ASHISH'S TEXT EDITOR

DEPARTMENT OF B.E(CSE)

UIET, PANJAB UNIVERSITY SSG REGIONAL CENTRE,
HOSHIARPUR-146021, Punjab (INDIA)

(2018-2019)

Submitted to:

Ms. Sukhpreet Kaur
Ms. Shama Pathania
(Prof. CSE Dept)

Submitted by:

Ashish Kumar Singh
Roll no. SG15313
CSE(7^{TH} SEM)

A REPORT ON

Text editor with 'python'

SUBMITTED IN PARTIAL FULFILLMENT FOR AWARD DEGREE OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

BY

Ashish Kumar singh

(SG-15313)

DEPARTMENT OF B.E(CSE)

UIET, PANJAB UNIVERSITY SSG REGIONAL CENTRE,
HOSHIARPUR-146021, Punjab (INDIA)

(2018-2019)

DECLARATION

The work embodied in the training report entitled, **"ASHISH'S TEXT"** submitted to the department of **Computer science & engg** at UIET, Panjab University Swami Sarvanand Giri Regional Centre, Hoshiarpur for the award of degree of Bachelor of Engineering, has been done by me. The training report is entirely based on my own work and not submitted elsewhere for the award of any other degree. All ideas and references have been duly acknowledged.

Name and Signature of student

COMPANY CERTIFICATE

ACKNOWLEDGMENT

I owe my deepest gratitude to the Department of Computer science & engg, U.I.E.T, Hoshiarpur for providing me with an opportunity to work on a project as a part of our syllabus. It is an honor for us to express our gratitude to our project supervisors for their constant support and guidance throughout the project.

I am grateful to all our teachers for their suggestions and inspirational lectures that paved the way towards the completion of this project.

Last but not the least; I would like to thank our colleagues for their valuable comments and their suggestion during this project.

ABSTRACT

TEXT EDITIOR GUI

In this Instructable, I have created a simple text editor with Python and the module Tkinter.

- A text editor is a type of computer program that edits plain text.
- Such programs are sometimes known as "notepad" software.
- A text editor is a program that saves your files without formatting.
- Text editors are provided with operating systems and software development packages, and can be used to change configuration files, documentation files and programming language source code.
- That gives us a simple Text Editor designed in Python & Tkinter with the basic functionalities.
- This application is made using Python 3.0, libraries,modules.

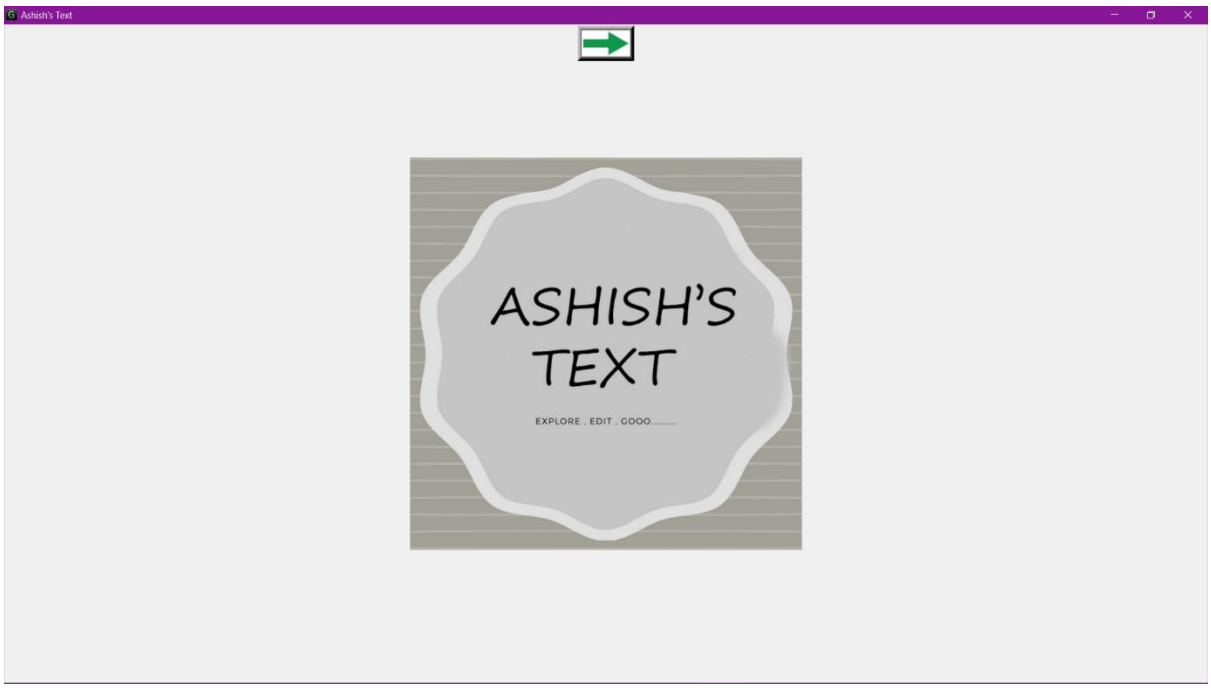


Fig (1): Text Editor

List of Tables/Figures/Abbreviations

LIST OF SYMBOLS,

Symbols	Title	Page No.
Symbol (1)	Tkinter	
Symbol (1)	GUI	
Symbol (1)	Requests Libraries	

Contents

CHAPTER-1
INTRODUCTION

I. INTRODUCTION TO PROJECT

1.1 Text Editor:

A text editor is a computer program that lets a user enter, change, store, and usually print text (characters and numbers, each encoded by the computer and its input and output devices, arranged to have meaning to users or to other programs). Typically, a text editor provides an "empty" display screen (or "scrollable page") with a fixed-line length and visible line numbers. You can then fill the lines in with text, line by line. A special command line lets you move to a new page, scroll forward or backward, make global changes in the document, save the document, and perform other actions. After saving a document, you can then print it or display it. Before printing or displaying it, you may be able to format it for some specific output device or class of output device. Text editors can be used to enter program language source statements or to create documents such as technical manuals.

Fig (1.1): Text Editor Symbol

Main Features -

1. Menubar

2. Toolbar

3. Line and column number at bottom right

4. Font color chooser

5 .Background color chooser

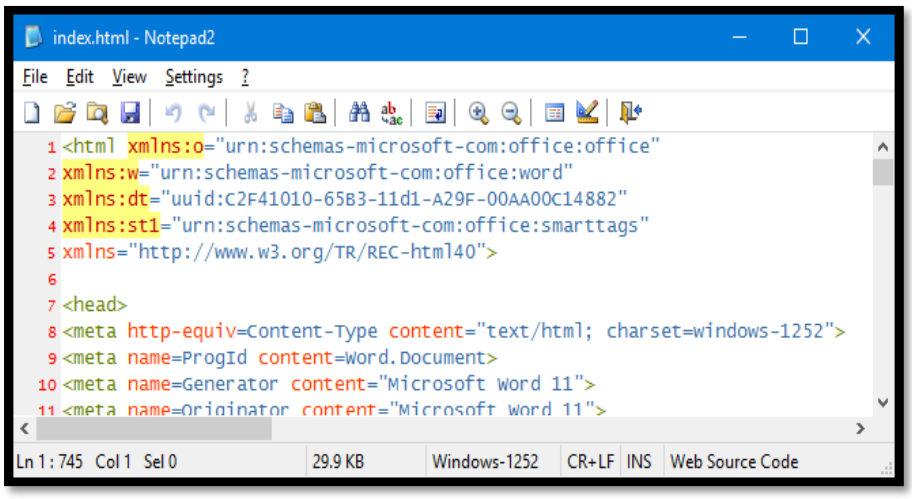


Fig: (1.2): Notepad

II. INTRODUCTION TO TEXT EDITOR:

A text editor is a type of computer program that edits plain text. There are important differences between plain text (created and edited by text editors) and rich text (such as that created by word processors or desktop publishing software).

Plain text exclusively consists of character representation. Each character is represented by a fixed-length sequence of one, two, or four bytes, or as a variable-length sequence of one to four bytes, in accordance to specific character encoding conventions, such as ASCII, ISO/IEC 2022, UTF-8, or Unicode. These conventions define many printable characters, but also non-printing characters that control the flow of the text, such space, line break, and page break. Plain text contains no other information about the text itself, not even the character encoding convention employed. Plain text is stored in text files, although text files do not exclusively store plain text. In the early days of computers, plain text was displayed using a monospace font, such that horizontal alignment and columnar formatting were sometimes done using whitespace characters. For compatibility reasons, this tradition has not changed.

Rich text, on the other hand, may contain metadata, character formatting data (e.g. typeface, size, weight and style), paragraph formatting data (e.g. indentation, alignment, letter and word distribution, and space between lines or other paragraphs), and page specification data (e.g. size, margin and reading direction). Rich text can be very complex. Rich text can be saved in binary format (e.g. DOC), text files adhering to a markup language (e.g. RTF or HTML), or in a hybrid form of both (e.g. Office Open XML).

Text editors are intended to open and save text files containing either plain text or anything that can be interpreted as plain text, including the markup for rich text or the markup for something else (e.g. SVG).

Fig (1.3): Logo

Before text editors existed, computer text was punched into cards with keypunch machines. Physical boxes of these thin cardboard cards were then inserted into a card-reader. Magnetic

tape and disk "card-image" files created from such card decks often had no line-separation characters at all, and assumed fixed-length 80-character records. An alternative to cards was punched paper tape..It could be created by some teleprinters (such as the Teletype), which used special characters to indicate ends of records.

The first text editors were "line editors" oriented to teleprinter- or typewriter-style terminals without displays. Commands (often a single keystroke) effected edits to a file at an imaginary insertion point called the "cursor". Edits were verified by typing a command to print a small section of the file, and periodically by printing the entire file. In some line editors, the cursor could be moved by commands that specified the line number in the file, text strings (context) for which to search, and eventually regular expressions. Line editors were major improvements over keypunching. Some line editors could be used by keypunch; editing commands could be taken from a deck of cards and applied to a specified file. Some common line editors supported a "verify" mode in which change commands displayed the altered lines.

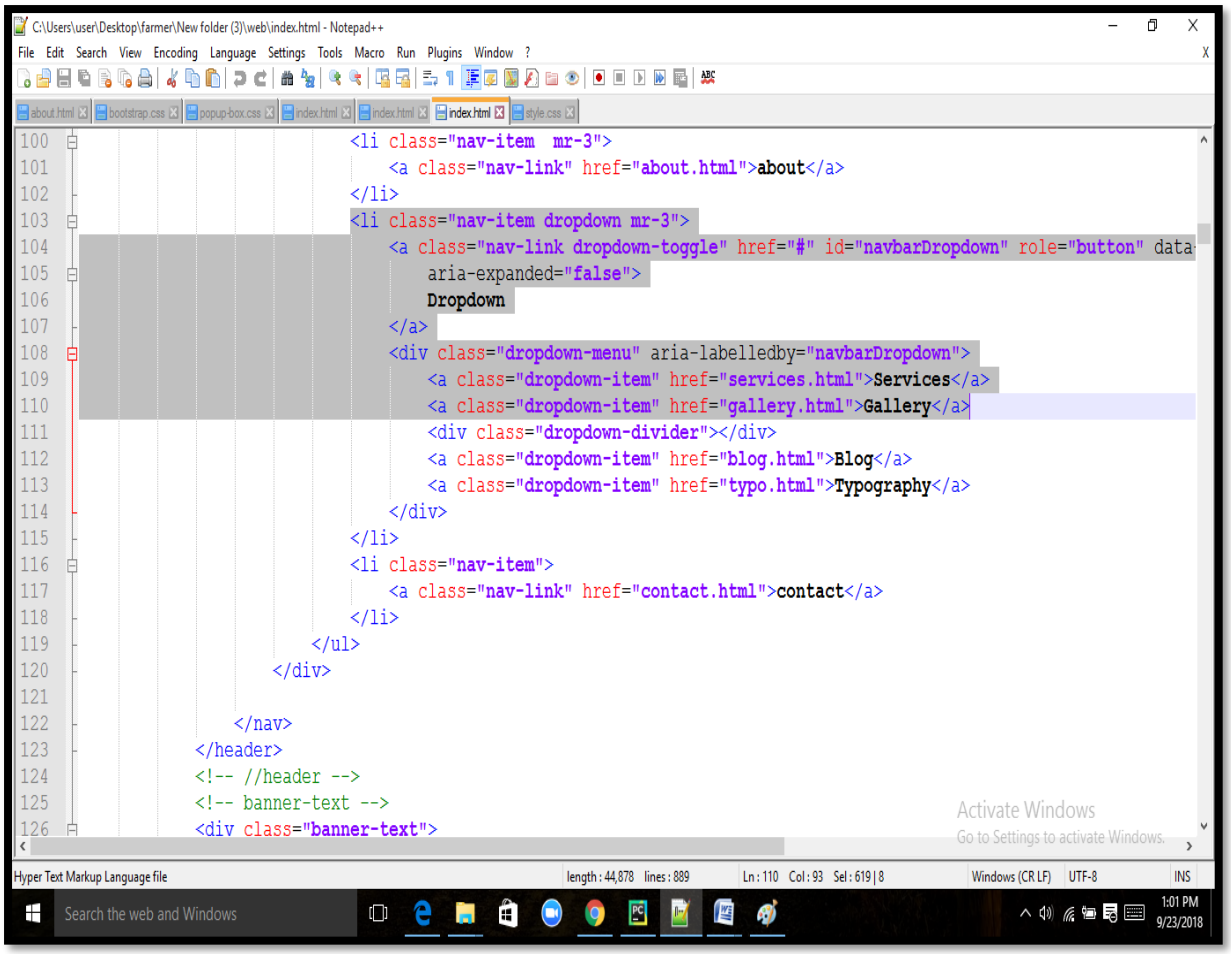


Fig :(1.4) :Notepad

CHAPTER-2

TECHNOLOGY USED

I. BACKEND DEVELOPMENT

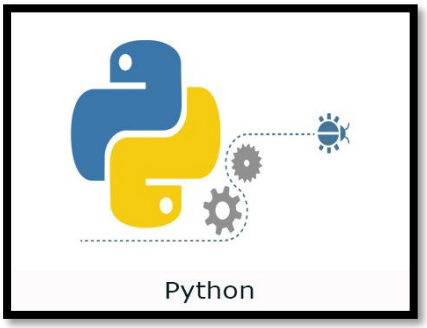
2.1 Python 3.0:

Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely backward-compatible with previous versions. However, many of its major features have been backported to the Python 2.6.x and 2.7.x version series, and releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.0

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter (), map (), and reduce () functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.



Fig(2.1):Python Logo

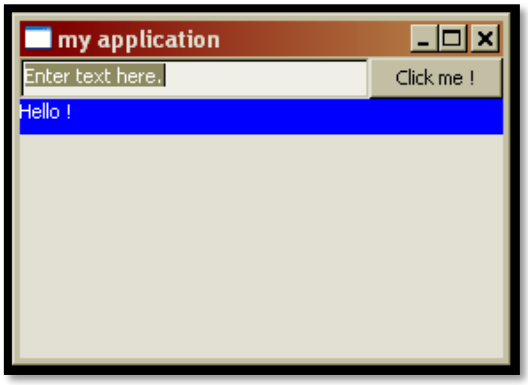
Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

Installing

Installing Python is generally easy, and nowadays many Linux and UNIX distributions include a recent Python. Even some Windows computers (notably those from HP) now come with Python already installed. If you *do* need to install Python and aren't confident about the task you can find a few notes on the Beginners Guide/Download wiki page, but installation is unremarkable on most platforms.

Tkinter

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. Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.



Fig(2.2):Tkinter

The Tkinter Interface GUI Using For making the app of Text editor GUI. By using Tkinter we implement design for the text editor.

- Help individuals share their own content with 3rd party apps.
- Help brands and advertisers understand, manage their audience and media rights.
- Help broadcasters and publishers discover content, get interface by using Tkinter GUI Interface.Visit this link to view About Tkinter: https://wiki.python.org/moin/TkInter/

II. PYCHARM COMMUNITY EDITION

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition released under a proprietary license - this has extra features.

The beta version was released in July 2010, with the 1.0 arriving 3 months later. Version 2.0 was released on 13 December 2011, version 3.0 on 24 September 2013, and version 4.0 on November 19, 2014.PyCharm Community Edition, the open source version of PyCharm, became available on 22 October 2013.

Fig(2.3):Pycharm Logo

Features

- Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes.
- Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages.

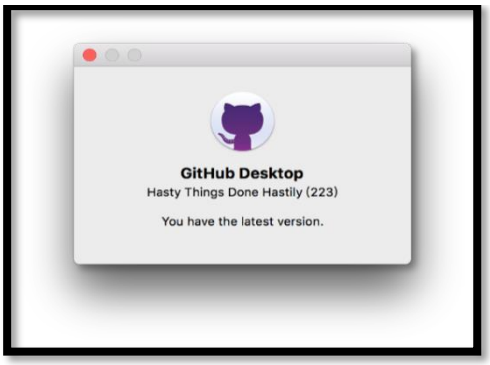
- Python refactoring: including rename, extract method, introduce variable, introduce constant, pull up, push down and others.
- Support for web frameworks: Django, web2py and Flask.
- Integrated Python debugger.
- Integrated unit testing, with line-by-line code coverage.
- Google App Engine Python development.

III. GITHUB DESKTOP

GitHub Desktop is currently only available for Windows and Mac. If you use Linux you will probably already be familiar with the command line and will be able to use the Command Line version of Git.

It is helpful to understand what version control is and why it might be useful for the work you are doing prior to getting stuck into the practicalities. At a basic level version control involves taking ‘snapshots’ of files at different stages. Many people will have introduced some sort of version control systems for files.Projects on GitHub can be accessed and manipulated using the standard Git command-line interface and all of the standard Git commands work with it. GitHub also allows registered and non-registered users to browse public repositories on the site. Multiple desktop clients and Git plugins have also been created by GitHub and other third parties that integrate with the platform

It is helpful to understand what version control is and why it might be useful for the work you are doing prior to getting stuck into the practicalities. At a basic level version control involves taking ‘snapshots’ of files at different stages. Many people will have introduced some sort of version control systems for files.Projects on GitHub can be accessed and manipulated using the standard Git command-line interface and all of the standard Git commands work with it. GitHub also allows registered and non-registered users to browse public repositories on the site. Multiple desktop clients and Git plugins have also been created by GitHub and other third parties that integrate with the platform. The site provides social networking-like functions such as feeds, followers, wikis (using wiki software called Gollum) and a social network graph to display how developers work on their versions ("forks") of a repository and what fork (and branch within that fork) is newest.



Fig(2.4):GitHub icon

The site provides social networking-like functions such as feeds, followers, wikis (using wiki software called Gollum) and a social network graph to display how developers work on their versions ("forks") of a repository and what fork (and branch within that fork) is newest.

CHAPTER-3

PROJECT WORK

3.1 TEXT EDITOR USING PYTHON

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a text editor GUI:

- Import module (Tkinter)
- Create a new window
- Add menu items
- Adding functionality in menu items
- Adding Icons and others functionality
- Adding line numbers and scrolling

3.1.1 Import Module:

Modules are Python .py files that consist of Python code. Any Python file can be referenced as a module. A Python file called hello.py has the module name of hello that can be imported into other Python files or used on the Python command line interpreter. Modules can define functions, classes, and variables that you can reference in other Python .py files or via the Python command line interpreter.

In Python, modules are accessed by using the import statement. When you do this, you execute the code of the module, keeping the scopes of the definitions so that your current file(s) can make use of these.When Python imports a module called hello for example, the interpreter will first search for a built-in module called hello. If a built-in module is not found, the Python interpreter will then search for a file named hello.py in a list of directories that it receives from the sys. path variable.

To make use of the functions in a module, you'll need to import the module with an import statement. An import statement is made up of the import keyword along with the name of the module.When we import a module, we are making it available to us in our current program as a separate namespace. This means that we will have to refer to the function in dot notation, as in [module]. [function].

To refer to items from a module within your program's namespace, you can use the from ... import statement. When you import modules this way, you can refer to the functions by name rather than through dot notationIn this construction, you can specify which definitions to reference directly. In other programs, you may see the import statement take in references to everything defined within the module by using an asterisk (*) as a wildcard, but this is discouraged by PEP 8.

```
#IMPORT MODULE
from PIL import ImageTK
from tkinter import *
import os
import tkinter.filedialog
import tkinter.messagebox
from tkcolorpicker import askcolor
```

Here, first call the from keyword, then tkinter for the module. Next, use the import keyword and call the specific function would like to use.

3.1.2 Create a new window new

Tkinter Python interface to Tcl/Tk. The Tkinter module ("Tk interface") is the standard Python interface to the Tk GUI toolkit. There are two main methods used you the user need to remember while creating the Python application with GUI.

Root window created. Here, that would be the only window, but you can later have windows within windows.

```
import sys
v=sys. python_version

if "2.7" in v:
    from Tkinter import *
elif "3.3" in v or "3.4" in v:
    from tkinter import *
root=Tk ("Text Editor")

root. mainloop ()
```

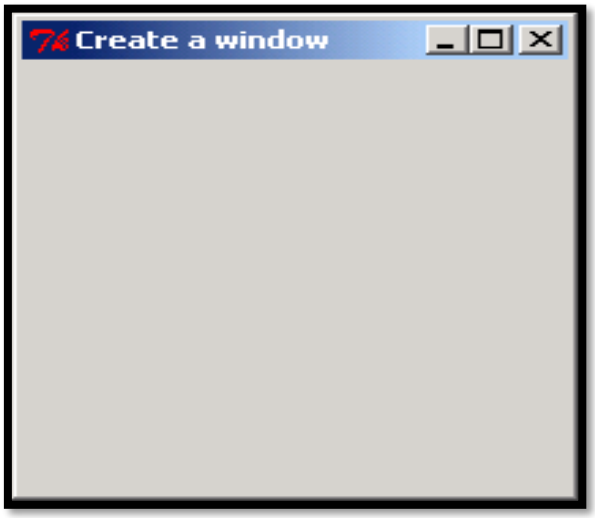
```
root =Tk ()
```

Then we actually create the instance.

```
app =Window(root)
```

Finally, show it and begin the mainloop.

```
root.mainloop ()
```



Fig(3.1):Text editor Window

3.1.3 Add Menu Items

Itis used to create all kinds of menus used by the application.The general syntax is:w = Menu (master, option=value) master is the parameter used to represent the parent window.There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas.

In the next step, we will add a menu bar with two main menu items: **File** , edit, Format , Run , Help Within **File** menu we will add 4 sub menu items **new**, **open**, **save** and **exit**. Within **Edit** , we will have one menu called **Cut , Copy , paste , delete, undo, redo.** In format , we do bold,italic,underline,color chooser for fonts and background. In help,some info about the developer will appear in the popup. In this section we simply add the menu button and assign each of them to a temporary dummy command function which we call “dummy” here. We will add the individual functionalities for each of these menu items in the next.

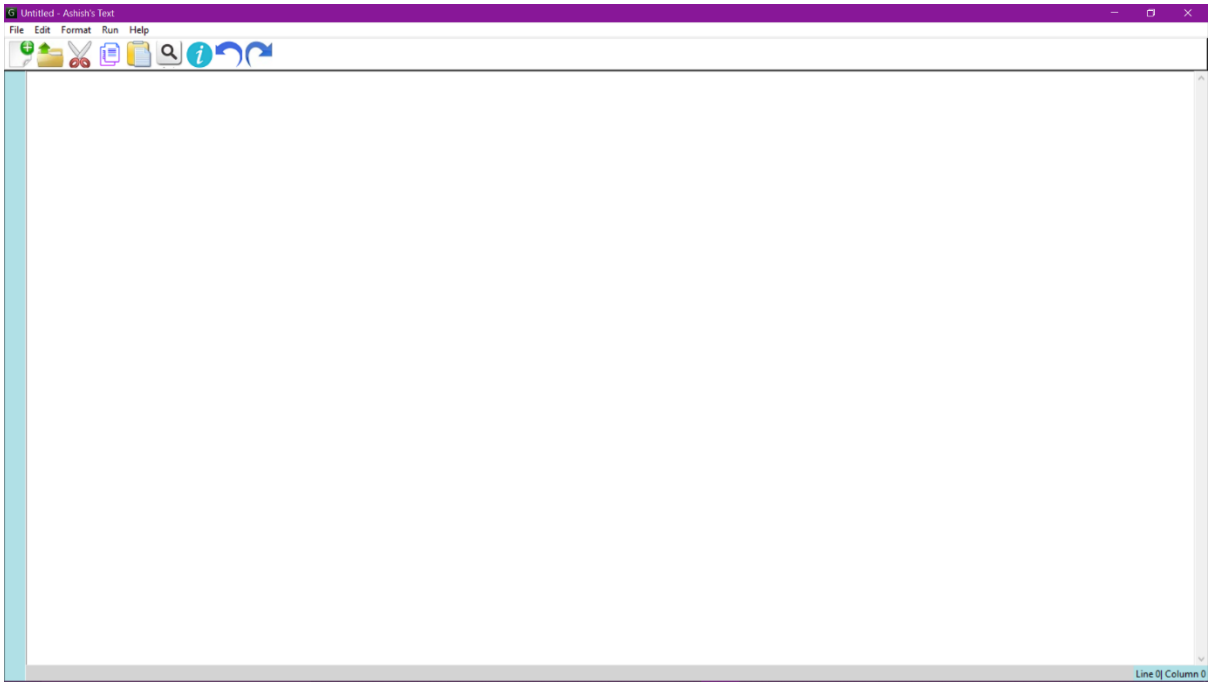


Fig (3.2):Menu items

Menus in GUIs are presented with a combination of text and symbols to represent the choices. Selecting with the mouse (or finger on touch screens) on one of the symbols or text, an action will be started. Such an action or operation can, for example, be the opening or saving of a file, or the quitting or exiting of an application. A context menu is a menu in which the choices presented to the user are modified according to the current context in which the user is located.We introduce in this chapter of our Python Tkinter tutorial the pull-down menus of Tkinter, i.e. the lists at the top of the windows, which appear (or pull down), if you click on an item like, for example "File", "Edit" or "Help"

3.1.4 Adding functionality in Menu items

The Button widget is a standard Tkinter widget, which is used for various kinds of buttons. A button is a widget which is designed for the user to interact with, i.e. if the button is pressed by mouse click some action might be started. They can also contain text and images like labels. While labels can display text in various fonts, a button can only display text in a single font. The text of a button can span more than one line.

All these functions exist without us having to add a single line of code toward these functionalities.Clearly the text widget comes built in with these events. Rather than coding these functions ourselves, let's use the built-in functions to add these features to our text editor.

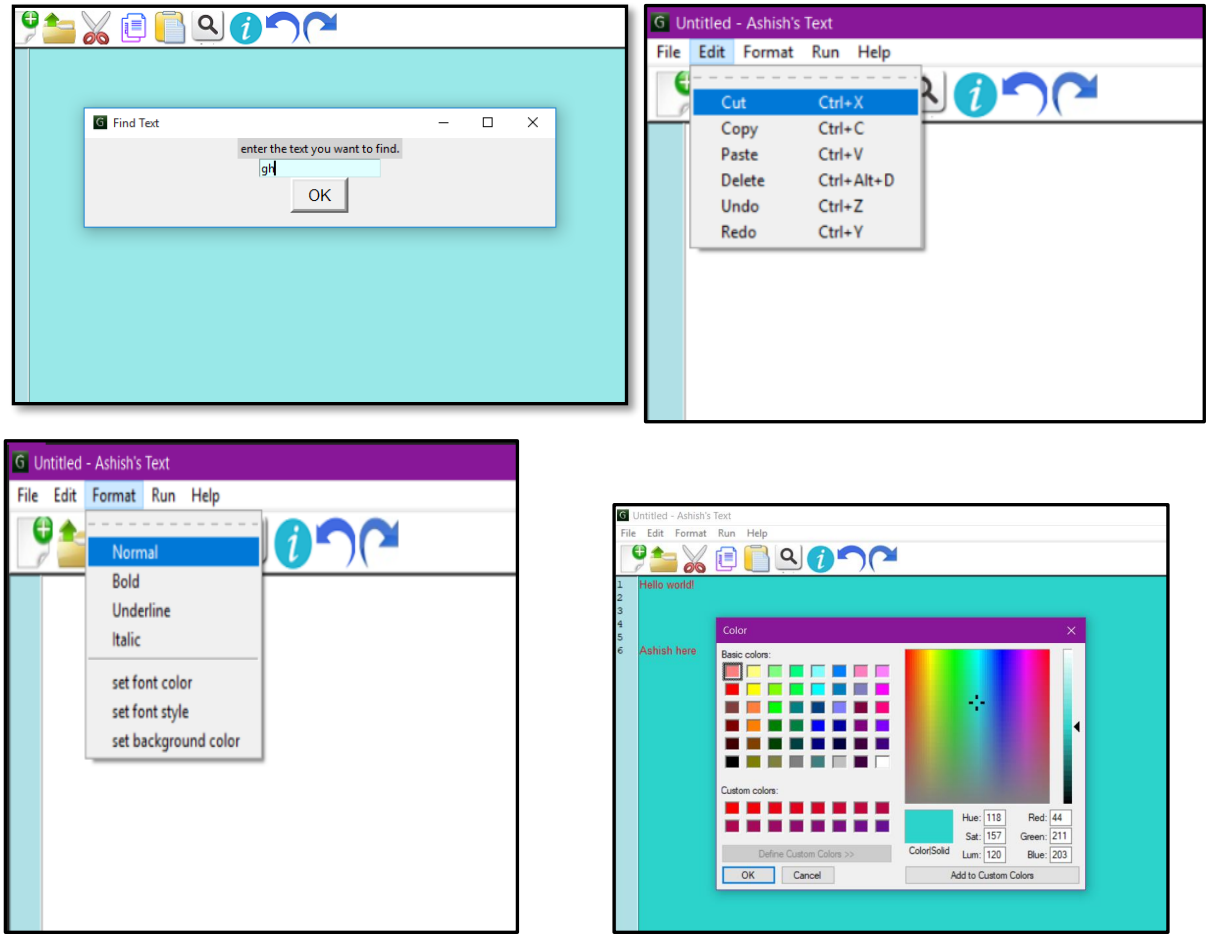
The documentation of TCL/TK “universal widgets methods “tells us that we can trigger events

Without any external stimulus using the following command. That gives us a simple Text Editor designed in Python &Tkinter with the basic functionalities. Implementing the Cut, Copy, and Paste features. We now have our editor GUI ready. If you open the program and play with the Text widget you will notice that you can perform basic functions such as cut, copy, and paste in the text area using the keyboard shortcuts Ctrl + X, Ctrl + C, and Ctrl + V.

The Open and Save dialogs. They are common across many programs. We know how these menu items behave. For instance, when you click on the Open menu, it opens up a

dialog form that lets you traverse to the location of the file you want to open. When you select a particular file and click on Open, it opens up in your editor. Similarly, we have the Save dialog.

The About and Help menus. The functionality is simple. When a user clicks on the Help or About menu, it pops up a message window and waits for the user to respond by clicking on a button. While we can easily code new Top-level windows to show our About and Help popup windows, we will instead use a module called tkMessageBox to achieve this functionality. This is because the module provides an efficient way to handle this and similar functionalities with minimal coding. We will also complete coding the Exit button functioning in this iteration. Currently, when a user clicks on the Close button, the window is simply closed. We want to ask the user if they really want to quit or have they clicked on the Close button accidentally.



Fig(3.3):Menu function

3.1.5 Adding icons and others functionality

The main goal of creating applications with a graphical interface is to build some comfortable interface for using and solving some tasks. We can add buttons, fields, frames, graphics and other things which we want to see on the couch. Buttons and text field would be good for getting effective results. For the best view of the buttons, we create a new class that will describe styling and behavior and set icons.We need to add some style, set size, and position to our application. Let's do this by adding some code. At first set stylesheet in the _init_ function of the main class. Next, set window opacity in the starting file "if" instruction and adding size in both parts of our code. You can change the color of the text and background-color of the window and elements with Your favorite rgba, hex or named color. Just do this with changing the line: self. element. set Stylesheet ("background-color: #hex number or rgba ();

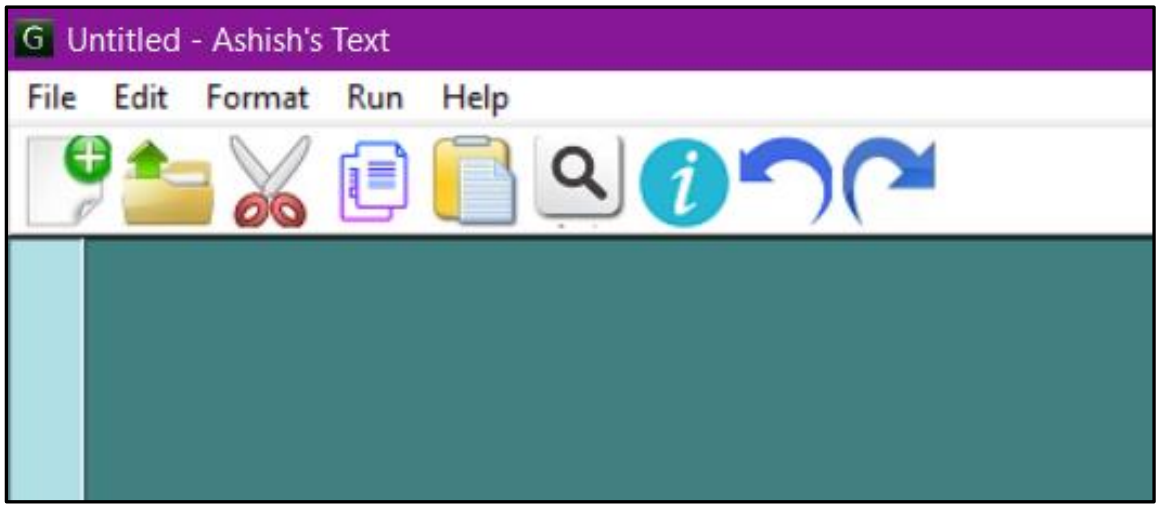
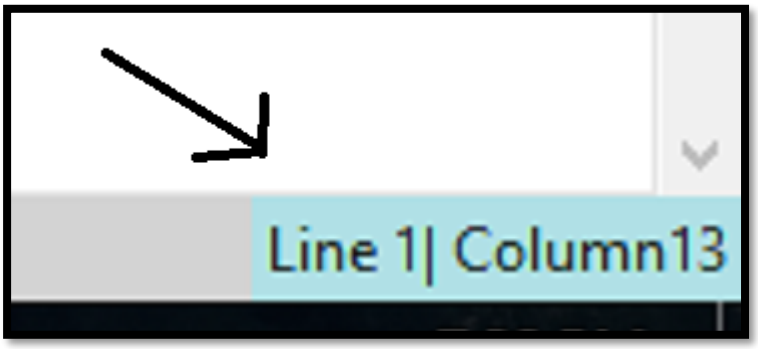


Fig (3.4):Menu icons

3.1.6 Adding line number and column number

In Text Editor, line number always help user to find any specific or for remembering any line. And for Programmers, line numbers are like a golden way to detect any syntax error in their code because during the execution of codes, mostly all programming language raise line number to highlight any syntax error in codes. Hence, Line Numbers and column numbers are very important features for programmers and for others also.



Fig(3.5):Line numbers and column number functions

In any Graphical Application, Scrollbars are always very useful for users and it is also very important Because with this widget we can easily scroll our page or panel up and down with the help of mouse in fastest way. So, In This Tutorials, I am going to show you how to add vertical and horizontal scrollbar in python tkinter Script or you can say how to use python tkinter scrollbar widget.

CHAPTER-4

RESULT AND ANALYSIS

4.1 MENU BAR

A menu bar is a user interface element that contains selectable commands and options for a specific program. In Windows, menu bars are typically located at the top of open windows. While menu bar items vary between applications, most menu bars include the standard File, Edit, and View menus. The File menu includes common file options such as New, Open…, Save, and Print. The Edit menu contains commands such as Undo, Select All, Copy, and Paste. The View menu typically includes zoom commands and options to show or hide elements within the window. Other menu bar items may be specific to the application. For example, a text editor may include a Format menu for formatting selected text and an Insert menu for inserting pictures or other media into a document

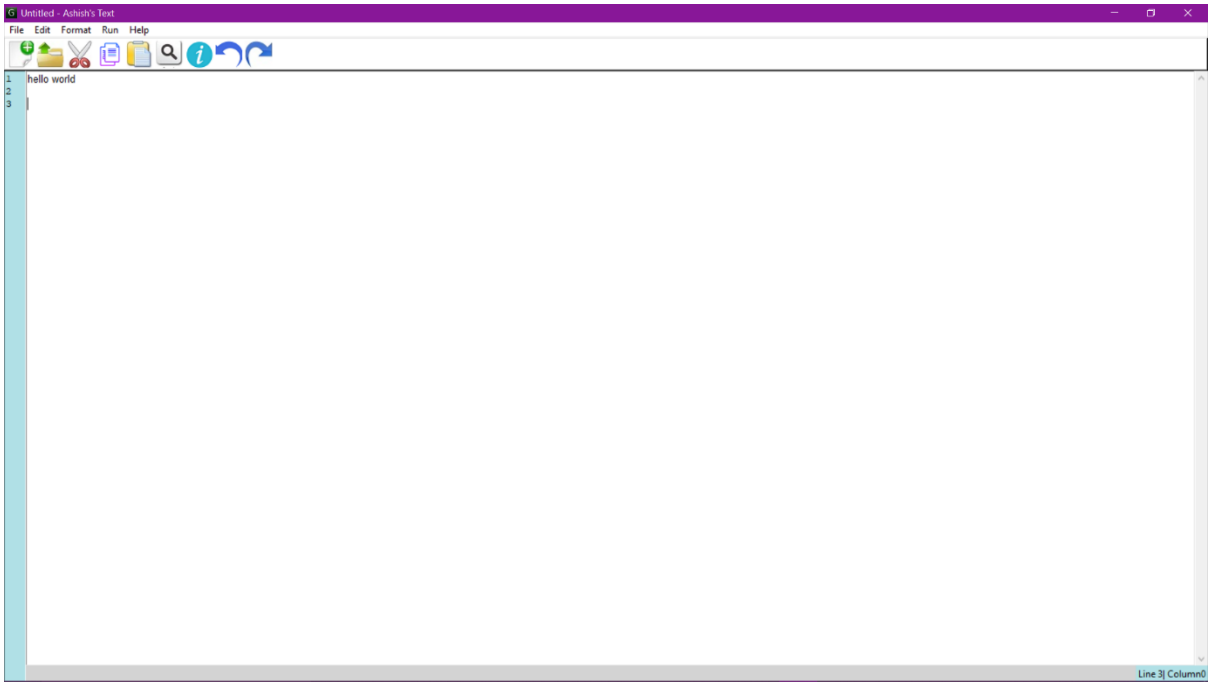


Fig (4.1):Menu bar

4.1.1 File Menu Option.

The File menu is a graphical control element formerly common to most file-handling computer programs, but more recently often replaced by a toolbar or ribbon. It often appears as the first item in the menu bar, and contains commands relating to the handling of files, such as open, save, print, etc. It may also contain a list of recently edited files**.**

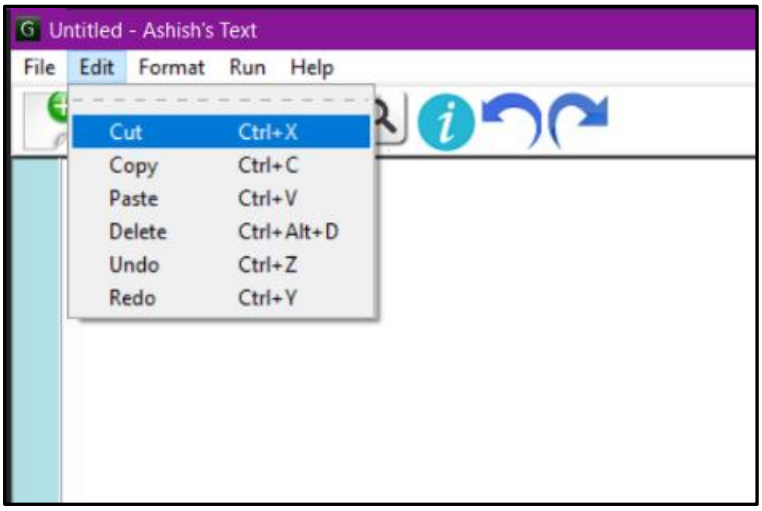
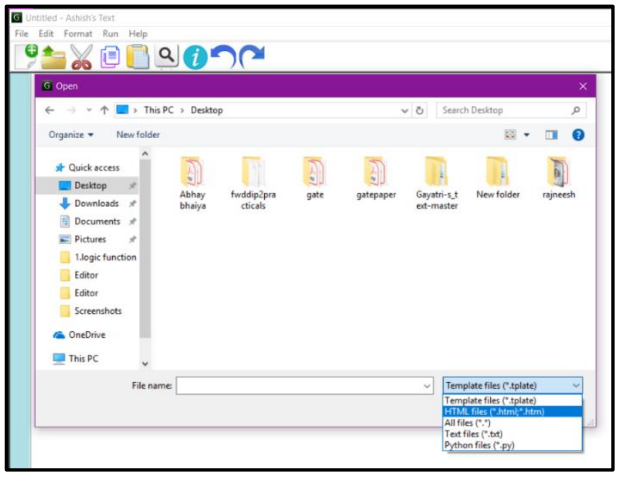
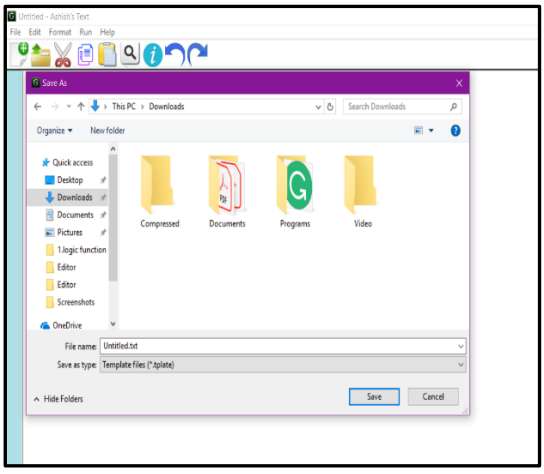


Fig (4.2):File menu



Fig(4.2.1):Open menu



Fig(4.2.2): Save as menu

4.1.2 Edit Menu Options.

The Edit menu is a menu-type graphical control element found in most computer programs that handle files, text or images. It is often the second menu in the menu bar, next to the file menu.

Whereas the file menu commonly contains commands about handling of files, such as open, save, and print, the edit menu commonly contains commands relating to the handling of information within a file, e.g. cut and paste and selection commands. In addition, it may also be home to the undo and redo commands, especially in word processors. It may also contain commands for locating information, e.g. find commands. In graphics-oriented programs, it often contains commands relating to the manipulation of images, for example the crop command.

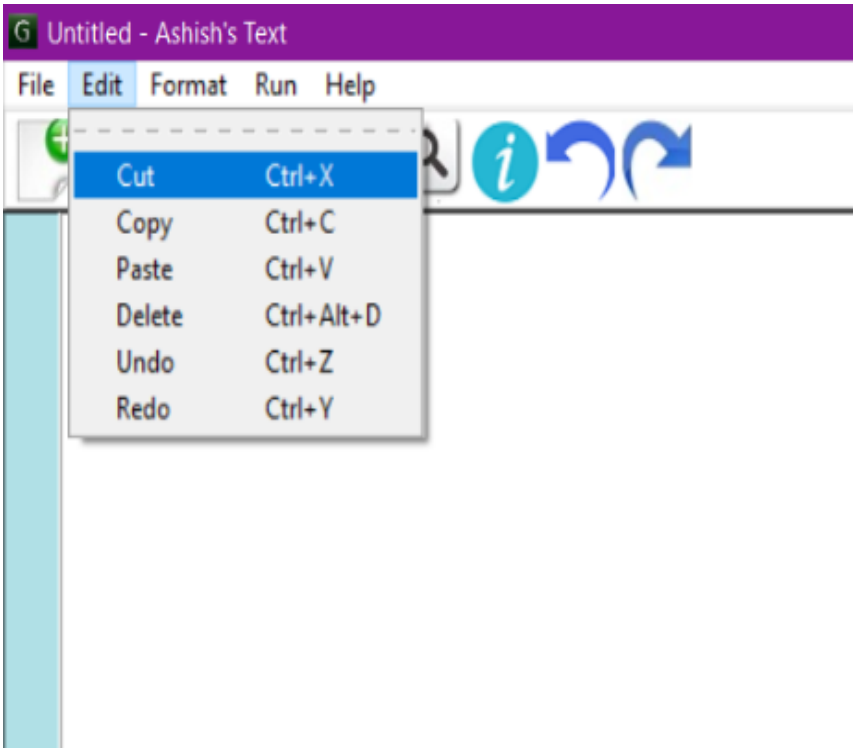
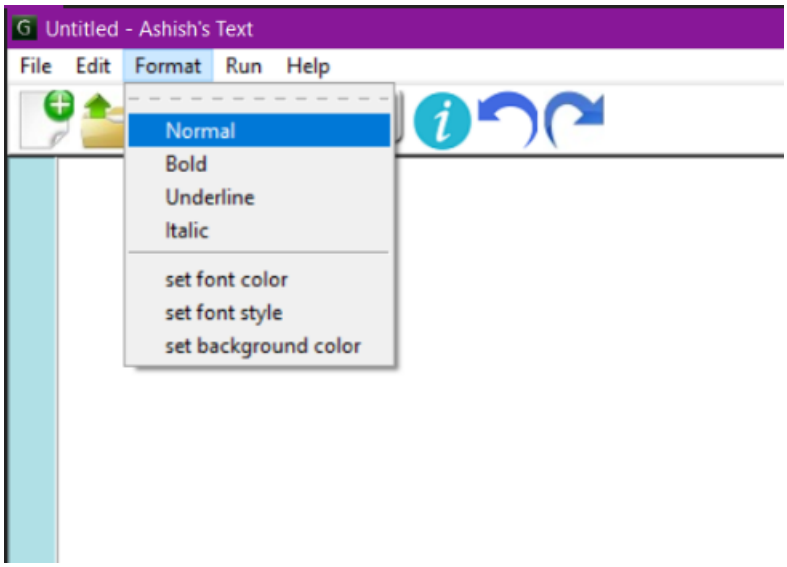


Fig (4.3):Edit menu

4.1.3 Format Menu Options.



Fig(4.4):Format menu

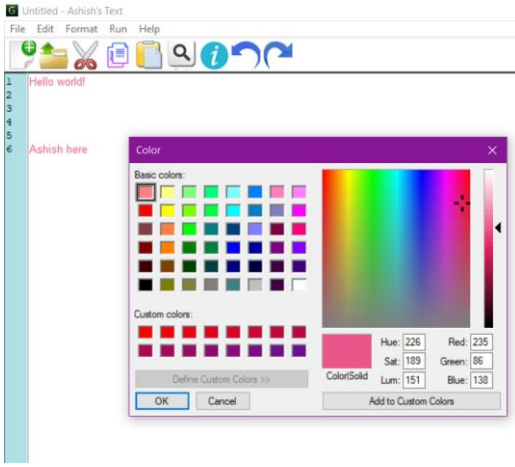


Fig (4.4.1): Font color chooser

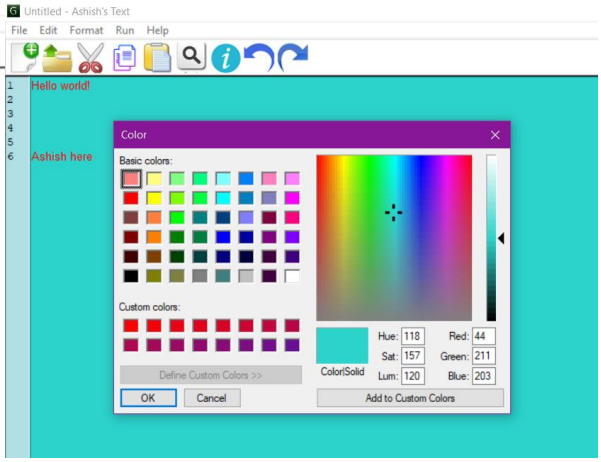


Fig (4.4.2): Background color chooser

4.1.4 Line numbers and scrolling.

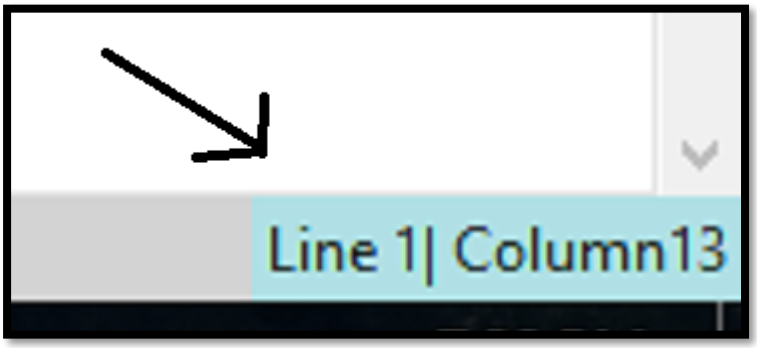


Fig (4.5):Line numbers and scrolling

CONCLUSION

My project (*ASHISH'S TEXT*): In this project we can create a new text document by editing.IN this project we can also create a new text document.in this project we can also view a already presence text document**.**Plain text exclusively consists of character representation. Each character is represented by a fixed-length sequence of one, two, or four bytes, or as a variable-length sequence of one to four bytes, in accordance to specific character encoding conventions.

Some features:

- Create A New Text Document.
- Open A Text Document File.
- Edit A Text Document file.
- View All Text Document.
- Find and replace.
- Cut, copy, and paste.
- Text formatting.
- Undo and redo and many more.
- Font coloring
- Background coloring
- Line numbering column numbering
- Including shortcuts for various icons with their symbols.

By using these functionalities, a user can interact with Text Editor in a much better way, like by using this Text editor Gui.

FUTURE SCOPE

It has become so much a part of technology.This application has a very good future scope and I am working on its Frontend Part as well so that it can be more interactive to every user.The real power of Text editor starts to show up when you want to edit code. Over 50 programming languages are built right in, and choosing one from the Language menu lets you turn on syntax highlighting, document maps, collapsible functions, autocomplete, and much more, depending on the language you choose. You can also define rules for new languages. Text editor does not come with Markdown defined, but helpfully, someone took care of that and released it, and setting that up is a snap.

The core data structure in a text editor is the one that manages the string (sequence of characters) or list of records that represents the current state of the file being edited. While the former could be stored in a single long consecutive array of characters, the desire for text editors that could more quickly insert text, delete text, and undo/redo previous edits led to the development of more complicated sequence data structures.

We can extend its functionalities by adding more features into it like:

- we can implement idea like if user write wrong word then it auto corrects
- we can Check info about the latest version update of Text editor.
- We can add keywords for whatever extension the user is working on.
- Language selection
- Browser running and running with CMD.

APPENDIX

SOURCE CODE(From github.com)

```
import tkinter
from tkinter import *
import sys
import tkinter as ttk
from PIL import Image,ImageTk
from tkinter import filedialog
from tkinter.filedialog import askopenfilename,asksaveasfilename
from tkinter import colorchooser
from tkinter import messagebox
from tkinter.messagebox import *
from tkinter.colorchooser import askcolor
from tkinter.font import *
import tkinter.ttk as ttk
import datetime
import webbrowser
import os
import inspect
import importlib
file = None
def Quit():
    def opn():
        text.delete(1.0, END)
        global file
        file = open(askopenfilename(filetypes=(("Template files", "*.tplate"),
                                               ("HTML files", "*.html;*.htm"),
                                               ("All files", "*.*"),("Text
files","*.txt"),("Python files","*.py") )))
        if file != '':
            x = str(file)
            y = x.split()[1]
            z = y.split("\'")[1]
            v=z.split("\\")[-1]
            t.title(os.path.basename(v)+"- Ashish's text")
            txt = file.read()
            text.insert(INSERT, txt)
            file.close()
        else:
            pass
    def new_file():
```

```
        t.title("Untitled - Ashish's text")
        text.delete(1.0,END)
        file=None
    def save():
        filename =
asksaveasfilename(initialfile='Untitled.txt',defaultextension='.txt',filetypes=(("T
emplate files", "*.tplate"),
                                            ("HTML files", "*.html;*.htm"),
                                            ("All files", "*.*"),("Text
files","*.txt"),("Python files","*.py") ))
        if filename:
            alltext = text.get(1.0, END)
            f=open(filename, 'w')
            f.write(alltext)
            x = str(f)
            # print(x)
            # print(f)
            y = x.split()[1]
            z = y.split("/")[-1]
            # print("value of z=",z)
            # v = z.split("\\")
            # print("val of v=",v)
            # c=v.split("\\")
            # print("Value of c: ",c)
            t.title(z+"- Ashish's text")
    def show():
        ab=Toplevel(t)
        ab.iconbitmap(r'letter-g-icon-png-21704-Windows.ico')
        ab.title("Information")
        l=Label(ab,text="Ashish's text\nauthor and developer-Ashish(the
decider)",fg="cyan",bg="grey",font=("Elephant",20))
        ab.geometry("580x90")
        l.pack()
        # showinfo("Ashish's text","author and developer-Ashish(the decider)")
    def normal():
        text.config(font=("Arial", 20))
    def bold():
        text.config(font=("Arial", 20, "bold"))
    def underline():
        text.config(font=("Arial", 20, "underline"))
    def italic():
        text.config(font=("Arial", 20, "italic"))
    def fontcolor():
        (triple, color) = askcolor()
        if color:
            text.config(foreground=color)
```

```
    def fontstyle():
        font=families("arial","helvetica")
        if font:
            text.config(font="font")
    def background():
        (triple, color) = askcolor()
        if color:
            text.config(background=color)
    def delete():
        MsgBox = messagebox.askquestion('Delete File', 'Are you sure you want to
delete the file?',icon='warning')
        if MsgBox == 'yes':
            x = str(file)
            y = x.split()[1]
            t.destroy()
            y.split("\'")[1]
            os.remove(y.split("\'")[1])
            Quit()
            # os.remove(os.path.curdir)
        else:
            messagebox.showinfo('Return', 'You will now return to the application
screen')
    def shw(event):
        y = ''
        x = text.index("end").split(".")[0]
        for i in range(1, int(x)):
            y += str(i) + '\n'
        text1.config(state='normal')
        text1.delete(1.0, 'end')
        text1.insert(1.0, y)
        text1.config(state='disabled')
        display()
    def display():
        x, y = text.index(INSERT).split('.')
        str = 'Line ' + x + '| Column' + y
        label.config(text=str)
    def find():
        ad=Toplevel(t)
        ad.title("Find Text")
        ad.iconbitmap(r'letter-g-icon-png-21704-Windows.ico')
        ad.geometry("480x90")
        l1=Label(ad,bg="light grey",text="enter the text you want to find.")
        l1.pack()
        e=Entry(ad,bg="light cyan")
        e.pack()
        def search():
```

```
            start = 1.0
            while 1:
                pos = text.search(e.get(), start, stopindex=END)
                if not pos:
                    break
                text.tag_config(e.get())
                print(pos)
                start = pos + "+1c"
        v1 = Button(ad, bd=4, width=5, height=1, text="OK", font=15,
command=search)
        v1.pack()
        mainloop()
    t=Toplevel()                                    #child window including all
functionality
    t.title("Untitled - Ashish's Text")
    screenwidth=t.winfo_screenwidth()
    screenheight=t.winfo_screenheight()
    t.grid_rowconfigure(0,weight=1)
    t.grid_columnconfigure(0,weight=1)
    t.geometry('600x600')
    toolbaar = Frame(t, bd=2, relief=RAISED,bg="white")     #Frame for icons
    img = Image.open(r"new.jpg")                            #icon for new file
    photo1 = ImageTk.PhotoImage(img)
    new = Button(toolbaar, bd=5,bg="white", width=22,height=22,
relief=FLAT,image=photo1,command=new_file)
    new.pack(side=LEFT, padx=2, pady=2)
    img = Image.open(r"open.jpg")                           #icon for open new file
    photo2 = ImageTk.PhotoImage(img)
    open1=Button(toolbaar, bd=5,bg="grey", width=22,height=22,
relief=FLAT,image=photo2,command=opn)
    open1.pack(side=LEFT, padx=2, pady=2)
    img = Image.open(r"cut.jpg")                            #icon for cut
    photo3 = ImageTk.PhotoImage(img)
    cut=Button(toolbaar, bd=5,bg="grey", width=22,height=22,
relief=FLAT,image=photo3,command=lambda :t.focus_get().event_generate("<<Cut>>"))
    cut.pack(side=LEFT, padx=2, pady=2)
    img = Image.open(r"copy.png")                           #icon for copy
    photo4 = ImageTk.PhotoImage(img)
    copy = Button(toolbaar, bd=5, bg="grey", width=22, height=22, relief=FLAT,
image=photo4,command=lambda: t.focus_get().event_generate("<<Copy>>"))
    copy.pack(side=LEFT, padx=2, pady=2)
    img = Image.open(r"paste.jpg")                          #icon for paste
    photo5 = ImageTk.PhotoImage(img)
    paste = Button(toolbaar, bd=5, bg="grey", width=22, height=22, relief=FLAT,
image=photo5,command=lambda: t.focus_get().event_generate("<<Paste>>"))
    paste.pack(side=LEFT, padx=2, pady=2)
```

```
    img = Image.open(r"find.jpg")                          #icon that helps find
position of text item
    photo6 = ImageTk.PhotoImage(img)
    find = Button(toolbaar, bd=5, bg="grey", width=22, height=22, relief=FLAT,
image=photo6,command=find)
    find.pack(side=LEFT, padx=2, pady=2)
    img = Image.open(r"info.png")                          #info
    photo7 = ImageTk.PhotoImage(img)
    info1 = Button(toolbaar, bd=5, bg="grey", width=22, height=22, relief=FLAT,
image=photo7,command=show)
    info1.pack(side=LEFT, padx=2, pady=2)
    img = Image.open(r"undo.png")                          #icon for undo operation
    photo8 = ImageTk.PhotoImage(img)
    undo = Button(toolbaar, bd=5, bg="grey", width=22, height=22, relief=FLAT,
image=photo8,command=lambda:t.focus_get().event_generate("<<Undo>>"))
    undo.pack(side=LEFT, padx=2, pady=2)
    img = Image.open(r"redo.jpg")                          #icon for redo operation
    photo9 = ImageTk.PhotoImage(img)
    redo = Button(toolbaar, bd=5, bg="grey", width=22, height=22, relief=FLAT,
image=photo9,command=lambda:t.focus_get().event_generate("<<Redo>>"))
    redo.pack(side=LEFT, padx=2, pady=2)
    toolbaar.pack(side=TOP, fill=X)
    #code for line number finding
    text1 = Text(t, bg='powderblue', fg='black', state='disabled', width=3)
    text1.pack(side=LEFT, fill=Y)
    label = Label(t, text="Line 0| Column 0", bg='powderblue', fg='black')
    label.pack(side=BOTTOM,anchor=SE)
    text = Text(t,  font=("Arial", 10),height=700)
    scroll = Scrollbar(t, command=text.yview)
    scroll.config(command=text.yview)
    text.config(yscrollcommand=scroll.set,bg="white")
    scroll.pack(side=RIGHT, fill=Y)
    text.pack(side=TOP,fill=BOTH)
    t.iconbitmap(r'letter-g-icon-png-21704-Windows.ico')
    text.bind("<Key>", shw)
    # t.pack(fill=BOTH,expand=1)
    menu=Menu(t)                          #main menu for editor
    t.config(menu=menu,bg="light grey")
    filemenu=Menu(menu)
    menu.add_cascade(label="File", menu=filemenu)
    filemenu.add_command(label="New
",accelerator='Ctrl+N',command=new_file)
    filemenu.add_command(label="Open            ",accelerator='Ctrl+O',command=opn)
    filemenu.add_command(label="Save
",accelerator="Ctrl+S",command=save)
    filemenu.add_command(label="Close"
```

```
,accelerator='Ctrl+W',command=t.quit)
    filemenu.add_separator()
    filemenu.add_command(label="Exit",command=t.quit)
    editmenu=Menu(menu)
    menu.add_cascade(label="Edit",menu=editmenu)
    editmenu.add_command(label='Cut
',accelerator='Ctrl+X',command=lambda :t.focus_get().event_generate("<<Cut>>"))
    editmenu.add_command(label='Copy
',accelerator='Ctrl+C',command=lambda :t.focus_get().event_generate("<<Copy>>"))
    editmenu.add_command(label='Paste
',accelerator='Ctrl+V',command=lambda :t.focus_get().event_generate("<<Paste>>"))
    editmenu.add_command(label='Delete'
,accelerator='Ctrl+Alt+D',command=delete)
    editmenu.add_command(label='Undo'
,accelerator='Ctrl+Z',command=lambda:t.focus_get().event_generate("<<Undo>>"))
    editmenu.add_command(label='Redo'
,accelerator='Ctrl+Y',command=lambda :t.focus_get().event_generate("<<Redo>>"))
    formatmenu=Menu(menu)
    menu.add_cascade(label='Format',menu=formatmenu)
    formatmenu.add_checkbutton(label='Normal',command=normal)
    formatmenu.add_checkbutton(label='Bold',command=bold)
    formatmenu.add_checkbutton(label='Underline',command=underline)
    formatmenu.add_checkbutton(label='Italic',command=italic)
    formatmenu.add_separator()
    formatmenu.add_command(label='set font color',command=fontcolor)
    formatmenu.add_command(label='set font style',command=fontstyle)
    formatmenu.add_command(label='set background color',command=background)
    runmenu=Menu(menu)
    menu.add_cascade(label="Run",menu=runmenu)
    runmenu.add_command(label="Run...           F5")
    runmenu.add_separator()
    runmenu.add_command(label="Launch in firefox            Ctrl+alt+Shift+X")
    runmenu.add_command(label="Launch in Chrome             Ctrl+alt+Shift+R")
    helpmenu=Menu(menu)
    menu.add_cascade(label="Help",menu=helpmenu)
    helpmenu.add_command(label="About Ashish's Text..",command=show)

    root.withdraw()
    mainloop()
root=Tk()                       #root window
style=ttk.Style(root)
root.title("Ashish's Text")    #title of root
root.config(bg="light grey")    #background color
screenwidth=root.winfo_screenwidth()       #auto adjust of width and height
screenheight=root.winfo_screenheight()
root.grid_rowconfigure(0,weight=1)
```

```
root.grid_columnconfigure(0,weight=1)
root.geometry("600x600")
root.iconbitmap(r'letter-g-icon-png-21704-Windows.ico')         #setting icon of
window
# root.resizable("False","False")
im=Image.open(r"gayu.jpg")
photo=ImageTk.PhotoImage(im)
# cv = Canvas()
# cv.config(width=600,height=600)
# cv.place( x=0,y=0,relwidth=1,relheight=1)
# cv.create_image(50, 50, image=photo)
l=Label(image=photo)                        #inserting main image
l.place(x=0,y=0,relwidth=1,relheight=1)
b=Button(l,justify=LEFT,bd=5,width=60,bg="grey",command=Quit)    #button forming
img=Image.open(r"jf.jpg")                                        #inserting image
into button
photo2=ImageTk.PhotoImage(img)
b.config(image=photo2)
# cv.create_window(200,200,anchor="center",window=b)
b.pack()
root.mainloop()
```

REFERENCES

[1] https://en.wikipedia.org/wiki/Python_(programming_language)

[2] https://en.wikipedia.org/wiki/PyCharm

[3] https://searchitoperations.techtarget.com/definition/GitHub

[4] https://github.com/Ashish-soni/Ashish-s_text

[5] https://en.wikipedia.org/wiki/Text_editor

[6] https://en.wikipedia.org/wiki/Graphical_user_interface

[7] https://github.com/Ashish-soni

[8]https://acadview.com/student/projects/objectives/4802