# how to create the notepad in python

import tkinter as tk

from tkinter import ttk

from tkinter import font,colorchooser,filedialog,messagebox

import os

main\_application = tk.Tk()

main\_application.geometry("800x600")

main\_application.title("Ms notepad")

main\_application.wm\_iconbitmap("icon.ico")

main\_menu = tk.Menu()

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 menu icon

copy\_icon = tk.PhotoImage(file = "icons2/copy.png")

paste\_icon = tk.PhotoImage(file="icons2/paste.png")

cut\_item\_icon = tk.PhotoImage(file="icons2/cut.png")

clear\_icon = tk.PhotoImage(file="icons2/clear\_all.png")

find\_icon = tk.PhotoImage(file="icons2/find.png")

edit = tk.Menu(main\_menu,tearoff = False)

tool\_bar = tk.PhotoImage(file = "icons2/tool\_bar.png")

status\_bar = tk.PhotoImage(file = "icons2/status\_bar.png")

view = tk.Menu(main\_menu,tearoff = False)

# color image

light\_theme = tk.PhotoImage(file = "icons2/light\_default.png")

light\_plus\_icon= tk.PhotoImage(file = "icons2/light\_plus.png")

dark\_theme = tk.PhotoImage(file = "icons2/dark.png")

red\_theme = tk.PhotoImage(file = "icons2/red.png")

monokia\_theme = tk.PhotoImage(file = "icons2/monokai.png")

night\_theme = tk.PhotoImage(file = "icons2/night\_blue.png")

color\_theme = tk.Menu(main\_menu,tearoff = False)

theme\_choose = tk.StringVar()

color\_icons=(light\_plus\_icon,light\_theme,dark\_theme,red\_theme,monokia\_theme,night\_theme)

color\_dict = {

'Light Default ' : ('#000000',"#ffffff"),

'Light Plus' : ('#474747', '#e0e0e0'),

'Dark' : ('#c4c4c4', '#2d2d2d'),

'Red' : ('#2d2d2d', '#ffe8e8'),

'Monokai' : ('#d3b774', '#474747'),

'Night Blue' :('#ededed', '#6b9dc2')

}

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)

tool\_bar\_label = ttk.Label(main\_application)

tool\_bar\_label.pack(side = tk.TOP,fill = tk.X)

font\_tuple = tk.font.families()

font\_family = tk.StringVar()

font\_box = ttk.Combobox(tool\_bar\_label,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,pady = 5)

# size box

size\_variable = tk.IntVar()

font\_size = ttk.Combobox(tool\_bar\_label,width = 20,textvariable = size\_variable,state = "readonly")

font\_size["values"] = tuple(range(8,100,2))

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\_label,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\_label, 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\_label, 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\_label, 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\_label, 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\_label, 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\_label, image=align\_right\_icon)

align\_right\_btn.grid(row=0, column=8, padx=5)

# 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 function

font\_now = "Arial"

font\_size\_now = 16

def change\_font(main\_application):

global font\_now

font\_now = font\_family.get()

text\_editor.configure(font = (font\_now,font\_size\_now))

def change\_size(main\_application):

global font\_size\_now

font\_size\_now = size\_variable.get()

text\_editor.configure(font = (font\_now,font\_size\_now))

font\_box.bind("<<ComboboxSelected>>",change\_font)

font\_size.bind("<<ComboboxSelected>>",change\_size)

# bold function

# print(tk.font.Font(font=text\_editor["font"]).actual())

def bold\_fun():

text\_get = tk.font.Font(font=text\_editor["font"])

if text\_get.actual()["weight"] == 'normal':

text\_editor.configure(font =(font\_now,font\_size\_now,"bold"))

if text\_get.actual()["weight"] == 'bold':

text\_editor.configure(font =(font\_now,font\_size\_now,"normal"))

bold\_btn.configure(command = bold\_fun)

def Italic\_fun():

text\_get = tk.font.Font(font=text\_editor["font"])

if text\_get.actual()["slant"] == 'roman':

text\_editor.configure(font =(font\_now,font\_size\_now,"italic"))

if text\_get.actual()["slant"] == 'italic':

text\_editor.configure(font =(font\_now,font\_size\_now,"roman"))

italic\_btn.configure(command = Italic\_fun)

def under\_line\_fun():

text\_get = tk.font.Font(font=text\_editor["font"])

if text\_get.actual()["underline"] == 0:

text\_editor.configure(font =(font\_now,font\_size\_now,"underline"))

if text\_get.actual()["underline"] == 1:

text\_editor.configure(font =(font\_now,font\_size\_now,"normal"))

underline\_btn.configure(command = under\_line\_fun)

def Color\_choose():

color\_var = tk.colorchooser.askcolor()

text\_editor.configure(fg=color\_var[1])

font\_color\_btn.configure(command = Color\_choose)

def align\_left():

text\_get\_all = 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\_get\_all,"left")

align\_left\_btn.configure(command = align\_left)

def align\_center():

text\_get\_all = 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\_get\_all,"center")

align\_center\_btn.configure(command = align\_center)

def align\_right():

text\_get\_all = 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\_get\_all,"right")

align\_right\_btn.configure(command = align\_right)

# status bar word and character count

status\_bars = ttk.Label(main\_application,text = "Status bar")

status\_bars.pack(side = tk.BOTTOM)

text\_change = False

def change\_word(event = None):

global text\_change

if text\_editor.edit\_modified():

text\_change = True

word = len(text\_editor.get(1.0,"end-1c").split())

chararcter = len(text\_editor.get(1.0,"end-1c").replace(" ",""))

status\_bars.config(text = f"character :{chararcter} word :{word}")

text\_editor.edit\_modified(False)

text\_editor.bind("<<Modified>>",change\_word)

# color theme function

# light\_theme = tk.PhotoImage(file = "icons2/light\_default.png")

# light\_plus\_icon= tk.PhotoImage(file = "icons2/light\_plus.png")

# dark\_theme = tk.PhotoImage(file = "icons2/dark.png")

# red\_theme = tk.PhotoImage(file = "icons2/red.png")

# monokia\_theme = tk.PhotoImage(file = "icons2/monokai.png")

# night\_theme = tk.PhotoImage(file = "icons2/night\_blue.png")

# color\_theme = tk.Menu(main\_menu,tearoff = False)

# theme\_choose = tk.stringVar()

# color\_icons=(light\_plus\_icon,light\_theme,dark\_theme,red\_theme,monokia\_theme,night\_theme)

# color\_dict = {

# 'Light Default ' : ('#000000',"#ffffff"),

# 'Light Plus' : ('#474747', '#e0e0e0'),

# 'Dark' : ('#c4c4c4', '#2d2d2d'),

# 'Red' : ('#2d2d2d', '#ffe8e8'),

# 'Monokai' : ('#d3b774', '#474747'),

# 'Night Blue' :('#ededed', '#6b9dc2')

# }

def change\_theme():

get\_theme = theme\_choose.get()

colour\_tuple = color\_dict.get(get\_theme)

fg\_color,bg\_color = colour\_tuple[0],colour\_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\_choose,compound = tk.LEFT,command = change\_theme)

count += 1

# view menu

# tool bar and status bar hide

# tool\_bar\_label = ttk.Label(main\_application)

# tool\_bar\_label.pack(side = tk.TOP,fill = tk.X)

# status\_bars = ttk.Label(main\_application,text = "Status bar")

# status\_bars.pack(side = tk.BOTTOM)

show\_status\_bar = tk.BooleanVar()

show\_status\_bar.set(True)

show\_toolbar = tk.BooleanVar()

show\_toolbar.set(True)

def hide\_toolbar():

global show\_toolbar

if show\_toolbar:

tool\_bar\_label.pack\_forget()

show\_toolbar = False

else:

text\_editor.pack\_forget()

status\_bars.pack\_forget()

tool\_bar\_label.pack(side = tk.TOP,fill = tk.X)

text\_editor.pack(fill=tk.BOTH, expand=True)

status\_bars.pack(side=tk.BOTTOM)

show\_toolbar = True

def hide\_statusbar():

global show\_status\_bar

if show\_status\_bar:

status\_bars.pack\_forget()

show\_status\_bar = False

else :

status\_bars.pack(side=tk.BOTTOM)

show\_status\_bar = True

view.add\_checkbutton(label = "Tool Bar",onvalue = True,offvalue = 0,variable =show\_toolbar ,image = tool\_bar,compound = tk.LEFT,command =hide\_toolbar)

view.add\_checkbutton(label = "Status Bar",onvalue = True,offvalue = 0,variable = show\_status\_bar,image = status\_bar,compound = tk.LEFT,command =hide\_statusbar)

# edit menu

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\_item\_icon,compound=tk.LEFT,accelerator = "Ctrl+x", command=lambda:text\_editor.event\_generate("<Control x>"))

edit.add\_command(label="Clear all",image=clear\_icon,compound=tk.LEFT,accelerator = "Ctrl+Alt+x",command= lambda:text\_editor.delete(1.0, tk.END))

def find\_fun(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 = "blue")

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\_popup = tk.Toplevel()

find\_popup.geometry("450x200")

find\_popup.title("find word")

find\_popup.resizable(0,0)

# fram for find

find\_fram = ttk.LabelFrame(find\_popup,text = "Find and Replac word")

find\_fram.pack(pady = 20)

# label

text\_find = ttk.Label(find\_fram,text = "Find")

text\_replace = ttk.Label(find\_fram,text = "Replace")

# entry box

find\_input = ttk.Entry(find\_fram,width = 30 )

replace\_input = ttk.Entry(find\_fram,width = 30 )

# button

find\_button = ttk.Button(find\_fram,text = "find",command = find)

replace\_button = ttk.Button(find\_fram,text = "Replace",command = replace)

# text label grid

text\_find.grid(row = 0,column = 0,padx = 4,pady = 4)

text\_replace.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)

edit.add\_command(label="Find",image=find\_icon,compound=tk.LEFT,accelerator = "Ctrl+f",command= find\_fun)

# file menu

text\_url = " "

def new\_file(event = None):

global text\_url

text\_url = " "

text\_editor.delete(1.0,tk.END)

file.add\_command(label="New",image = new\_icon,compound = tk.LEFT,accelerator = "Ctrl+N",command = new\_file)

def open\_file(event = None):

global text\_url

text\_url = filedialog.askopenfilename(initialdir = os.getcwd(),title= "select file",filetypes = (("Text file","\*.txt"),("All files","\*.\*")))

try:

with open(text\_url,"r") as for\_read:

text\_editor.delete(1.0,tk.END)

text\_editor.insert(1.0,for\_read.read())

except FileNotFoundError:

return

except:

return

main\_application.title(os.path.basename(text\_url))

file.add\_command(label="Open",image=open\_icon,compound=tk.LEFT,accelerator = "Ctrl+o",command = open\_file)

def save\_file(event = None):

global text\_url

try:

if text\_url:

content = str(text\_editor.get(1.0,tk.END))

with open(text\_url,"w",encoding="utf-8") as for\_read:

for\_read.write(content)

else:

text\_url = filedialog.asksaveasfile(mode = "w",defaultextension = ".txt",filetypes = (("Text file","\*.txt"),("All files","\*.\*")))

content2 = text\_editor.get(1.0,tk.END)

text\_url.write(content2)

text\_url.close()

except:

return

file.add\_command(label="Save",image=save\_icon,compound=tk.LEFT,accelerator = "Ctrl+s",command = save\_file)

def Save\_as(event = None):

global text\_url

try:

content = text\_editor.get(1.0,tk.END)

text\_url = filedialog.asksaveasfile(mode = "w",defaultextension = ".txt",filetypes = (("Text file","\*.txt"),("All files","\*.\*")))

text\_url.write(content)

text\_url.close()

except:

return

file.add\_command(label="Save as",image=save\_as\_icon,compound=tk.LEFT,accelerator = "Ctrl+Alt+s",command = Save\_as)

def Exit\_fun(event = None):

global text\_url,text\_change

try:

if text\_change:

mbox = messagebox.askyesnocancel("warning","Do you want to save this file")

if mbox is True:

if text\_url:

content = text\_editor.get(1.0,tk.END)

with open(text\_url,"w",encoding = "utf-8") as for\_read:

for\_read.write(content)

main\_application.destroy()

else:

content2 = str(text\_editor.get(1.0,tk.END))

text\_url = filedialog.asksaveasfile(mode = "w",defaultextension = ".txt",filetypes = (("Text file","\*.txt"),("All files","\*.\*")))

text\_url.write(content2)

text\_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+",command = Exit\_fun)

main\_application.config(menu = main\_menu)

main\_application.mainloop()