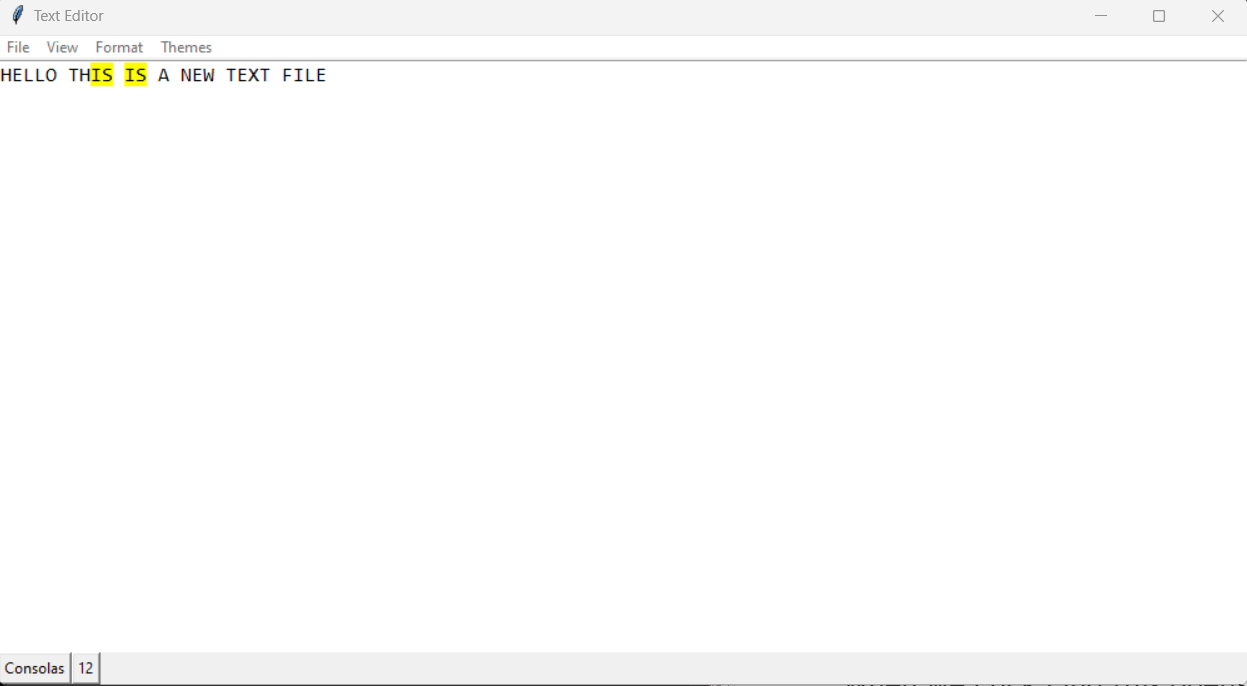
What you see when you click Themes

The New,Open,Save and Save As buttons works the same as Notepad app on Windows.



When we click Find this opens

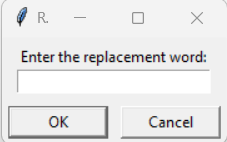
And after entering a word and clicking OK we get



When we click Replace then

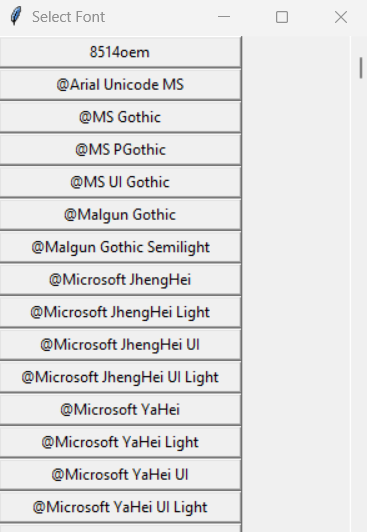
We get the same Find word box

Then after entering that we get this

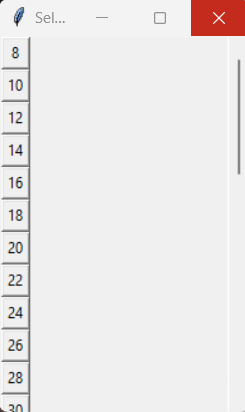


Then after replacing IS with HEY we get

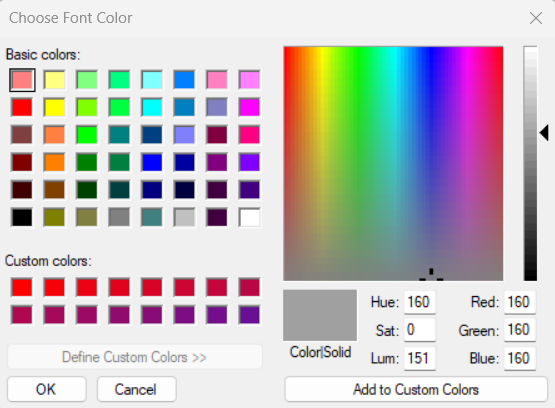




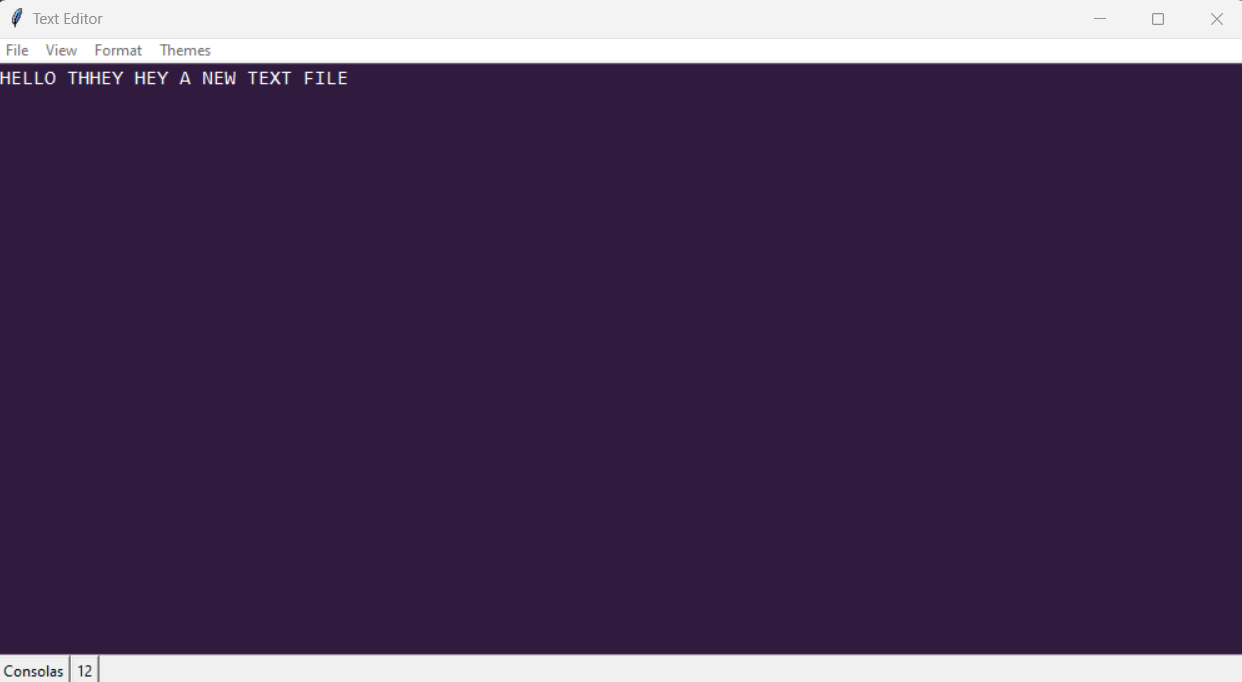
Font Button opens this where we can select which font we want to use



Font size opens this where we can select the font size



Font Color Opens this tab where we can select the exact color we want to use as the font color.

  
This is the Dark Theme for the Editor.

This is the Rainbow Theme which selects a random RGB color for the font color and the background color and applies it.

**Conclusion:**

Thus we have written a program to make a text editor using basic file handling operations in python with the help of tkinter.

Our Text Editor has the basic functionalities such as making a new text file, opening an existing file, saving the file on your pc.

We have also added options to change our font, font color and font size.

We can also search and replace texts of string in the file with this application.

We have also implemented 3 different themes in this which are Light theme, Dark theme and the Rainbow theme which gives a random font and background color.