Vpad.py (COMPLETE PROGRAM)

```
import tkinter as tk
from tkinter import ttk
from tkinter import font, colorchooser,
    file-dialog, message box

import os
main_application = tk.Tk()
main_application.geometry('1200x100')
main_application.title('Vpad text editor')

---------- main menu ----------

main_menu = tk.Menu()

#File icons

new_icon = tk.PhotoImage(file='icons2/new.png')
open_icon = tk.PhotoImage(file='icons2/open.png')
save_icon = tk.PhotoImage(file='icons2/save.png')
save_as_icon = tk.PhotoImage(file='icons2/save_as.png')
exit_icon = tk.PhotoImage(file='icons2/exit.png')

file = tk.Menu(main_menu, tearoff=False)

#edit icons
copy_icon = tk.PhotoImage(file='icons2/copy.png')

paste_icon = tk.PhotoImage(file='icons2/paste.png')
cut_icon = tk.PhotoImage(file='icons2/cut.png')
clear_all_icon = tk.PhotoImage(file='icons2/clear_
                                all.png')

edit = tk.Menu(main_menu, tearoff=False)
```

LIC
भारतीय जीवन बीमा निगम
LIFE INSURANCE CORPORATION OF INDIA

```
# view icons
tool_bar_icon = tk.PhotoImage(file='icons2/tool_bar.png')
status_bar_icon = tk.PhotoImage(file='icons2/status_bar.png')
view = tk.Menu(main_menu, tearoff=False)

# color theme
light_default_icon = tk.PhotoImage(file='icons2/light_default.png')
light_plus_icon = tk.PhotoImage(file='icons2/light_plus.png')
dark_icon = tk.PhotoImage(file='icons2/dark.png')
red_icon = tk.PhotoImage(file='icons2/red.png')
monokai_icon = tk.PhotoImage(file='icons2/monokai.png')
night_blue_icon = tk.PhotoImage(file='icons2/night_blue.png')

color_theme = tk.Menu(main_menu, tearoff=False)

theme_choice = tk.StringVar()
color_icons = (light_default_icon, light_plus_icon, dark_icon, red_icon, monokai_icon, night_blue_icon)

color_dict = {
    'Light Default': ('#000000', '#ffffff'),
    'Light Plus': ('#474747', '#e0e0e0'),
    'Dark': ('#c4c4c4', '#2d2d2d'),
    'Red': ('#2d2d2d', '#ffe8e8'),
    'Monokai': ('#d3b774', '#474747'),
    'Night Blue': ('#ededed', '#6b9dc2')
}
```

#cascade

```
main_menu.add_cascade(label='File', menu=file)
main_menu.add_cascade(label='Edit', menu=edit)
main_menu.add_cascade(label='View', menu=view)
main_menu.add_cascade(label='Color Theme',
                      menu=color_theme)
```

---------- End main menu ----------

---------- toolbar ----------

```
tool_bar = ttk.Label(main_application)
tool_bar.pack(side=tk.TOP, fill=tk.X)
```

#font box

```
font_tuple = tk.font.families()
font_family = tk.StringVar()
font_box = ttk.Combobox(tool_bar, width=30,
           textvariable=font_family, state='readonly')

font_box['values'] = font_tuple
font_box.current(font_tuple.index('Arial'))
font_box.grid(row=0, column=0, padx=5)
```

#size box

```
size_var = tk.IntVar()

font_size = ttk.Combobox(tool_box, width=14, text-
            variable=size_var, state='readonly')
font_size['values'] = tuple(range(8,81))
#print(tuple(range(8, 80,2)))
font_size.current(4)
font_size.grid(row=0, column=1, padx=5)
```

bold button

```
bold_icon = tk.PhotoImage(file='icons2/bold.png')
bold_btn = ttk.Button(tool_bar, image=bold_icon)
bold_btn.grid(row=0, column=2, padx=5)
```

Italic button

```
italic_icon = tk.PhotoImage(file='icons2/italic.png')
italic_btn = ttk.Button(tool_bar, image='italic_icon)
italic_btn.grid(row=0, column=3, padx=5)
```

underline button

```
underline_icon = tk.PhotoImage(file='icons2/underline.png')
underline_btn = ttk.Button(tool_bar, image=underline_icon
underline_btn.grid(row=0, column=4, padx=5)
```

font color button

```
font_color_icon = tk.PhotoImage(file='icons2/font_color.png')
font_color_btn = ttk.Button(tool_bar, image=font_color_icon)
font_color_btn.grid(row=0, column=5, padx=5)
```

align left

```
align_left_icon = tk.PhotoImage(file='icons2/
                  align_left.png')
align_left_btn = ttk.Button(tool_bar, image=
                 align_left_icon)
align_left_btn.grid(row=0, column=6, padx=5)
```

align center

```
align_center_icon = tk.PhotoImage(file='icons2/
                    align_center.png')
align_center_btn = ttk.Button(tool_bar,
                   image=align_center_icon)
align_center_btn.grid(row=0, column=7, padx=5)
```

align right

```
align_right_icon = tk.PhotoImage(file='icons2/
                   align_right.png')
align_right_btn = ttk.Button(tool_bar, image=
                  align_right_icon)
align_right_btn.grid(row=0, column=8,
                     padx=5)
```

— — — — — End toolbar — — — —

LIC
भारतीय जीवन बीमा निगम
LIFE INSURANCE CORPORATION OF INDIA

text editor :

```
text_editor = tk.Text(main_application)
text_editor.config(wrap='word', relief=tk.FLAT)

scroll_bar = tk.Scrollbar(main_application)
text_editor.focus_set()
scroll_bar.pack(side=tk.RIGHT, fill=tk.Y)

text_editor.pack(fill=tk.BOTH, expand=True)
scroll_bar.config(command=text_editor.yview)
text_editor.config(yscrollcommand=scroll_bar.set)
```

font family and font size functionality

```
current_font_family = 'Arial'
current_font_size = 12

def change_font(event=None):
    global current_font_finally
    current_font_family = font_finally.get()
    text_editor.configure(font=(current_font_family, current_font_size))

def change_fontsize(event=None):
    global current_font_size
    current_font_size = size_var.get()
    text_editor.configure(font=(current_font_family, current_font_size))

font_box.bind("<<ComboboxSelected>>", change_font)
font_size.bind("<<ComboboxSelected>>", change_fontsize)
```

buttons functionality

#bold button functionality

```
def change_bold():
    text_property = tk.font.Font(font=text_editor
                                 ['font'])
    if text_property.actual()['weight'] ==
                                 'normal':
        text_editor.configure(font=(current_font_
            family, current_font_size, 'bold'))

    if text_property.actual()['weight']=='bold':
        text_editor.configure(font=(current_font_
            family, current_font_size, 'normal'))

bold_btn.configure(command=change_bold)
```

italic functionality

```
def change_italic():
    text_property = tk.font.Font(font=text_
        editor['font'])
    if text_property.actual()['slant']=='roman':
        text_editor.configure(font=(current_
            font family, current_font_size,
            'italic'))

    if text_property.actual()['slant']==
        'italic':
        text_editor.configure(font=(current_
            font family, current_font_size, 'normal'))

italic_btn.configure(command=change_italic)
```

underline functionality

```
def change_underline():
    text_property = tk.font.Font(font = text_
        editor['font'])
    if text_property.actual()['underline']==0:
        text_editor.configure(font=(current_
            font_family, current_font_size,'underline'))
    if text_property.actual()['underline']==1:
        text_editor.configure(font=(current_
            font_family, current_font_size,'normal'))

underline_btn.configure(command=change_
                        underline)
```

font color functionality

```
def change_font_color():
    color_var = tk.colorchooser.askcolor()
    text_editor.configure(fg = color_var[1])

font_color_btn.configure(command=change_font_
                         color)
```

align functionality

left

```
def align_left():
    text_content = text_editor.get(1.0, 'end')
    text_editor.tag_config('left', justify=
                           tk.LEFT)
    text_editor.delete(1.0, tk.END)
```

```
        text_editor.insert(tk.INSERT, text_
                           content, 'left')
align_left_btn.configure(command = align_
                                   left)
#center
def align_center():
        text_content = text_editor.get(1.0,
                                   'end')
        text_editor.tag_config('center', justify=
                                   tk.CENTER)
        text_editor.delete(1.0, tk.END)
        text_editor.insert(tk.INSERT, text_
                           content, 'center')
align_center_btn.configure(command = align_
                                   center)
#right
def align_right():
        text_content = text_editor.get(1.0,'end')
        text_editor.tag_config('right', justify=
                tk.RIGHT)
        text_editor.delete(1.0, tk.END)
        text_editor.insert(tk.INSERT, text_
                           content, 'right')

align_right_btn.configure(command= align_right)

text_editor.configure(font = 'Arial', 12))

---- End text editor ----
```

— — — — — — status bar — — — — — —

```
status_bar = ttk.Label(main_application,
                       text='Status Bar')
status_bar.pack(side=tk.BOTTOM)

text_changed=False
def changed(event=None):
    global text_changed
    if text_editor.edit_modified():
        text_changed = True
        words=len(text_editor.get(1.0,
                  'end-1c').
                  split())
        characters=len(text_editor.get(1.0,
                       'end-1c'))
        status_bar.config(text=f'Characters:
        {characters} Words:{words}')

    text_editor.edit_modified(False)
text_editor.bind('<<Modified>>', changed)
```

— — — — End status bar — — — —

— — — — main menu functionality — — —

variable

```
url=''
```

new functionality

```
def new_file(event=None):
    global url
    url=''
    text_editor.delete(1.0, tk.END)

# file commands
file.add_command(label='New', image=new_icon, compound=tk.LEFT, accelerator='Ctrl+N', command=new_file)

# Open functionality
def open_file(event=None):
    global url
    url = filedialog.askopenfilename(initialdir=os.getcwd(), title='Select File', filetypes=(('Text File', '*.txt'), ('All files', '*.*')))

    try:
        with open(url,'r') as fr:
            text_editor.delete(1.0, tk.END)
            text_editor.insert(1.0, fr.read())

    except FileNotFoundError:
        return
    except:
        return

    main_application.title(os.path.basename(url))

file.add_command(label='Open', image=open_icon, compound=tk.LEFT, accelerator='Ctrl+O', command=open_file)
```

भारतीय जीवन बीमा निगम
LIFE INSURANCE CORPORATION OF INDIA

```
# save file functionality

def save_file(event=None):
    global url
    try:
        if url:
            content=str(text_editor.get(1.0,tk.END))
            with open(url,'w',encoding='utf-8')
                                        as fw:
                fw.write(content)

        else:
            url=filedialog.asksaveasfile(mode='w',
            defaultextension='.txt',filetypes=
            ('Text File','*.txt'),('All files',
                                    '*.*'))
            content2=text_editor.get(1.0,tk.END)
            url.write(content2)
            url.close()

    except:
        return

file.add_command(label='save',image=save_icon,
    compound=tk.LEFT, accelerator='ctrl+s',
    command=save_file)

# Save as functionality

def save_as(event=None):
    global url
    try:
        content=text_editor.get(1.0,tk.END)
        url=filedialogasksaveasfile(mode='w'
```

```
            defaultextension='.txt', filetypes=
            [('Text File', '*.txt'), ('All files', '*.*')])
        url.write(content)
        url.close()
    except:
        return

file.add_command(label='Save As', image=
    new_icon, compound=tk.LEFT, accelerator='Ctrl+
    Alt+S', command=save_as)

# exit functionality
def exit_func(event=None):
    global url, text_changed
    try:
        if text_changed:
            mbox = messagebox.askyesnocancel
            ('Warning', 'Do you want to save
            the file?')

            if mbox is True:
                if url:
                    content = text_editor.get(1.0,
                    tk.END)
                    with open(url, 'w', encoding
                    ='utf-8') as fw:
                    fw.write(content)
                    main_application.destroy()

                else:
                    content2 = str(text_editor.get(1.0,
                    tk.END))
                    url = filedialog.asksaveasfile
                    (mode='w', defaultextension=
                    '.txt', filetypes=[('Text File', '*.txt'),
                    ('All files', '*.*')])
```

```
            url.write(content2)
            url.close()
            main_application.destroy()
        elif mbox is False:
            main_application.destroy()

        else:
            main_application.destroy()
    except:
        return

file.add_command(label='Exit', image=exit_icon,
    compound=tk.LEFT, accelerator='Ctrl+Q',
    command=exit_func)

# find functionality
def find_func(event=None):

    def find():
        word=find_input.get()
        text_editor.tag_remove('match', '1.0',
            tk.END)
        matches=0
        if word:

            start_pos='1.0'
            while True:
                start_pos=text_editor.search(word,
                    start_pos, stopindex=tk.END)
                if not start_pos:
                    break
                end_pos=f'{start_pos}+{len(word)}c'
                text_editor.tag_add('match', start_
                    pos, end_pos)
                matches+=1
```

```
        start_pos = end_pos
        text_editor.tag_config('match', foreground='red', background='yellow')

    def replace():
        word = find_input.get()
        replace_text = replace_input.get()
        content = text_editor.get(1.0, tk.END)
        new_content = content.replace(word, replace_text)
        text_editor.delete(1.0, tk.END)
        text_editor.insert(1.0, new_content)

    find_dialogue = tk.Toplevel()
    find_dialogue.geometry('450x250+500+200')
    find_dialogue.title('Find')
    find_dialogue.resizable(0, 0)

    ## frame
    find_frame = ttk.LabelFrame(find_dialogue, text='Find/Replace')
    find_frame.pack(pady=20)

    ## labels
    text_find_label = ttk.Label(find_frame, text='Find:')
    text_replace_label = ttk.Label(find_frame, text='Replace')

    ## entry
    find_input = ttk.Entry(find_frame, width=30)
    replace_button = ttk.Entry(find_frame, width=30)
```

```
## button
find_button = ttk.Button(find_frame, text='Find', command=find)

replace_button = ttk.Button(find_frame, text='Replace', command=replace)

## label grid
text_find_label.grid(row=0, column=0, padx=4, pady=4)

text_replace_label.grid(row=1, column=0, padx=4, pady=4)

## entry grid
find_input.grid(row=0, column=1, padx=4, pady=4)

replace_input.grid(row=1, column=1, padx=4, pady=4)

## button grid
find_button.grid(row=2, column=0, padx=8, pady=4)

replace_button.grid(row=2, column=1, padx=8, pady=4)

find_dialogue.mainloop()
```

```
# edit commands
edit.add_command(label='Copy', image=copy_icon, compound=tk.LEFT, accelerator='Ctrl+C', command=lambda: text_editor.event_generate("<Control c>"))

edit.add_command(label='Paste', image=paste_icon, compound=tk.LEFT, accelerator='Ctrl+V', command=lambda: text_editor.event_generate("<Control V>"))
```

```
edit.add_command(label='Cut', image=cut_icon,
compound=tk.LEFT, accelerator='Ctrl+X', command
=lambda: text_editor.event_generate("<Control x>")
edit.add_command(label='Clear All', image=clear_all
_icon', compound=tk.LEFT, accelerator='Ctrl+Alt+X',
command='lambda: text_editor.delete(1.0, tk.END))
edit.add_command(label='Find', image=find_icon,
compound=tk.LEFT, accelerator='Ctrl+F')

# View check button
show_statusbar=tk.BooleanVar()
show_statusbar.set(True)
show_toolbar=tk.BooleanVar()
show_toolbar.set(True)

def hide_toolbar():
    global show_toolbar
    if show_toolbar:
        tool_bar.pack_forget()
        show_toolbar=False
    else:
        text_editor.pack_forget()
        status_bar.pack_forget()
        tool_bar.pack(side=tk.TOP, fill=tk.X)
        text_editor.pack(fill=tk.BOTH, expand=True)
        status_bar.pack(side=tk.BOTTOM)

    show_toolbar=True
def hide_statusbar():
    global show_statusbar
    if show_statusbar:
        status_bar.pack_forget()
        show_statusbar=False
```

```
    else:
        status_bar.pack(side=tk.BOTTOM)
        show_statusbar = True
view.add_checkbutton(label='Tool Bar', onvalue=True,
  offvalue=0, variable=show_toolbar, image=tool=
  bar_icon, compound=tk.LEFT, command=hide_
  toolbar)
view.add_checkbutton(label='Status Bar', onvalue=1,
  offvalue=False, variable=show_statusbar, image=
  status_bar_icon, compound=tk.LEFT, command=
  hide_statusbar)

# color theme
def change_theme():
    chosen_theme = theme_choice.get()
    color_tuple = theme_choice.get()
    fg_color, bg_color = color_tuple[0], color_tuple[1]
    text_editor.config(background=bg_color, fg=
    fg.color)

count=0
for i in color_dict:
    color_theme.add_radiobutton(label=i, image=
      color_icons[count], variable=theme_choice,
      compound=tk.LEFT, command=change_theme)
---------- End main menu functionality ----
main_application.config(menu=main_menu)
# bind shortcut keys
```

```
main_application.bind("<Control-n>", new_file)
main_application.bind("<Control-O>", open_file)
main_application.bind("<Control_S>", save_file)
main_application.bind("<Control_Alt_S>", save_as)
main_application.bind("<Control_q>", exit_func)
main_application.bind("<Control-f>", find_func)
main_application.mainloop()
```