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"
```

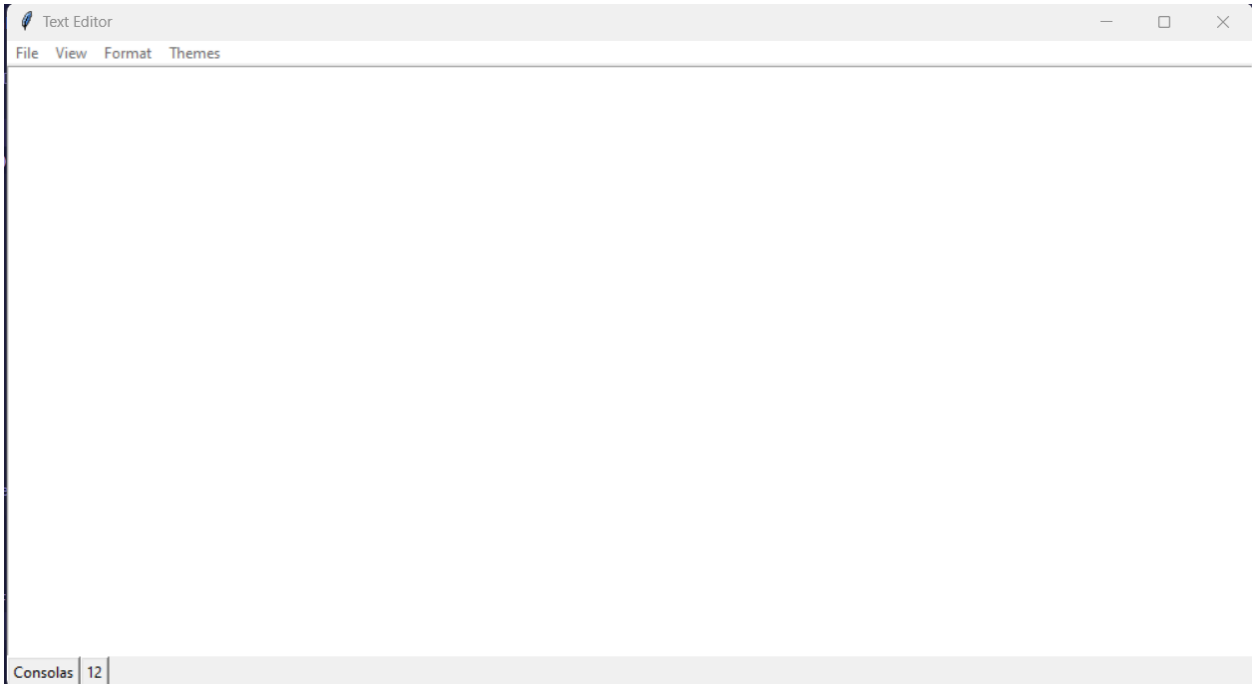
```
            # Highlight the found word
            self.text_area.tag_add('highlight', pos, end)
            start = end

        # 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')
```

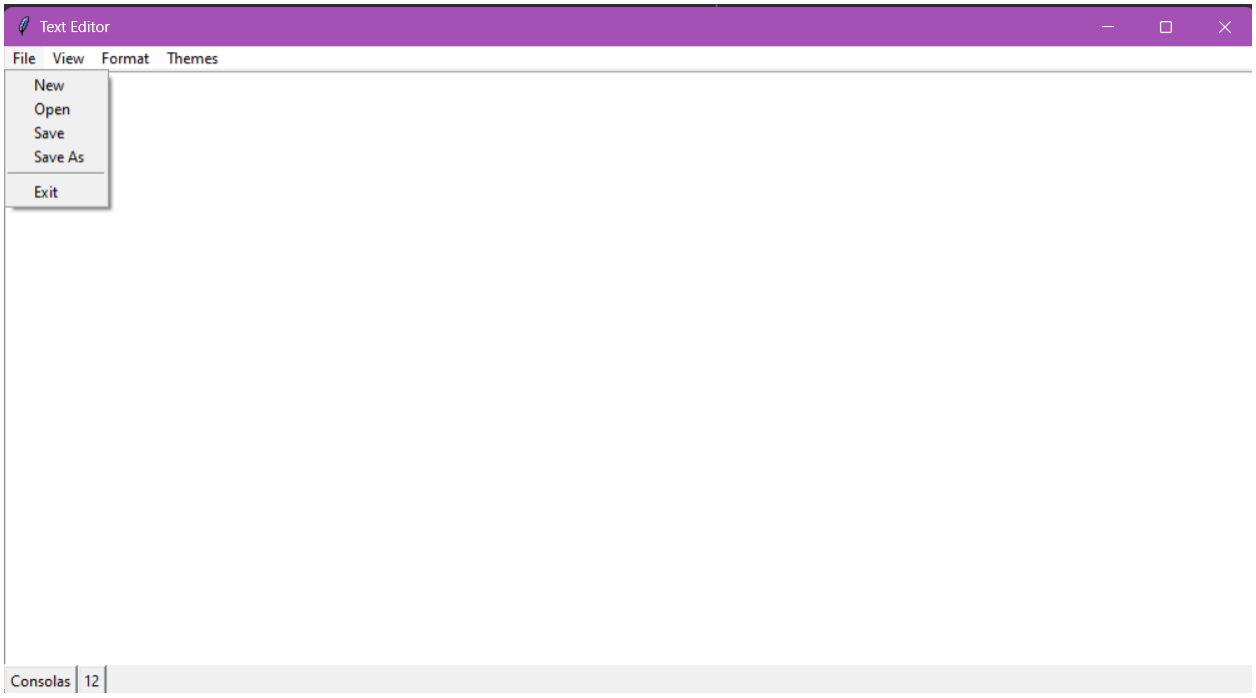
```
        # Pack canvas and scrollbar
        canvas.create_window((0, 0), window=frame, anchor='nw')
        canvas.configure(yscrollcommand=scrollbar.set)
        scrollbar.pack(side='right', fill='y')
        canvas.pack(side='left', fill='both', expand=True)


        # 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()
```

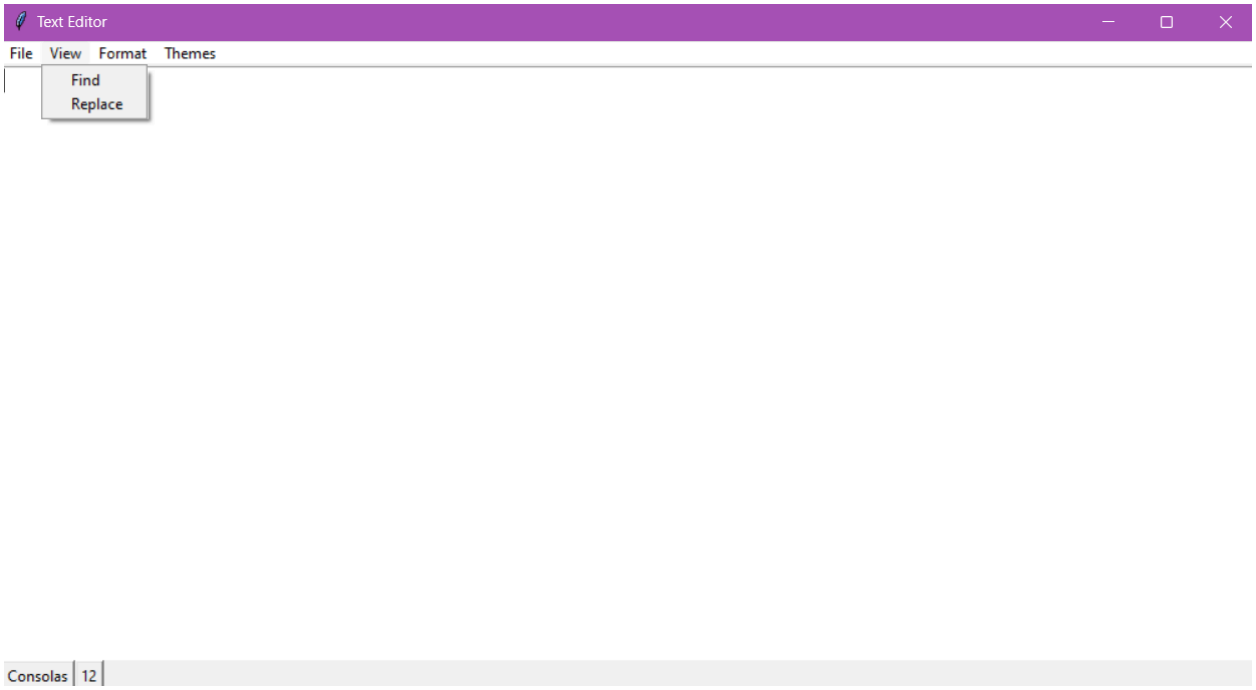
OUTPUT:



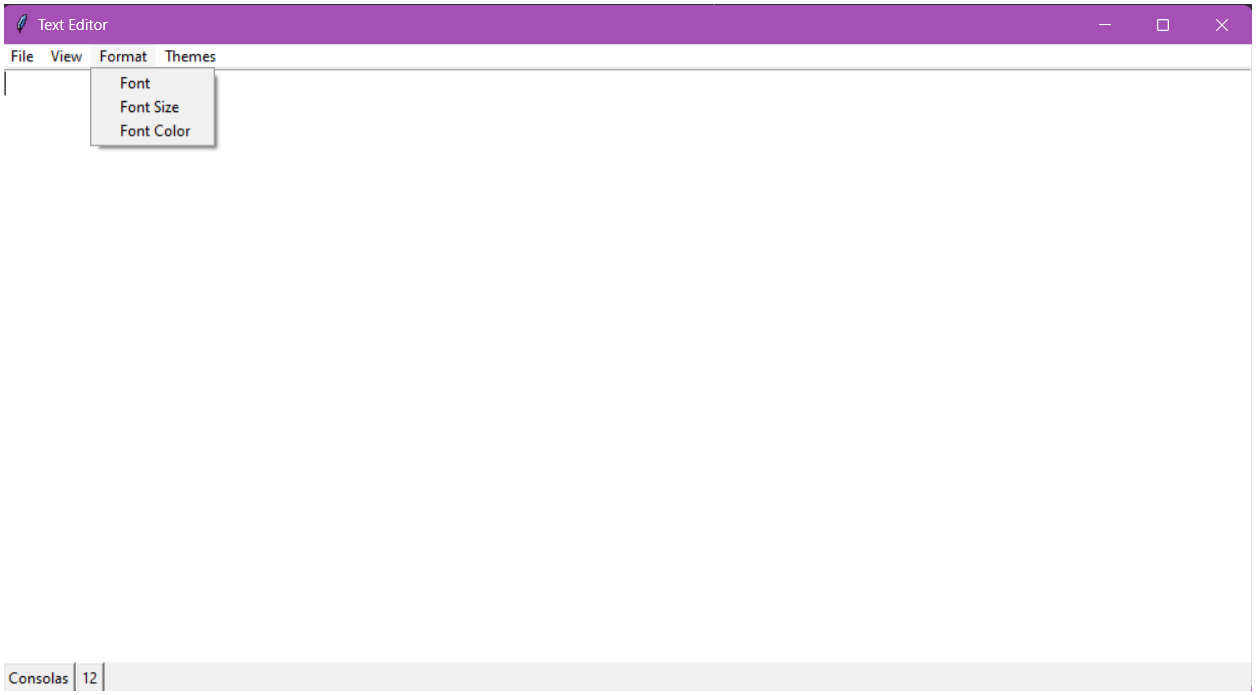
What you see when you run the code.



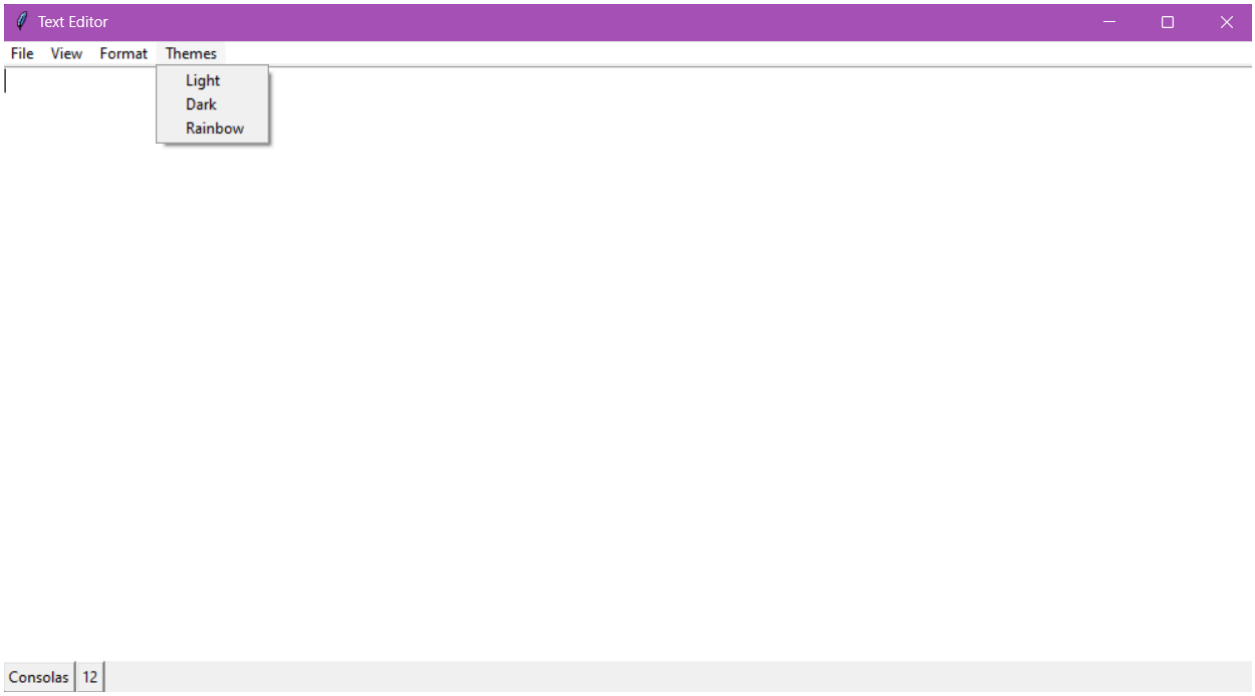
When you click the File button



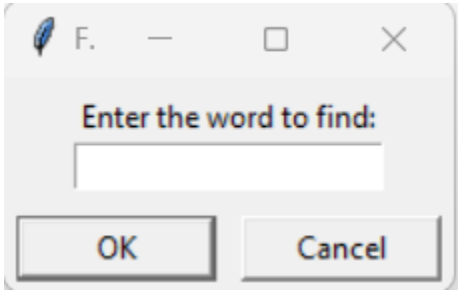
When you click the View Button



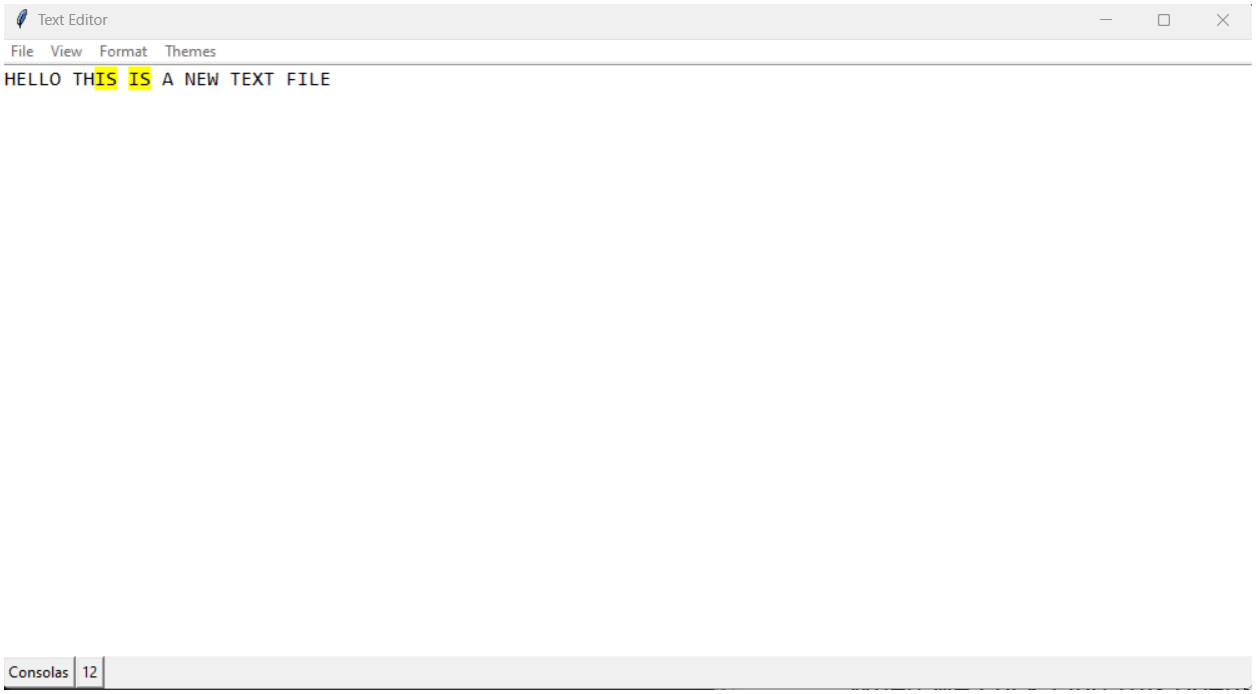
What you see when you click Format



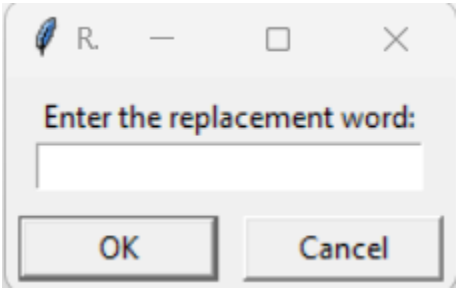
What you see when you click Themes
The New,Open,Save and Save As buttons works the same as Notepad app on Windows.



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

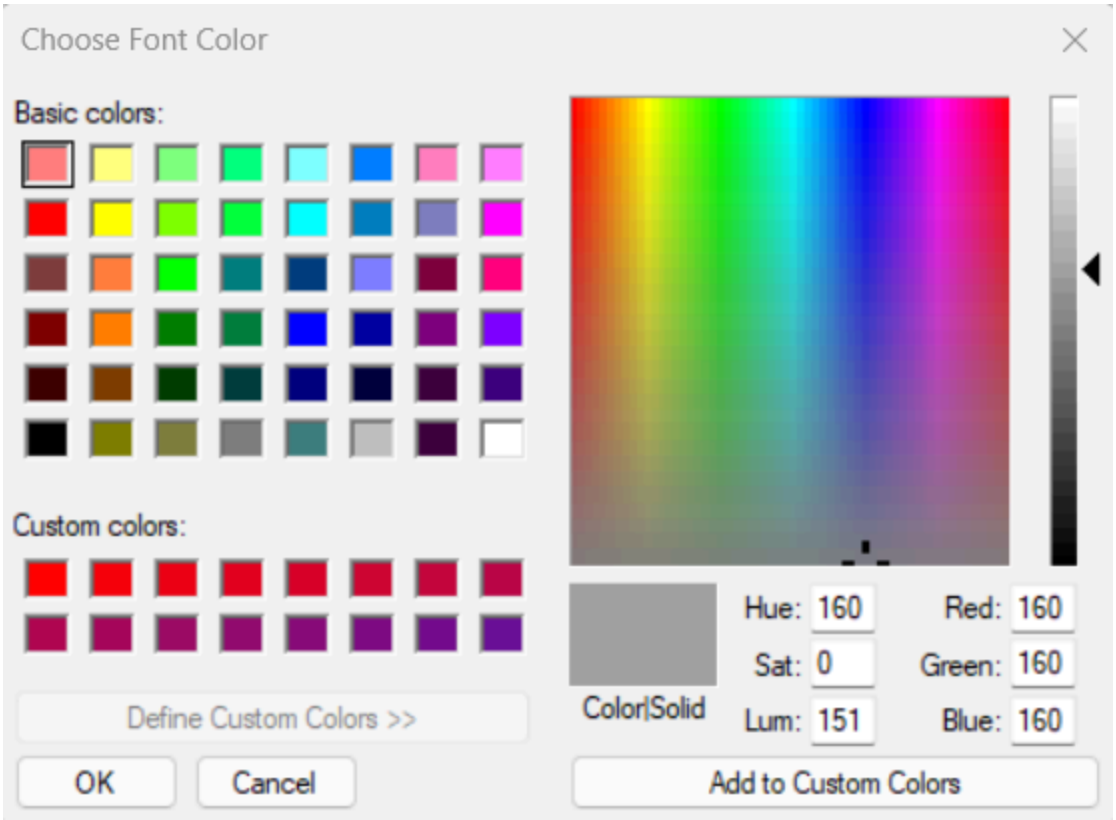


Then after replacing IS with HEY we get

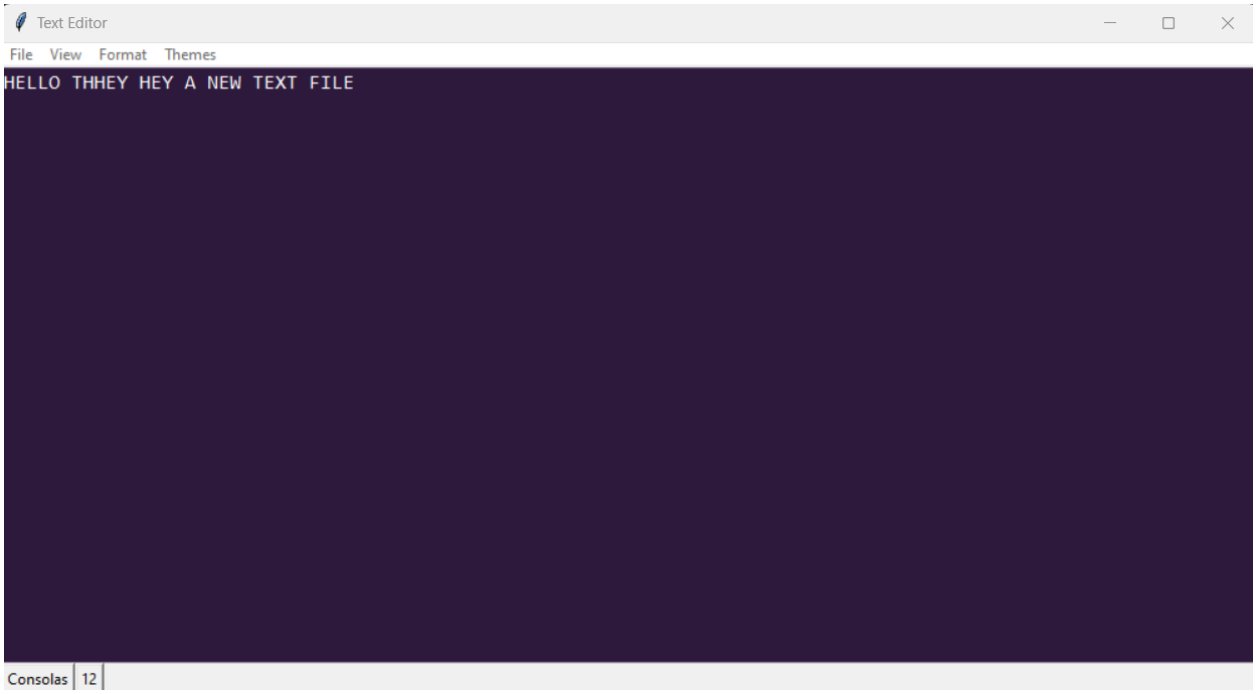

Text Editor
File View Format Themes
HELLO THHEY HEY A NEW TEXT FILE
Consolas 12

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.