|  |  |
| --- | --- |
| Airforce school jamnagar  ACADEMIC YEAR : 2020-21 | **PROJECT ON NOTEPAD USING TKINTER MODULE IN PYTHON**  Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.  NAME : KIRAN KHER  CLASS : XII-SCIENCE  ROLL NO:  SUBJECT: COMPUTER SCIENCE  PROJECT GUIDE: MRS.AMITA PANDEY |

**Notepad using tkinter through Python**



Certificate

This is to certify that KIRAN KHER student of class XII-SCIENCE has successfully completed the project work entitled NOTEPAD USING TKINTER in the subject of COMPUTER SCIENCE (083) laid down in the regulations of CBSE for the purpose of AISSE(Practical Examination 2020) in Class XII to be held in Airforce School Jamnagar.

MRS.AMITA PANDEY

PGT Comp Sci.

EXAMINER:

NAME:

SIGNATURE:

Acknowledgement

Apart from the efforts of me, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project.

I express deep sense of gratitude to almighty God for giving me strength for the successful completion of the project.

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this project.

I gratefully acknowledge the contribution of the individuals who contributed in bringing this project up to this level, who continues to look after me despite my flaws,

I express my deep sense of gratitude to the luminary **Mrs.Shubhra Gupta**, Principal Airforce School Jamnagar who has been continuously motivating and extending their helping hand to us.

I am overwhelmed to express my thanks to The Administrative Officer for providing me an infrastructure and moral support while carrying out this project in the school.

My sincere thanks to **Mrs. Amita Pandey.** A guide, Mentor all the above a friend, who critically reviewed my project and helped in solving each and every problem, occurred during implementation of the project

The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their constant support and help.

Table Of Contents

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Description** | **Page no.** |
| 1. | INTRODUCTION | 5 |
| 2. | ABOUT TKINTER | 6 |
| 3. | RENDERING A BASIC GUI | 7 |
| 4. | SOURCE CODE | 8 |
| 5. | OUTPUT | 22 |
| 6. | TESTING | 26 |
| 7. | SOFTWARE REQUIREMENTS | 28 |
| 8. | CONCLUSION | 29 |
| 9. | BIBLIOGRAPHY | 30 |

Introduction

What is Python?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

What can Python do?

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

About Tkinter

Python has a lot of [GUI frameworks](http://wiki.python.org/moin/GuiProgramming), but [Tkinter](https://wiki.python.org/moin/TkInter) is the only framework that’s built into the Python standard library. Tkinter has several strengths. It’s **cross-platform**, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they’re run.

Although Tkinter is considered the de-facto Python GUI framework, it’s not without criticism. One notable criticism is that GUIs built with Tkinter look outdated. If you want a shiny, modern interface, then Tkinter may not be what you’re looking for.

However, Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that’s functional and cross-platform quickly.

Rendering a basic GUI

Source Code

Notepad.py

import tkinter as tk

from tkinter import ttk

from tkinter import font , colorchooser, filedialog, messagebox

import os

main\_application = tk.Tk()

main\_application.geometry('1200x800')

main\_application.title('Notepad')

main\_application.wm\_iconbitmap('mainicon.ico')

########## 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

#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')

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

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

##commands are added after edit menu

####view

#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)

#all icons saved in a tuple

theme\_choice = tk.StringVar()

color\_icons = (light\_default\_icon ,light\_plus\_icon,dark\_icon,red\_icon,monokai\_icon,night\_blue\_icon)

## text ,background

#

color\_dict = {

'Light Default' :('#000000','fffffff'),

'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\_bar,width=14,textvariable = size\_var,state='readonly')

font\_size['values']=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\_icon = tk.PhotoImage(file='icons2/font\_color.png')

font\_color\_btn = ttk.Button(tool\_bar,image=font\_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 &&&&&----------#

########## 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\_family

current\_font\_family = font\_family.get()

text\_editor.config(font=(current\_font\_family,current\_font\_size))

def change\_size(event=None):

global current\_font\_size

current\_font\_size = size\_var.get()

text\_editor.config(font=(current\_font\_family,current\_font\_size))

##binding combobox with function

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

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

####### buttons functionality

#bold buttton functionality

def change\_bold():

text\_property=tk.font.Font(font=text\_editor['font'])

##upper line gives a dictionary whose attributes we are changing

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 button functionality

def change\_italic():

text\_property=tk.font.Font(font=text\_editor['font'])

##upper line gives a dictionary whose attributes we are changing

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 button functionality

def underline():

text\_property=tk.font.Font(font=text\_editor['font'])

##upper line gives a dictionary whose attributes we are changing

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=underline)

##font color functionality

def change\_font\_color():

color\_var = tk.colorchooser.askcolor()

##ask color asks for a color and stores into the color\_var

##text color is called foreground color also abbrivated as fg

## a tuple in which 0th ondex shows the RGB values where as 1st index shows hexa value for color

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

font\_color\_btn.configure(command=change\_font\_color)

### align functionality

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)

###align 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)

##align 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():###checks if any character is added or not

text\_changed= True

words = len(text\_editor.get(1.0, 'end-1c').split()) ##it even counts new line character so end-1c subtracts one char

characters = len(text\_editor.get(1.0,'end-1c'))

status\_bar.config(text=f' Words: {words} Characters : {characters}')

text\_editor.edit\_modified(False)

text\_editor.bind('<<Modified>>',changed)

#----------&&&&& End main status bar &&&&&----------#

########## main menu functinality #############

##file commands

##variable

url = ''

##new functionality

def new\_file(event=None):

global url

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 )

##open functionality

## it is coppying the data from the desired file into the working file

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 )

##save 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','\*.\*')))

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

url.write(content)

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 = filedialog.asksaveasfile(mode = 'w' ,defaultextension = '.txt',filetypes=(('Text File','\*.txt'),('All files','\*.\*')))

url.write(content)

url.close

except :

return

file.add\_command(label='Save As', image=save\_as\_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 user wants to save the file and it already exists

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 )

###edit commands

### find functionality

def find\_func(event=None):

##using tag inbuilt function

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='')

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.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 boxes

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

replace\_input = 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.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',command=find\_func)

#view check button

##it will have 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():

choose\_theme = theme\_choice.get()

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

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)

count+=1

#----------&&&&& End main menu functinality &&&&&----------#

##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-q>", exit\_func)

main\_application.bind("<Control-f>", find\_func)

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

main\_application.mainloop()

setup.py

import cx\_Freeze

import sys

import os

base = None

if sys.platform == 'win32':

base = "Win32GUI"

os.environ['TCL\_LIBRARY'] = r"C:\Users\MAUT\AppData\Local\Programs\Python\Python37\tcl\tcl8.6"

os.environ['TK\_LIBRARY'] = r"C:\Users\MAUT\AppData\Local\Programs\Python\Python37\tcl\tk8.6"

executables = [cx\_Freeze.Executable("Notepad.py", base=base, icon="mainicon.ico")]

cx\_Freeze.setup(

name = " Notepad ",

options = {"build\_exe": {"packages":["tkinter","os"], "include\_files":["mainicon.ico",'tcl86t.dll','tk86t.dll', 'icons2']}},

version = "0.01",

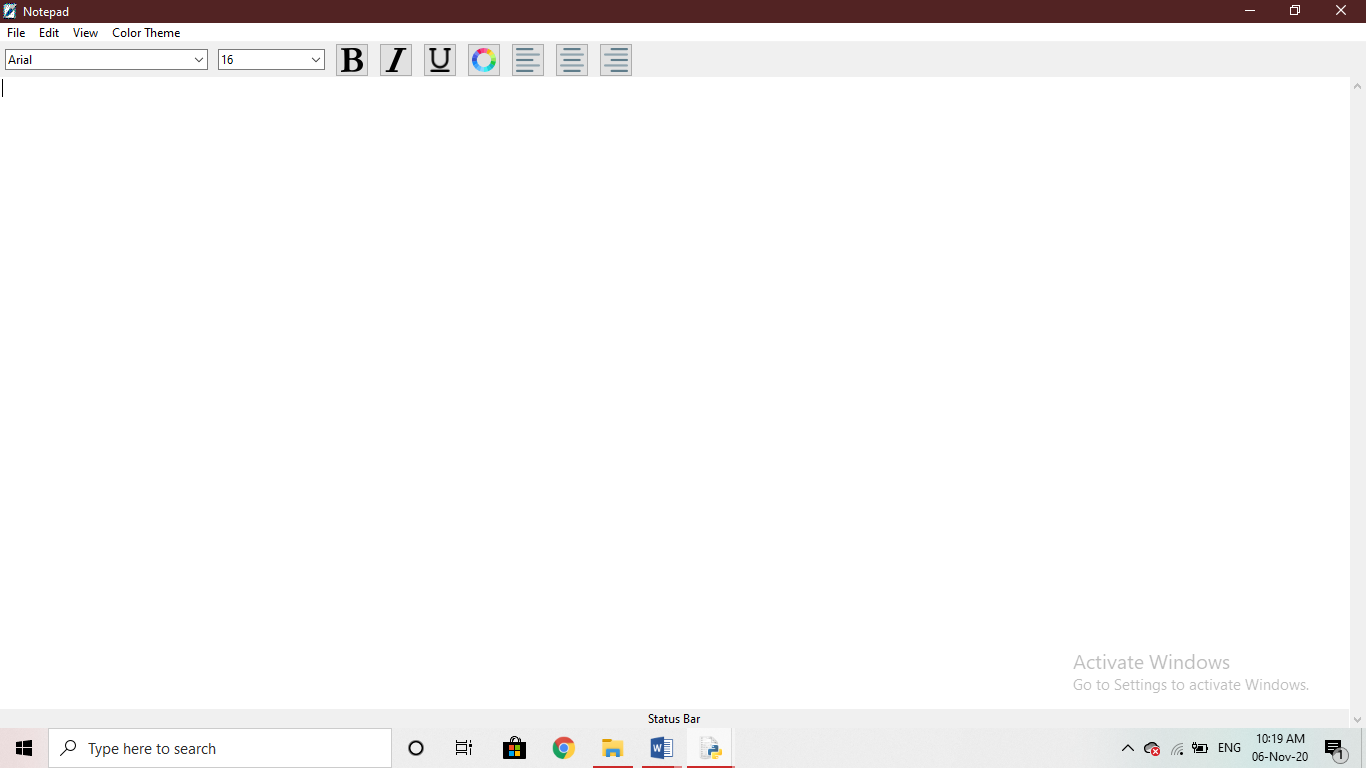
description = "Tkinter Application",

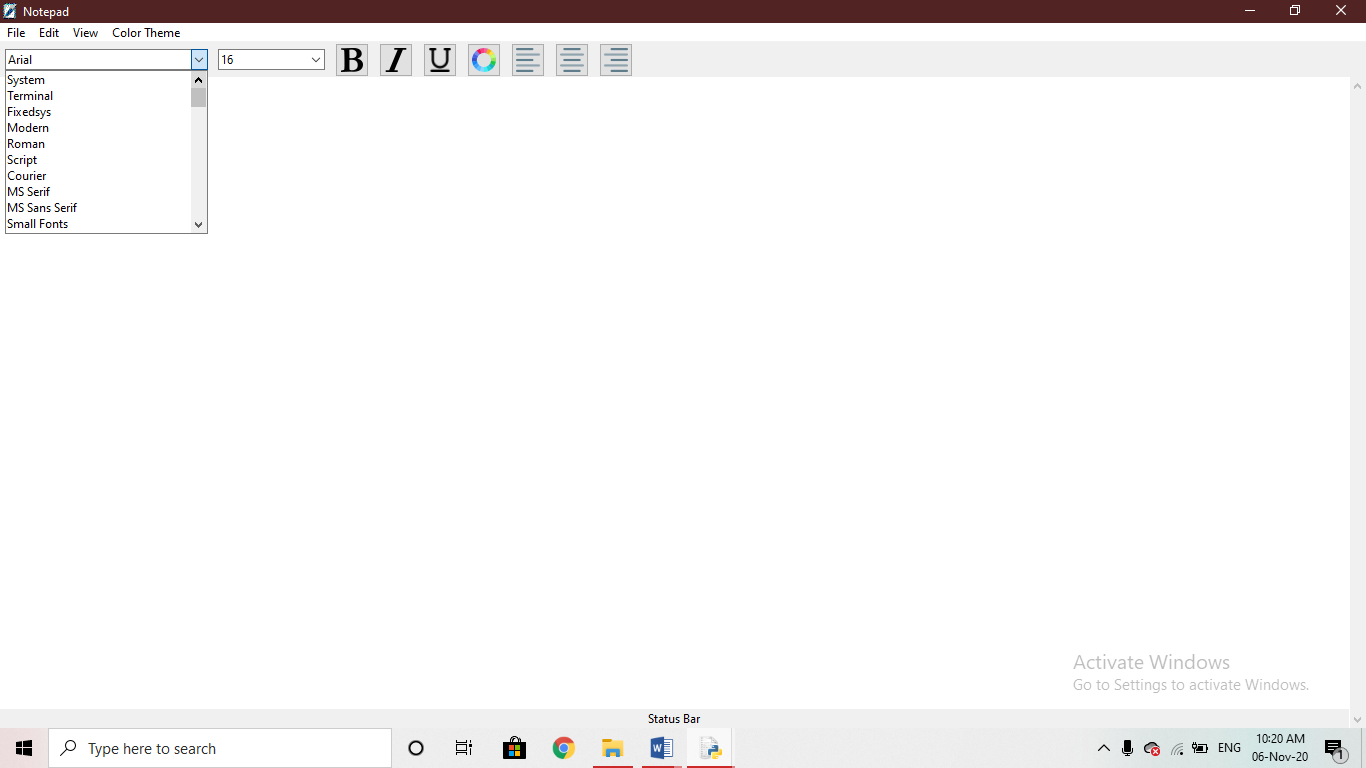
executables = executables

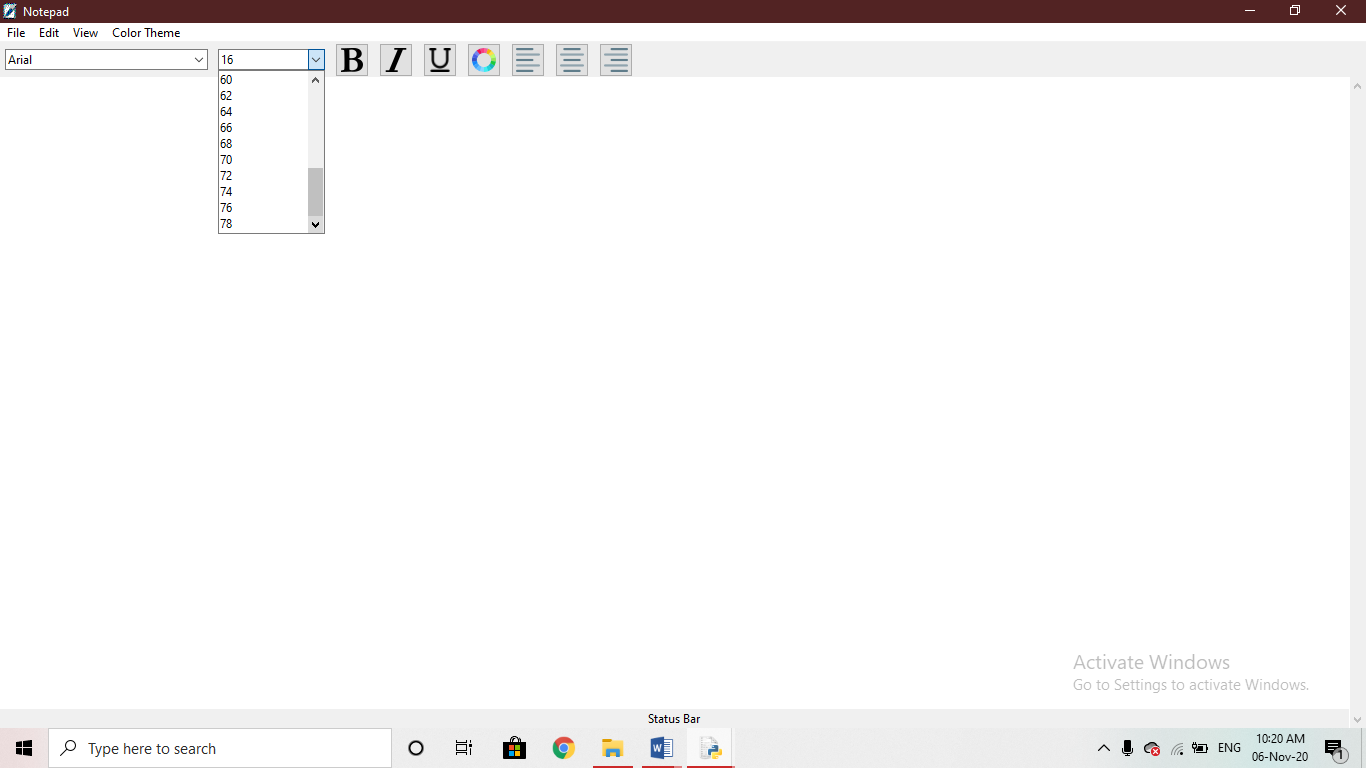
)

Output

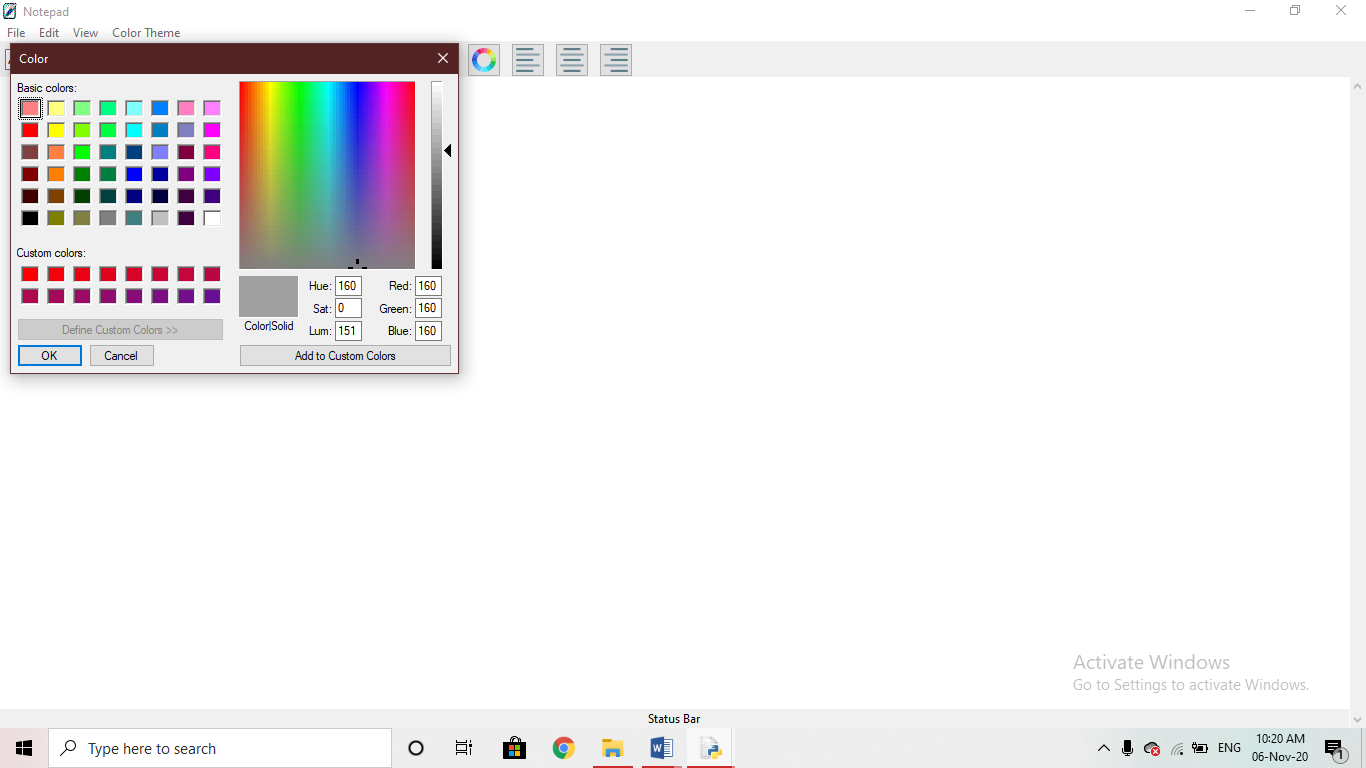
* Notepad



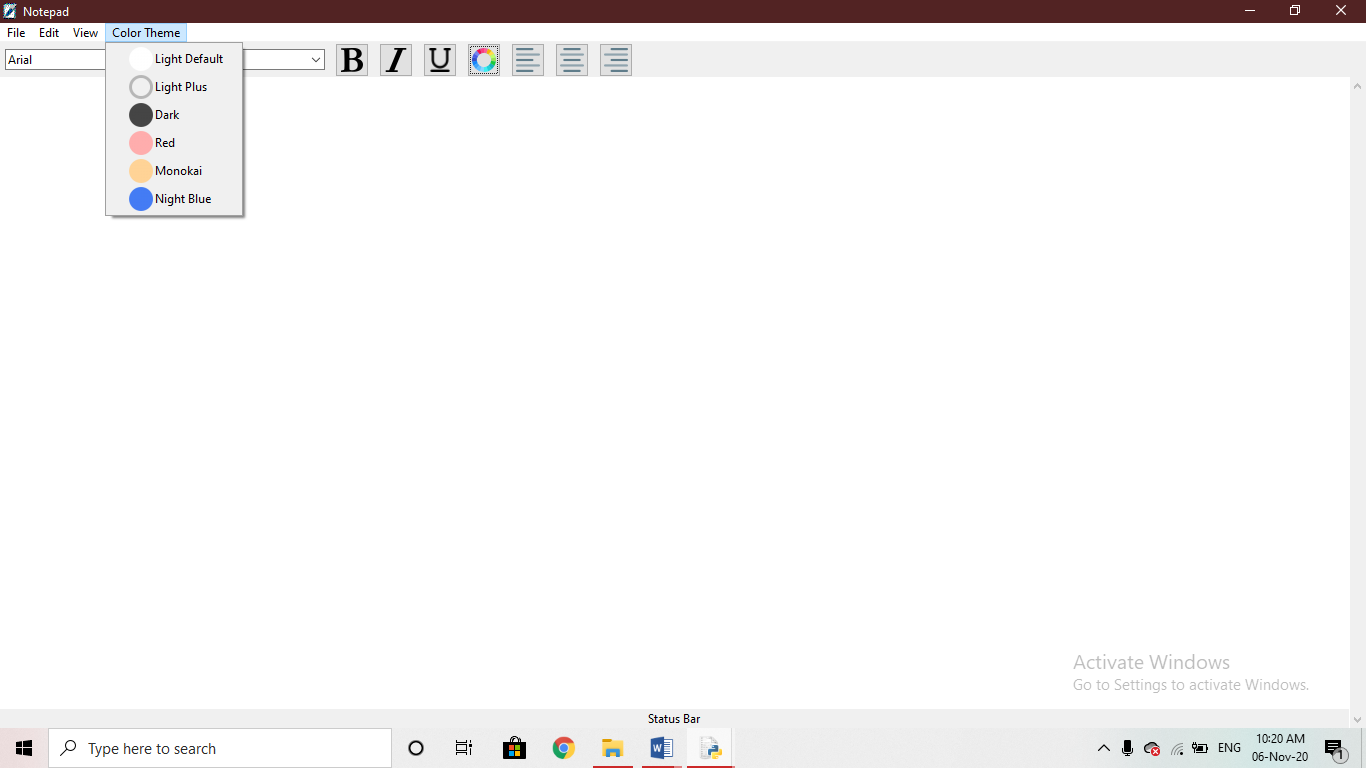
* Fonts
* Font Size



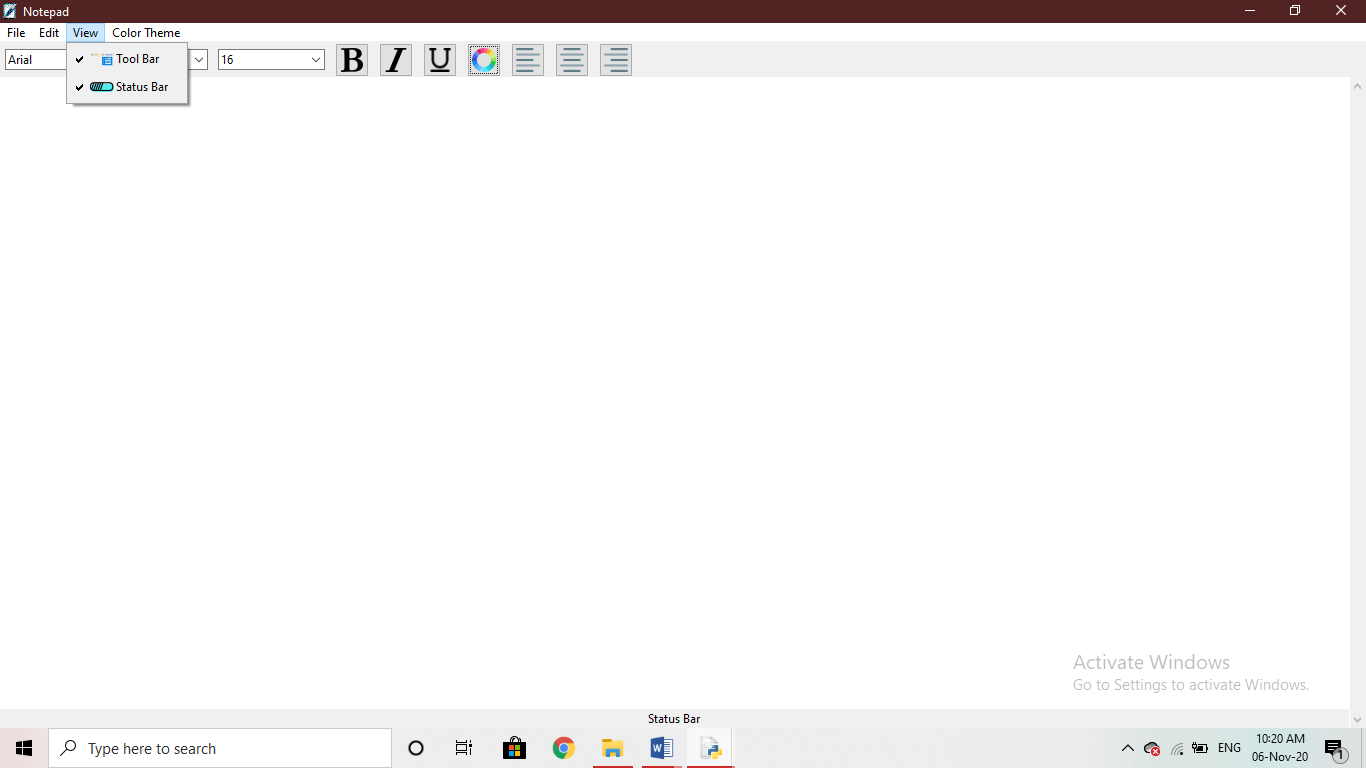
* Font Colour



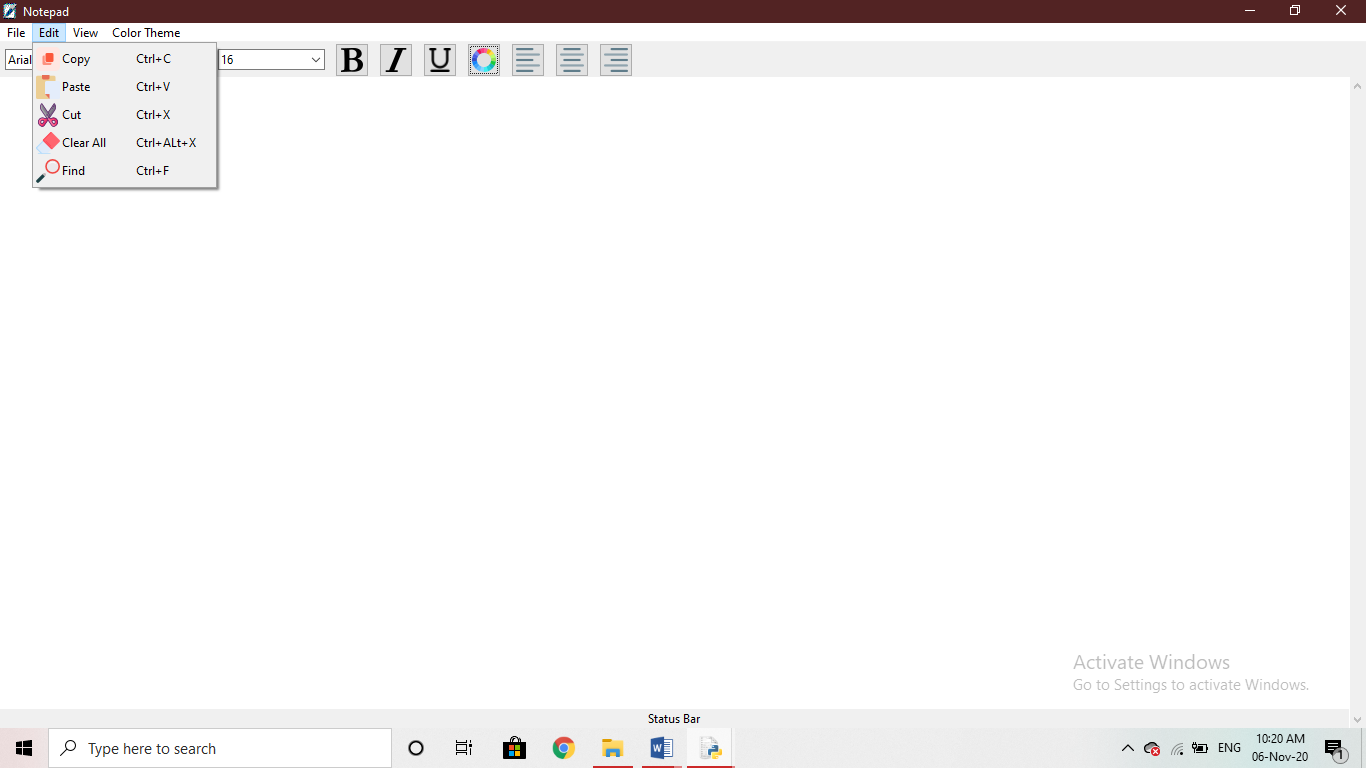
* Themes



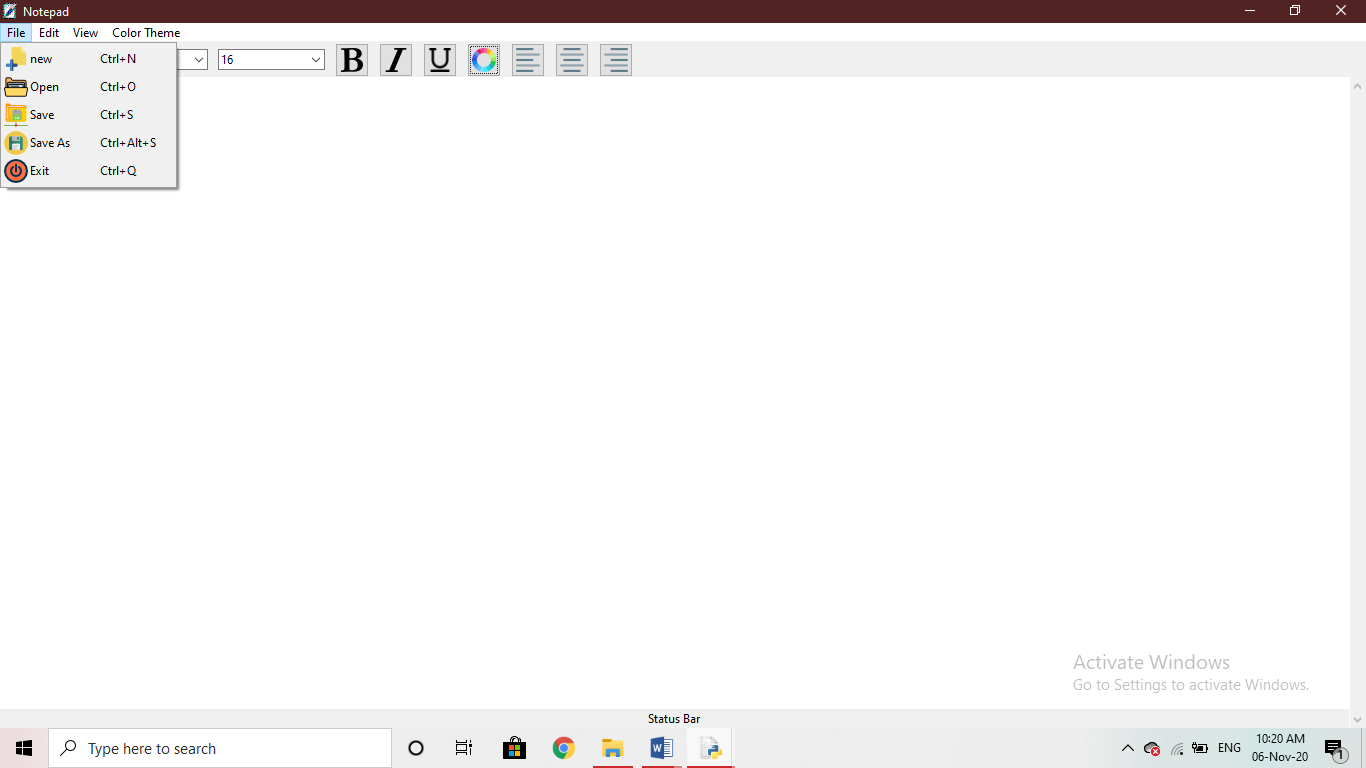
* Toolbars



* Edit

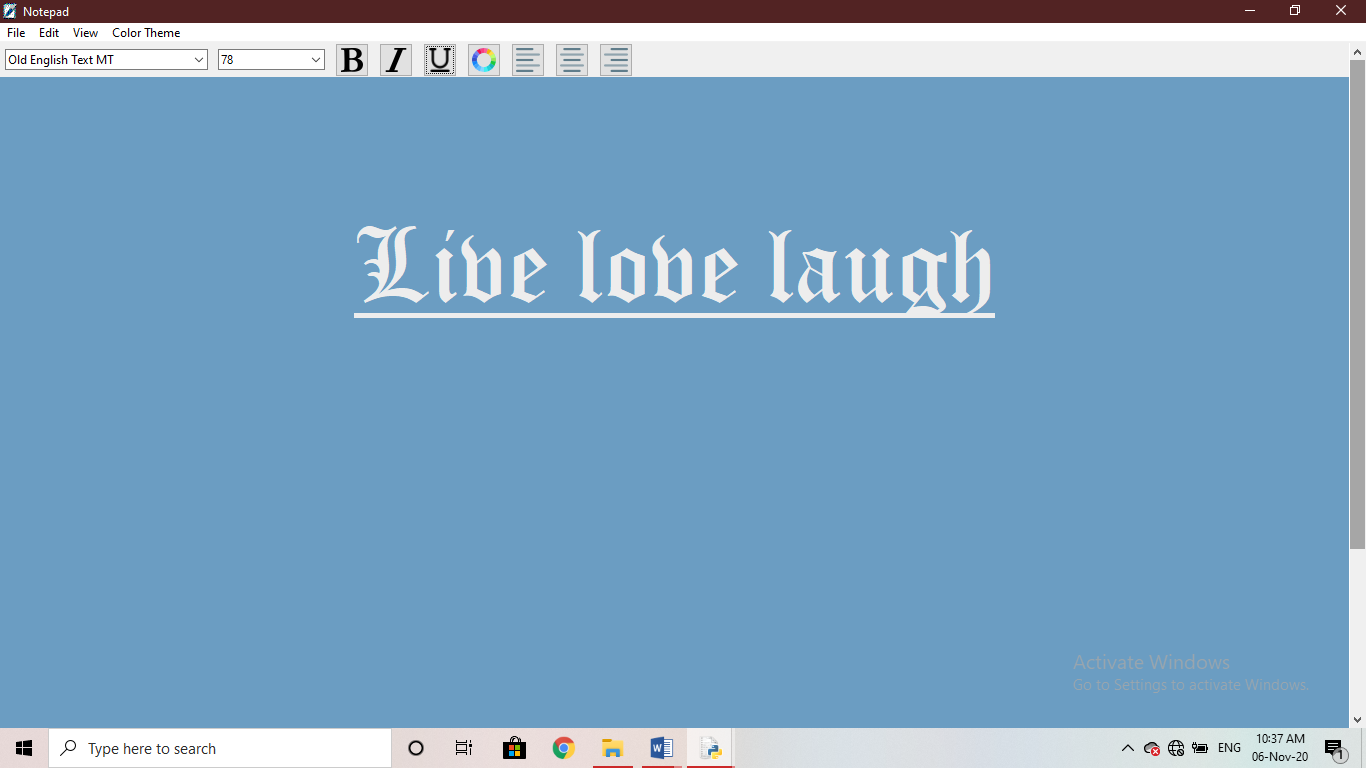


* Files

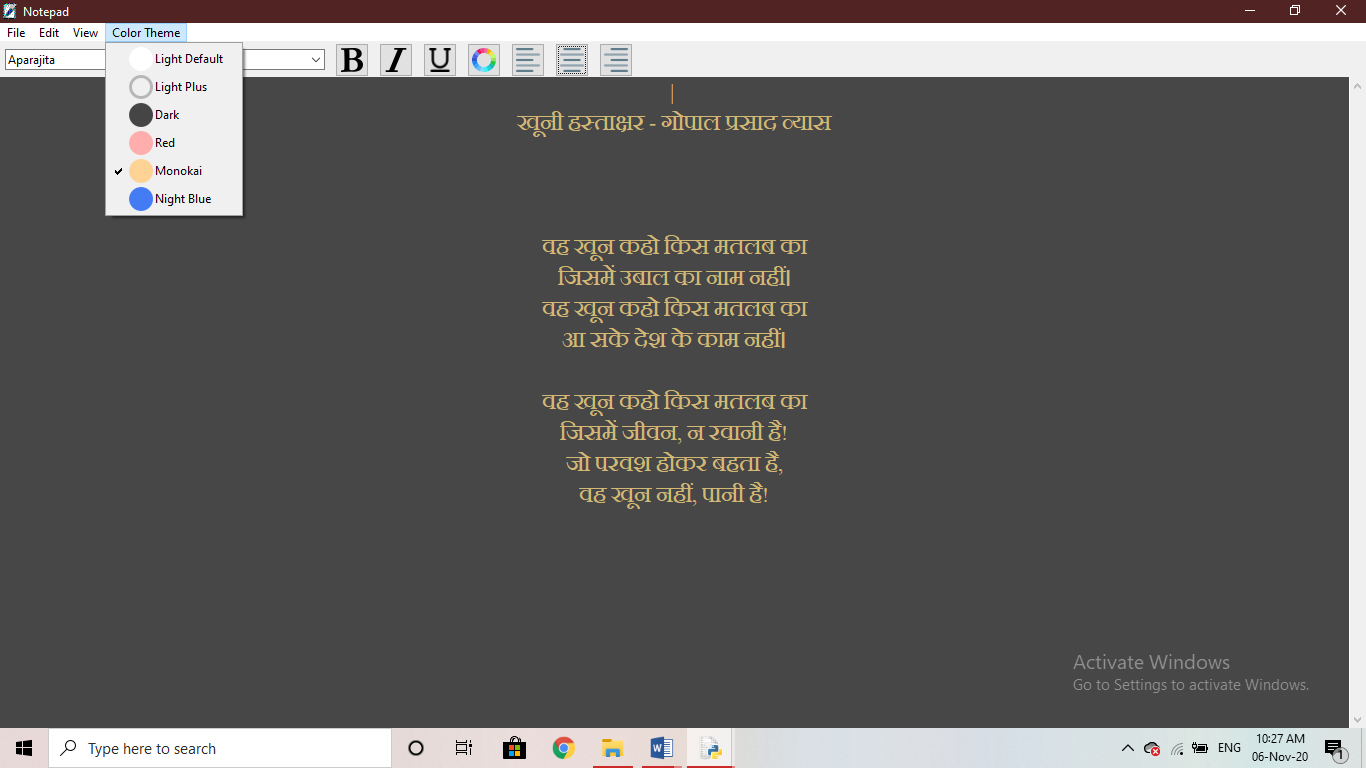


Testing

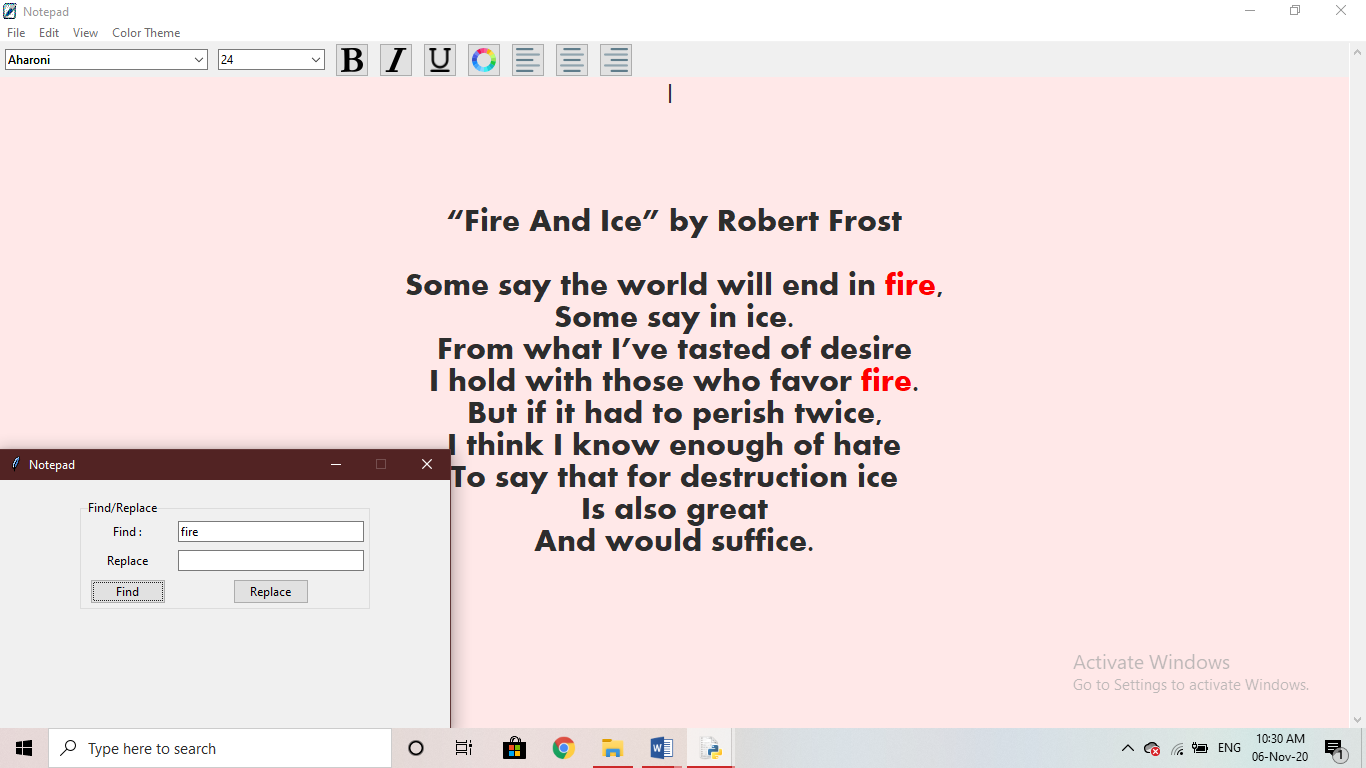
* Changing Font Style



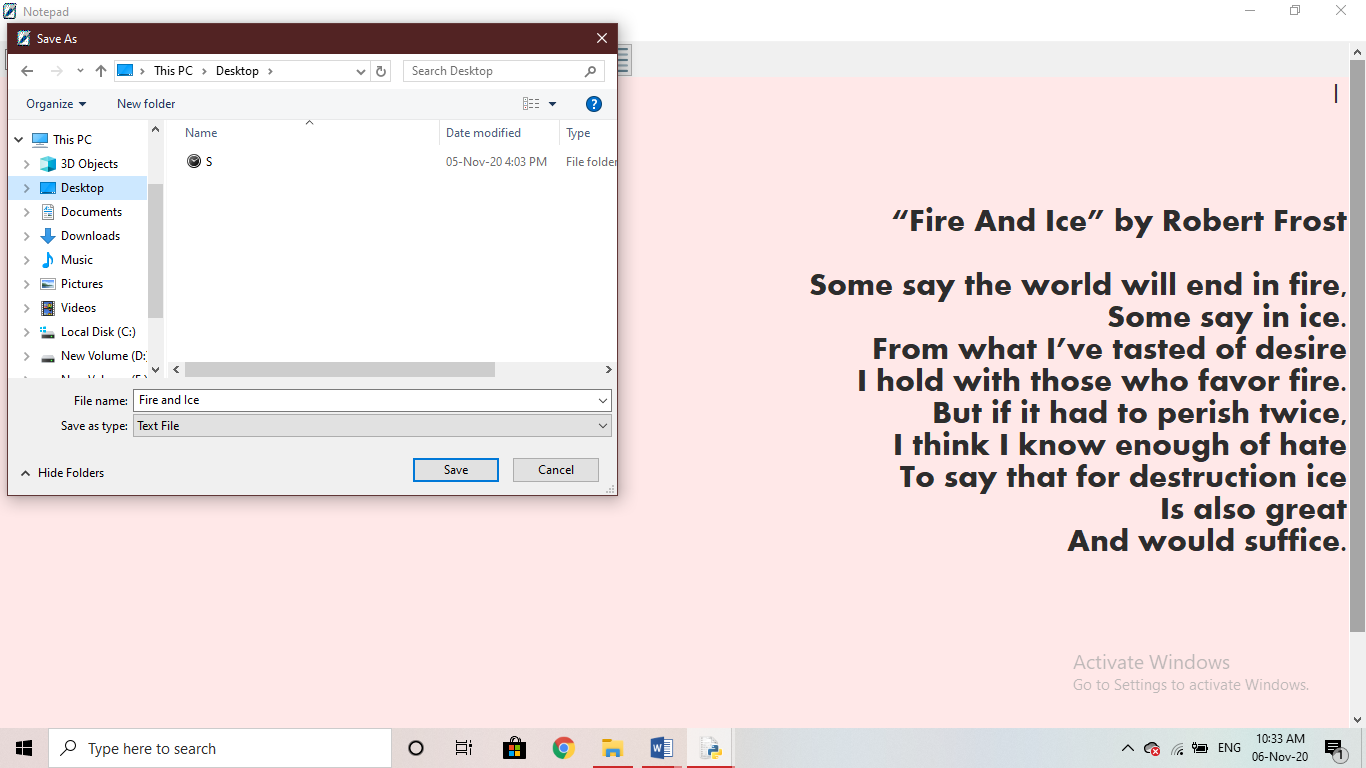
* Changed Theme



* Find And Replace



* Saving a File

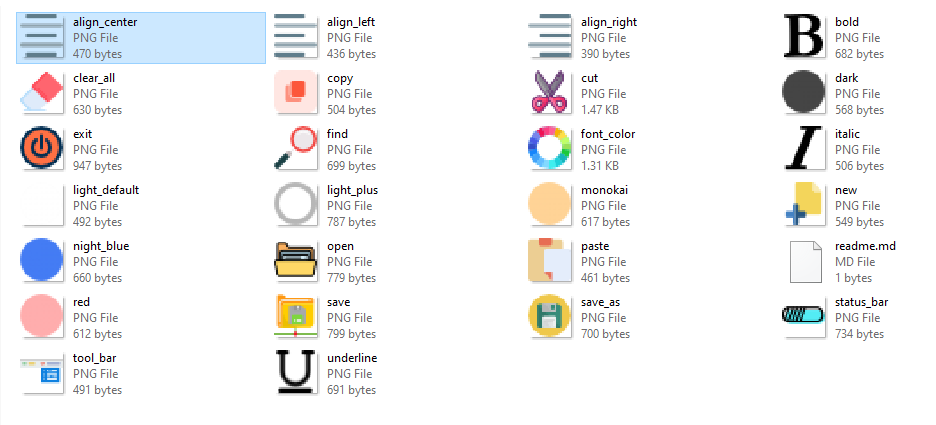


C

Software Requirements

* Windows OS
* Python IDLE/Spyder/Pycharm
* Files Attached;

1. mainicon 
2. icons2



Conclusion

Here, using tkinter I have created a notepad like text editor. This notepad will have the menu where we can create new file, open existing file, save the file, editing, cut and paste, all functionality will be there. I have also added the function to change the themes or the background of the text editor and simultaneously we can also change the foreground as well. This small program has helped me enhance my knowledge on the python GUI modules and surely proved to be helpful in future.

Bibliography

* Referrences taken from;
* Computer Science with Python Textbook of class XI: Preeti Arora
* Computer Science with Python Textbook of class XII: Sumita Arora
* Computer Science with Python Textbook of class XII: Preeti Arora
* Websites:

1. <https://docs.python.org/3/library/tkinter.html>
2. <https://wiki.python.org/moin/TkInter>
3. <https://wiki.python.org/moin/TkInter>