CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

TEXT EDITOR USING PYTHON

PROJECT REPORT

BY

ANKIT KUMAR

18bca1104

UNIVERSITY INSTITUTE OF COMPUTING

CHANDIGARH UNIVERSITY

Ghraun,mohali,punjab

CONTENTS:

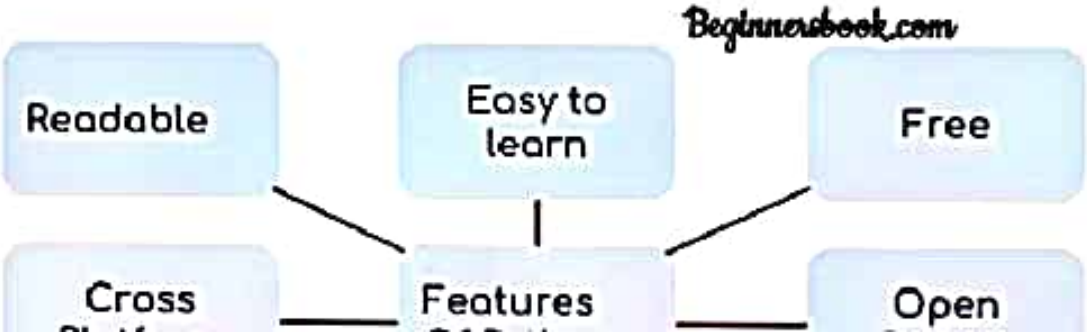
Abstract:

In this project I have used several kinds of Python feature such as Tkinter+database(sqlite3) etc. Name of my text editor is V PAD TEXT EDITOR.Ihave used visual code platform to develop the text editor. There are severalkinds of functions available in my text editor like we can make our text bold,italic, normal and colourful. I have developed a real software which canperform all the basic task similar to MS Word. I have used several kinds oficons for better navigation, which is very-very useful for the user. This texteditor is helpful for beginners who want to learn the typing. It is very-veryfriendly for the new users. Tkinter interface used for Text editor. Basic taskslike cursor, line number File handling to manage the text files. Different iconsfor cut, copy, info, find_ text, new file, open file. Line numbers and columnnumber display at the bottom. To open and save files, the project is usingPython file handling system and its basic method. The text editor offersfunctionalities like open, close, edit and create new files using Python.Texteditor is an application like notepad to write documents and other files

INTRODUCTION

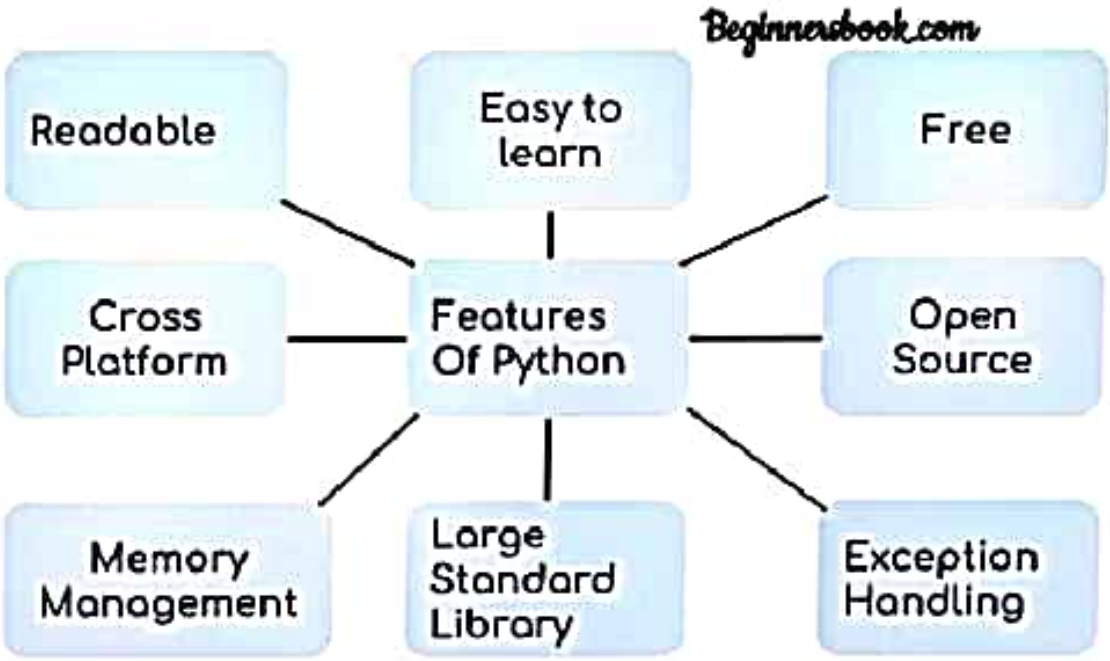
Python is developed by **Guido van Rossum**. Guido van Rossum started implementing Python in 1989. Python is a very simple programming language so even if you are new to programming, you can learn python without facing any issues

Features of Python programming language



Python is developed by **Guido van Rossum**. Guido van Rossum started implementing Python in 1989. Python is a very simple programming language so even if you are new to programming, you can learn python without facing any issues

Features of Python programming language



1. **Readable:** Python is a very readable language.

2. **Easy to Learn:** Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn.

3. **Cross platform:** Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.

4. **Open Source:** Python is a open source programming language.

5. **Large standard library:** Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.

6. **Free:** Python is free to download and use. This means you can download it for free and use it in your application. See: Open Source Python License. Python is an example of a FLOSS (Free/Libre Open Source Software), which means you can freely distribute copies of this software, read its source code and modify it.

7. **Supports exception handling:** If you are new, you may wonder what is an exception? An exception is an event that can occur during program exception and can disrupt the normal flow of program. Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.

8. **Advanced features:** Supports generators and list comprehensions. We will cover these features later.

9. **Automatic memory management:** Python supports automatic memory management which means the memory is cleared and freed automatically. You do not have to bother clearing the memory.

What Can You Do with Python?

You may be wondering what all are the applications of Python. There are so many applications of Python, here are some of the them.
1. Web development – Web framework like Django and Flask are based on Python. They help you write server side code which helps you manage database, write backend programming logic, mapping urls etc.

2. Machine learning – There are many machine learning applications written in Python. Machine learning is a way to write a logic so that a machine can learn and solve a particular problem on its own. For example, products recommendation in websites like Amazon, Flipkart, eBay etc. is a machine learning algorithm that recognises user's interest. Face recognition and Voice recognition in your phone is another example of machine learning.

3. Data Analysis – Data analysis and data visualisation in form of charts can also be developed using Python.

4. Scripting – Scripting is writing small programs to automate simple tasks such as sending automated response emails etc. Such type of applications can also be written in Python programming language.

5. Game development – You can develop games using Python.

6. You can develop Embedded applications in Python.

7. Desktop applications – You can develop desktop application in Python using library like TKinter or QT.

OBJECTIVE AND SCOPE:

A text editor is a computer pro有ram that lets a user enter, chan有e, store, and usually print text "characters and
numbers,each encoded by the computer and its input and outputdevic es, arran有ed to have meanin有 to users or to other pro有rams#.! ext editors can be used to enter pro有ram lan有ua有e sourcestatements or to create documents such as technical manuals

INRO OF PROJECT:

V PAD TEXT EDITOR:
PROS:

PROS

- **Faster to Edit**: For simple edits, it is often faster to make changes to a page using a text editor.
- **Helps You Learn HTML**: Text editors teach you to read HTML. They often have wizards and functions to do the more common tasks (like the basic page tags), but you'll learn HTML and basic coding if you use a text editor.
- **More Marketable**: A web developer who can write HTML using a text editor will be more marketable than one who can only use a WYSIWYG editor. The former is more flexible and can get up to speed on any HTML editing tool, while the latter has to start all over with each new editing tool.
- **No "Funky" HTML**: The only HTML that will be in the document will be tags that you placed there deliberately. This will help your pages download faster, as well as look leaner.

- **Human Readable HTML**: This is especially important if you work on a team of web developers. The HTML can be spaced as your team likes it, and include comments or other notes to allow more efficient editing by other team members.

CONS

- **Must Know HTML**: While most HTML text editors can help with tags and suggest attributes and so on, these helpers are no substitute for knowing HTML. Most modern text editors offer drag-and-drop styles such as bold and italic, but if you can't remember the code for "non-breaking space" your editor might not be able to help.
- **Steeper Learning Curve**: Because you have to learn both HTML and the editor functions itself, a beginner will find a text editor more difficult to use.
- **Harder to "Design" With**: Some people find text editors more difficult to design pages in because they can't visualize how the page will look from just the HTML.

TECHNOLOGIES USED:

PYTHON

TKinter

GUI

SYSTEM RERQUIREMENTS:

HARDwARE

:-

The minimum hardware configuration for application to runis as follows.

Microprocessor :- Intel Pentium600MHZ or equivalent

Ram: 256 mb

Hard drive 106.1 gb

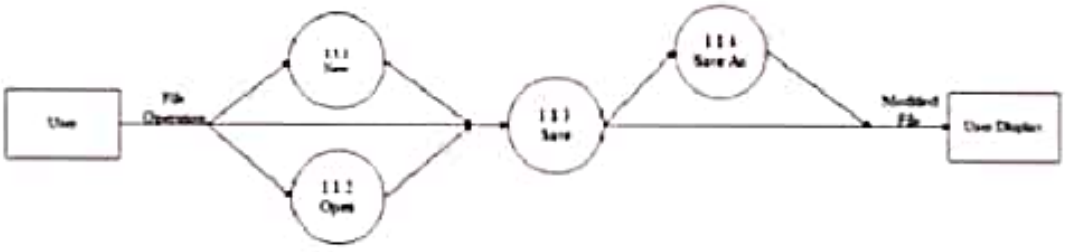
Software:
Operating system
Windows:7/8/10

DATA FLOW DIAGRAMS:

Data Flow Diagrams:

Level 0

Level 1



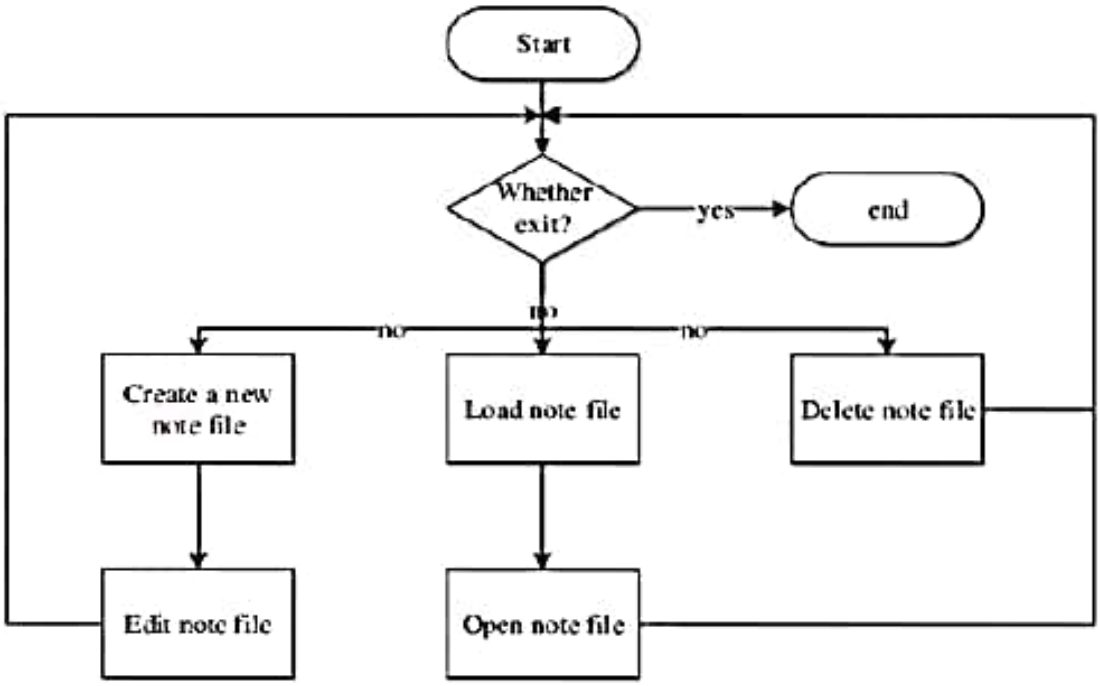
ALGORITHMS AND FLOW CHARTS:

Algorithm:

The word Algorithm means "a process or set of rules to be followed in calculations or other problem-solving operations". Therefore Algorithm refers to a set of rules/instructions that step-by-step define how a work is to be executed upon inorder to get the expected results.

Flowchart:

A flowchart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing.
The process of drawing a flowchart for an algorithm is known as "flowcharting".



SOURCE CODE FOR PROJECT:

```
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('Vpad text editor')

main_application.wm_iconbitmap('icon.ico')



################################################ main menu
####################################################

# ----------------------------------&&&&&&&& End 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)



######## 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)



################################################ 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(8,81))
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  &&&&&&&&&& ----------------------


############################################## 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.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 functionlaity
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

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 functinality
#######################################################


## 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)

## save file
```

```
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 = 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=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_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 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', command = find_func)

## 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 = color_dict.get(chosen_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&&&&&&&&&&
-------------------------



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()
```

EXECUTABLE SETUP FILE:

```
import cx_Freeze
import sys
import os
base = None

if sys.platform == 'win32':
    base = "Win32GUI"

os.environ['TCL_LIBRARY'] = r"C:\Users\Admin\AppData\Local\Programs\Python\Python37\tcl\tcl8.6"
os.environ['TK_LIBRARY'] = r"C:\Users\Admin\AppData\Local\Programs\Python\Python37\tcl\tk8.6"

executables = [cx_Freeze.Executable("vpad.py", base=base, icon="icon.ico")]


cx_Freeze.setup(
```

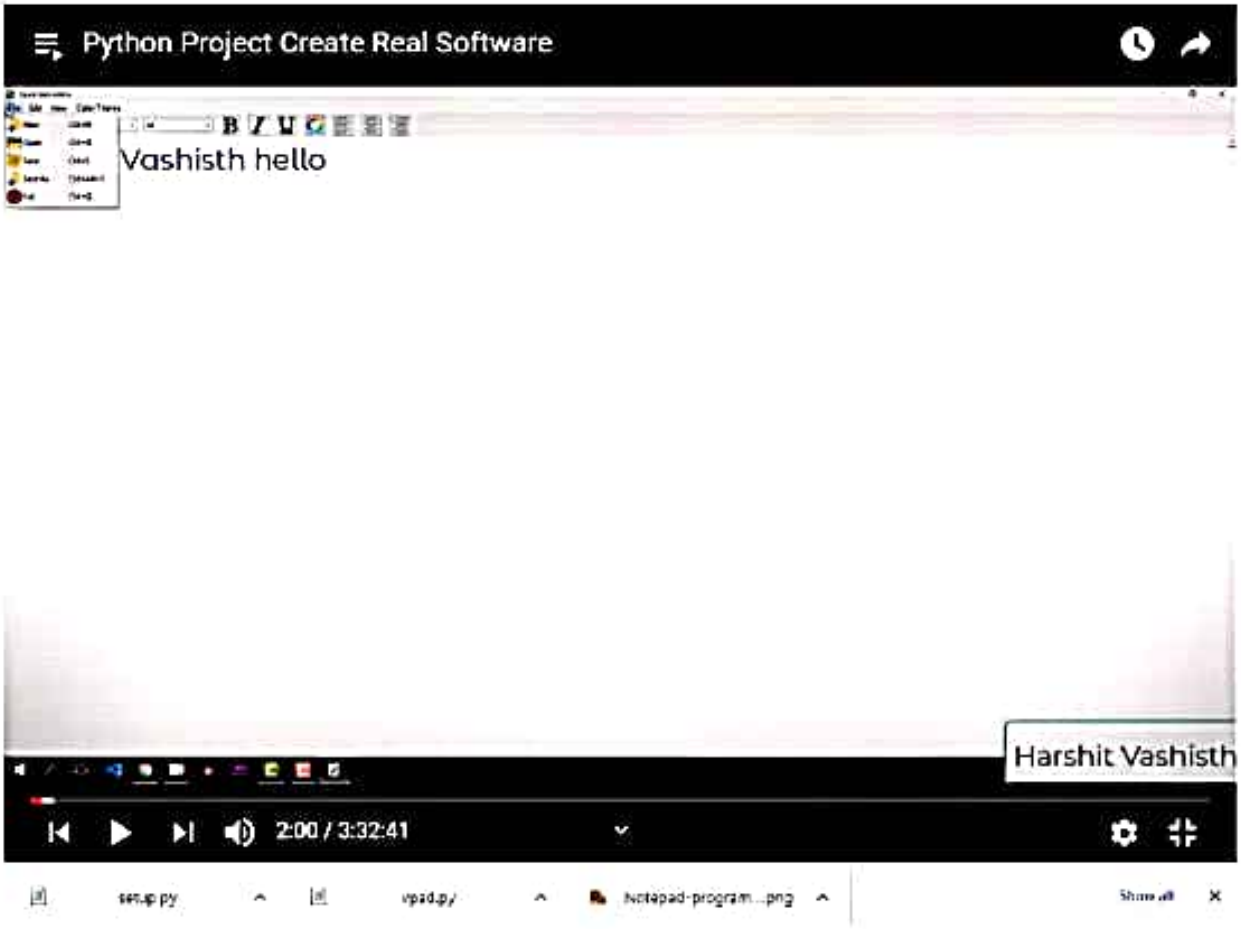
```
name = "Vpad Text Editor",
options = {"build_exe": {"packages":["tkinter","os"], "include_files":["icon.ico",'tcl86t.dll','tk86t.dll', 'icons2']}},
version = "0.01",
description = "Tkinter Application",
executables = executables
)
```

SCREENSHOTS:


shisth hello


lo


Vashisth hello


Python Project Create Real Software
Vashisth hello
Harshit Vashisth
2:00 / 3:32:41
setup.py
vpad.py
notepad-program...png
Show all


Harsnit Vashisth hello