MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

Project Report

on

Notepad using tkinter

Submitted By:

Saurabh Shakya

0901CS191110

Sourabh Mourya

0901CS191124

Faculty Mentor:

Mr. Mir Shahnawaz Ahmad

(Assistant Professor, Computer Science and Engineering)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE

GWALIOR - 474005 (MP) est. 1957

MAY-JUNE 2022

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

Project Report

on

Notepad using tkinter

A project report submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Saurabh Shakya

0901CS191110

Sourabh Mourya

0901CS191124

Faculty Mentor:

Mr.Mir Shahnawaz Ahmad

(Assistant Professor, Computer Science and Engineering)

Submitted to:

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE
GWALIOR - 474005 (MP) est. 1957

MAY-JUNE 2022

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

CERTIFICATE

This is certified that **Saurabh Shakya** (0901CS191110) has submitted the project report titled [illegible] **Notepad using tkinter** under the mentorship of **Mr. Mir Shahnawaz Ahmad.**, in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering from Madhav Institute of Technology and Science, Gwalior.

[illegible] 09/05/22

Mr. Mir shahnawaz Ahmad
Faculty Mentor
Assistant Professor
Computer Science and Engineering

[illegible] 09/05/2022

Dr. Manish Dixit
Professor and Head,
Computer Science and Engineering

Dr. Manish [illegible]
Professor & HOD
Department of CSE
M.I.T.S. Gwalior

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

[illegible]

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

CERTIFICATE

This is certified that **Sourabh Mourya** (0901CS191124) has submitted the project report titled **Notepad using tkinter** under the mentorship of **Mr. Mir Shahnawaz Ahmad.**, in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering from Madhav Institute of Technology and Science, Gwalior

[illegible] 09/05/22

Mr. Mir shahnawaz Ahmad
Faculty Mentor
Assistant Professor
omputer Science and Engineering

[illegible] 09/05/2022

Dr. Manish Dixit
Professor and Head,
Computer Science and Engineering

Dr. Manish Dixit
Professor & HOD
Department of CSE
M.I.T.S. Gwalior

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

DECLARATION

ereby declare that the work being presented in this project report, for the partial fulfilment of requirement for award of the degree of Bachelor of Technology in Computer Science and Engineering at Madhav Institute of Technology & Science, Gwalior is an authenticated and original record of my work under the mentorship of **Mr. r Shahnawaz Ahmad**, **Assistant Professor**, Computer Science Engineering .

declare that I have not submitted the matter embodied in this report for the award of any degree or diploma nywhere else.

[illegible]
Saurabh Shakya
0901CS191110
3rd Year
Computer Science and Engineering

Sourabh Mourya
0901CS191124
3rd Year,
Computer Science and Engineering

[illegible]

MADHAV INSTITUTE OF TECHNOLOGY & SCIENCE, GWALIOR

(A Govt. Aided UGC Autonomous & NAAC Accredited Institute Affiliated to RGPV, Bhopal)

ACKNOWLEDGEMENT

The full semester project has proved to be pivotal to my career. I am thankful to my institute, **Madhav Institute of chnology and Science** to allow me to continue my disciplinary/interdisciplinary project as a curriculum requirement, der the provisions of the Flexible Curriculum Scheme (based on the AICTE Model Curriculum 2018), approved by the Academic Council of the institute. I extend my gratitude to the Director of the institute, **Dr. R. K. Pandit** and Dean ademics, **Dr. Manjaree Pandit** for this.

I would sincerely like to thank my department, **Department of Computer Science and Engineering, for allowing** me to explore this project. I humbly thank **Dr. Manish Dixit**, Professor and Head, Department of Computer Science and gineering, for his continued support during the course of this engagement, which eased the process and formalities involved.

I am sincerely thankful to my faculty mentors. I am grateful to the guidance of **Mr. Mir Shahnawaz Ahmad , ofessor Assistant**, Computer Science and Engineering , for his continued support and guidance throughout the project. I am also very thankful to the faculty and staff of the department.

[illegible]
Saurabh Shakya
0901CS191110
3rd Year
Computer Science and Engineering

Sourabh Mourya
0901CS191124
3rd Year,
Computer Science and Engineering

[illegible]

ABSTRACT

Abstract:

This project is about a text editor similar to the one provided with Windows Operating system's application software "Notepad".

It has some additional features such as an option to choose between dark mode and light mode. It can also check spelling errors for text written in English.

This project is based on Python3. And it make use of a GUI library Tkinter.

सार:

यह प्रोजेक्ट विंडोज ऑपरेटिंग सिस्टम के एप्लिकेशन सॉफ्टवेयर "नोटपैड" के साथ उपलब्ध कराए गए टेक्स्ट एडिटर के समान है।

इसमें कुछ अतिरिक्त विशेषताएं हैं जैसे कि डार्क मोड और लाइट मोड के बीच चयन करने का विकल्प। यह अंग्रेजी में लिखे गए टेक्स्ट के लिए स्पेलिंग एरर को भी चेक कर सकता है।

यह प्रोजेक्ट Python3 पर आधारित है। और यह एक जीयूआई पुस्तकालय का उपयोग करता है टिंकर

TABLE OF CONTENTS

Chaptrer 1:

Introduction

Overview

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

To create a tkinter app:

- Importing the module – tkinter
- Create the main window (container)
- Add any number of widgets to the main window
- Apply the event Trigger on the widgets.

In our Minor project we are making NOTEPAD using TKinter.
It is based on python3 and it's GUI library Tkinter.

In our notepad we are adding following features:

- **Find a word :**

  With the help of this feature we find a word from the content i.e. written on the notepad.

- **Find and Replace a word :**

  After finding a word, if one wants to replace that word with another word they can do it by using this feature.

- **Spell check :**

  We added a spell check feature in our notepad. With the help of this feature we can spell check a word that is written on the notepad.

- **Dark Mode Option :**

  Every application running on Windows 10 supports dark mode. You can name Web browsers such as Chrome, Firefox, and Microsoft Edge as an example.The default white background may hurt your eyes. So we add a darkode in this notepad. This becomes the most useful notepad out there.

  .

These are the features that we added in our notepad.

Chapter 2:

Required Tools description

1. Visual studio code : Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

   Although this application can be created without using any IDE, IDE cuts the production time by a significant amount. Besided, IDE makes the development process more efficient and it also has convenient collaboration options.

Library used

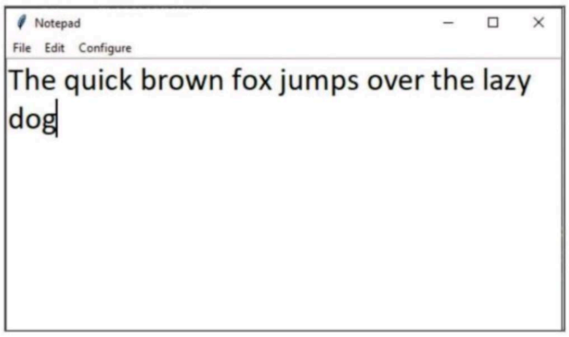
1. Tkinter : Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard GNU/Linux, Microsoft Windows and macOS installs of Python.

Chapter 3:

Features

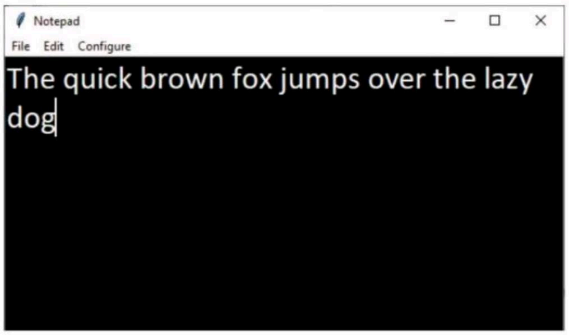
1. **Multiple display modes**
   This application has an option to toggle to and from dark mode to light mode.

   Dark mode there is implemented by specifying background, foreground, color of cursor properties of Tkinter Text object. We specified the background color to 'black' and foreground to 'white'. This implementation can be changed in configurations.



The quick brown fox jumps over the lazy dog

(Light mode)



The quick brown fox jumps over the lazy dog

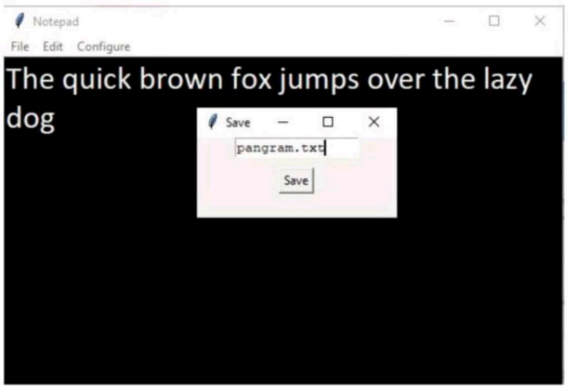
(Dark mode)

2. Saving and opening of a text file

This application has an option to save and open a text file (having extension .txt only).

For saving the file we are using the *open()* function included in python3.

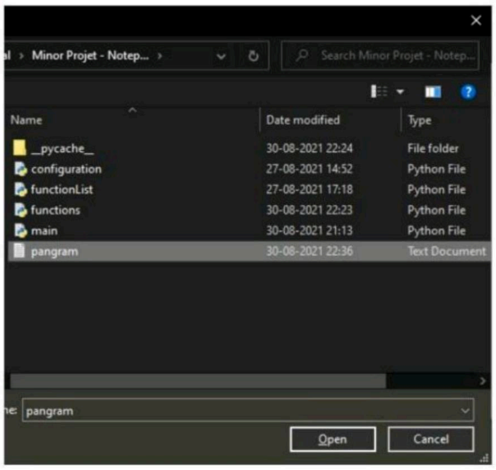
For opening the file we are using the *filedialog* module provided in tkinter.It opens the file chooser prompt. After which with *open()* we can retrieve the stored text and with the *insert()* method of Tkinter text we can display that text.



The quick brown fox jumps over the lazy dog

pangram.txt

(Prompt asking name of file to save)



Name	Date modified	Type
__pycache__	30-08-2021 22:24	File folder
configuration	27-08-2021 14:52	Python File
functionList	27-08-2021 17:18	Python File
functions	30-08-2021 22:23	Python File
main	30-08-2021 21:13	Python File
pangram	30-08-2021 22:36	Text Document

pangram

(File chooser dialog)

3. Find

This application has an option to find the occurrence of a string in the text typed so far.

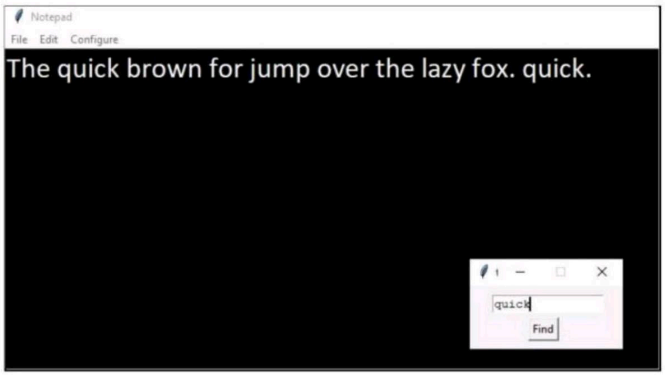
We are using Krap-Robin algorithm for searching the multiple occurrence of the string over text to search from. The main advantage of this algorithm is that, it's time complexity is O(n).
Naive algorithm for the same problem will be of O(n.w) where w is the length of string to search.

<u>Description of Krap-Robin algorithm:</u>
Rabin–Karp algorithm is a string-searching algorithm created by Richard M. Karp and Michael O. Rabin (1987) that uses hashing to find an exact match of a pattern string in a text. It uses a rolling hash to quickly filter out positions of the text that cannot match the pattern, and then checks for a match at the remaining positions.

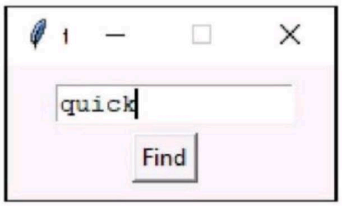
The algorithm is as shown:

```
function RabinKarp(string s[1..n], string pattern[1..m])
   hpattern := hash(pattern[1..m]);
   for i from 1 to n-m+1
       hs := hash(s[i..i+m-1])
       if hs = hpattern
           if s[i..i+m-1] = pattern[1..m]
               return i
   return not found
```

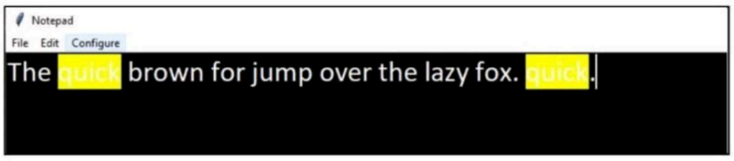
Find feature in action:



The quick brown for jump over the lazy fox. quick.



(Window prompt asking string to search)

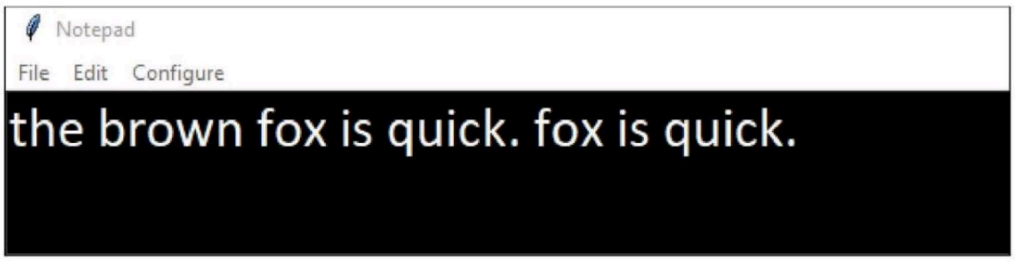


The quick brown for jump over the lazy fox. quick.

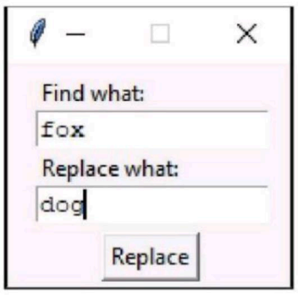
(Searched text is highlighted)

4. Replace

This application has an option to substitute strings. Find and replace both use Krap-Robin algorithm to locate the occurrence of string. Then, those strings are replaced. We use Tkinter text's insert and delete methods for this.



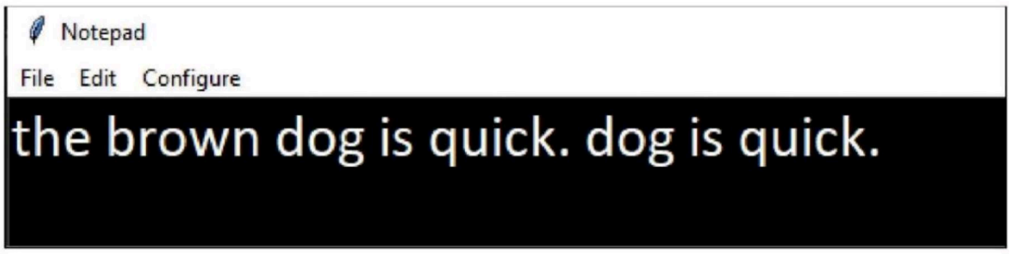
the brown fox is quick. fox is quick.



fox

dog

(Prompt asking which substitute with what)



the brown dog is quick. dog is quick.

(Here 'fox' is replaced with 'dog')

5. Spell check

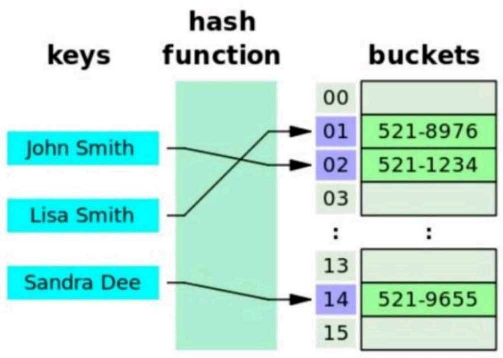
This application has an option to check the correctness of spellings of words in the text. This feature is accessed by clicking the sub menu named ‘Spell check’ under edit options. It will highlight words with incorrect spellings. It can only check for English language.

Implementation

We downloaded a text file that has the most words in the English language (about 466K). Since these are so many words, an efficient algorithm should be used, or else this feature may take a lot of time. Linear search takes *O(n)* time complexity, binary search takes *O(log(n))*. We can do better by creating a hashtable of words in the file during run time. Hashtable takes *O(1)* in average and best case scenarios.

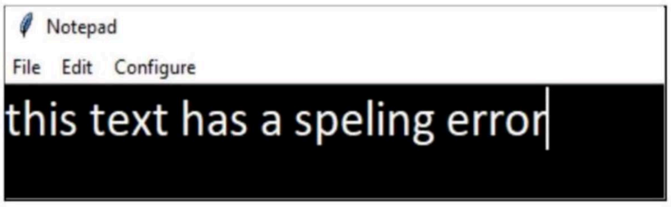
Hashtable

Hash table (hash map) is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index, also called a hash code, into an array of buckets or slots, from which the desired value can be found. During lookup, the key is hashed and the resulting hash indicates where the corresponding value is stored.



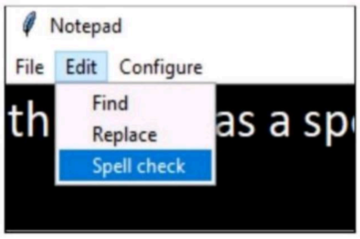
set() data type in python3 uses a hash table. We insert all words in the text file to a *set()* during run time that can be used later.

Spell check feature in action

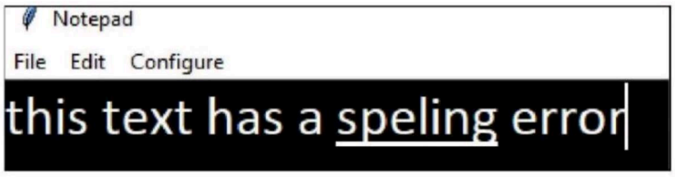


this text has a speling error

(Text here has a spelling error)



(Option to perform spell check)



this text has a speling error

(Wrong spelling is highlighted by underlining)

Chapter 4:

Project Requirements

To run this application:

This application is made keeping desktop users in the mind. That’s why the Tkinter library is used, which is generally supported by desktops.

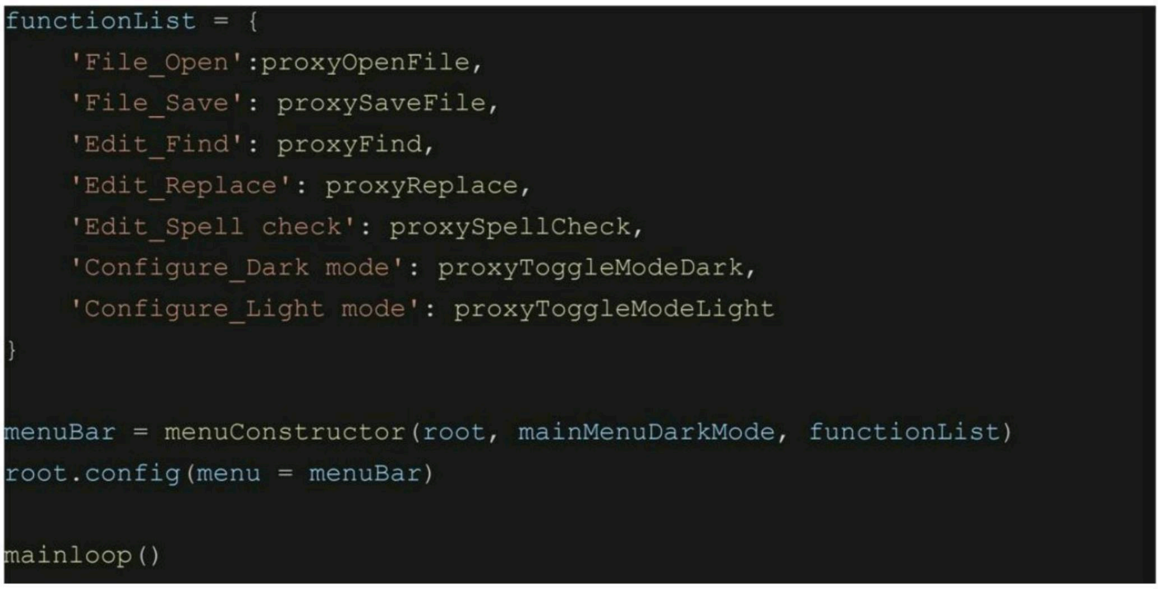
That’s why the minimum requirement to run this application will be Windows, MacOS, Linux, or an operating system that supports python and Tkinter.

This application doesn't hog memory. Which means all devices having the above mentioned operating system will be able to run this application smoothly.

Source code

File: *main.py*

```
from tkinter import *
from configuration import *
from functions import full_spell_check, findWindow, load_words,
menuConstructor, openTextFile, replaceWindow, saveFileWindow,
textBoxConstruction, toggleModeDark, toggleModeLight

def proxyToggleModeLight():
    root.config(menu = menuConstructor(root, mainMenuDarkMode,
functionList))
    toggleModeLight(textArea)

def proxyToggleModeDark():
    root.config(menu = menuConstructor(root, mainMenuLightMode,
functionList))
    toggleModeDark(textArea)

def proxySaveFile():
    saveFileWindow(root, textArea)

def proxyOpenFile():
    openTextFile(textArea)

def proxyFind():
    findWindow(root, textArea)

def proxyReplace():
    replaceWindow(root, textArea)

def proxySpellCheck():
    full_spell_check(textArea)

root = Tk()
root.title(APP_NAME)
textArea = textBoxConstruction(root, CURRENT_MODE);
textArea.pack()
load_words()
```



```
functionList = {
    'File_Open':proxyOpenFile,
    'File_Save': proxySaveFile,
    'Edit_Find': proxyFind,
    'Edit_Replace': proxyReplace,
    'Edit_Spell check': proxySpellCheck,
    'Configure_Dark mode': proxyToggleModeDark,
    'Configure_Light mode': proxyToggleModeLight
}

menuBar = menuConstructor(root, mainMenuDarkMode, functionList)
root.config(menu = menuBar)

mainloop()
```

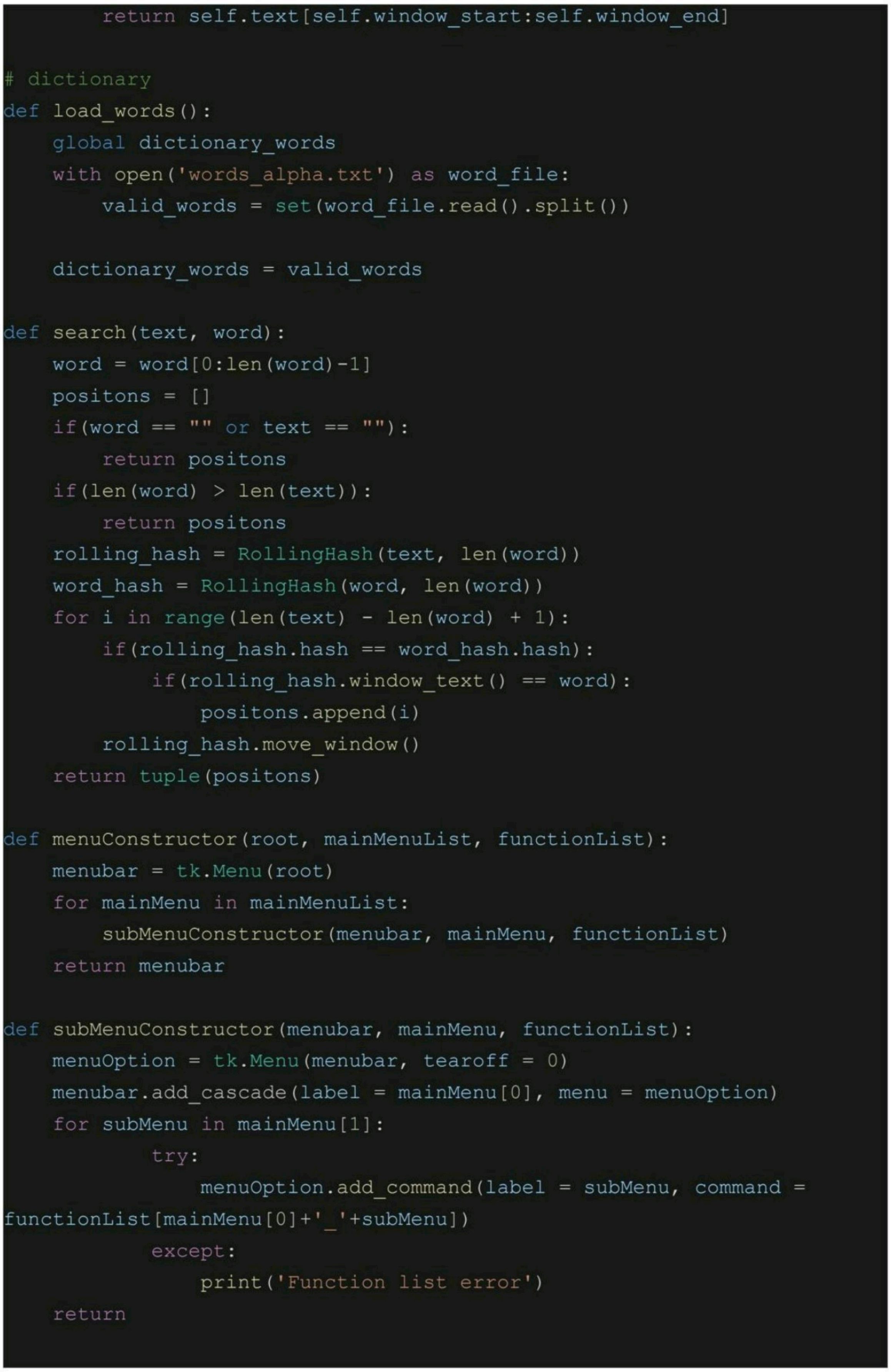
Source code

File: *configuration.py*

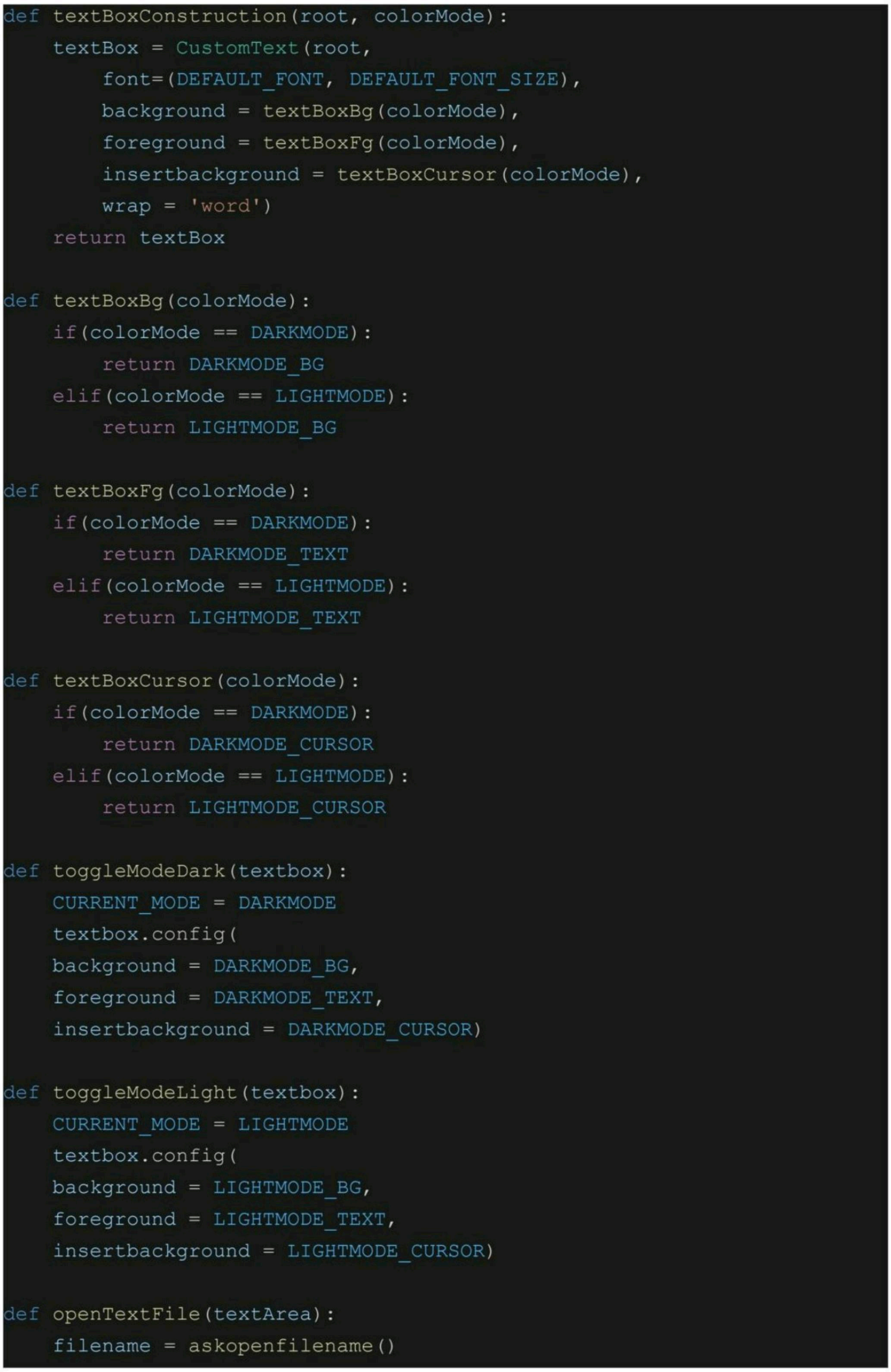
```
# configuration
# constants
DARKMODE = 0
LIGHTMODE = 1
APP_NAME = 'Notepad'

DARKMODE_BG = 'black'
DARKMODE_TEXT = 'white'
DARKMODE_CURSOR = 'white'

LIGHTMODE_BG = 'white'
LIGHTMODE_TEXT = 'black'
LIGHTMODE_CURSOR = 'black'

DEFAULT_FONT = 'Calibri'
DEFAULT_FONT_SIZE = 24

# default settings
CURRENT_MODE = DARKMODE

falg_isTextHighlighted = False
words_ = None

# menu bar
mainMenuBase = [['File', ['Open', 'Save']], ['Edit', ['Find',
'Replace', 'Spell check']]]
mainMenuDarkMode = mainMenuBase + [['Configure', ['Dark mode']]]
mainMenuLightMode = mainMenuBase + [['Configure', ['Light mode']]]

# functionList name is Menu+'_'+SubMenu, ensure no collision
```

<u>Source code</u>
File: *functions.py*

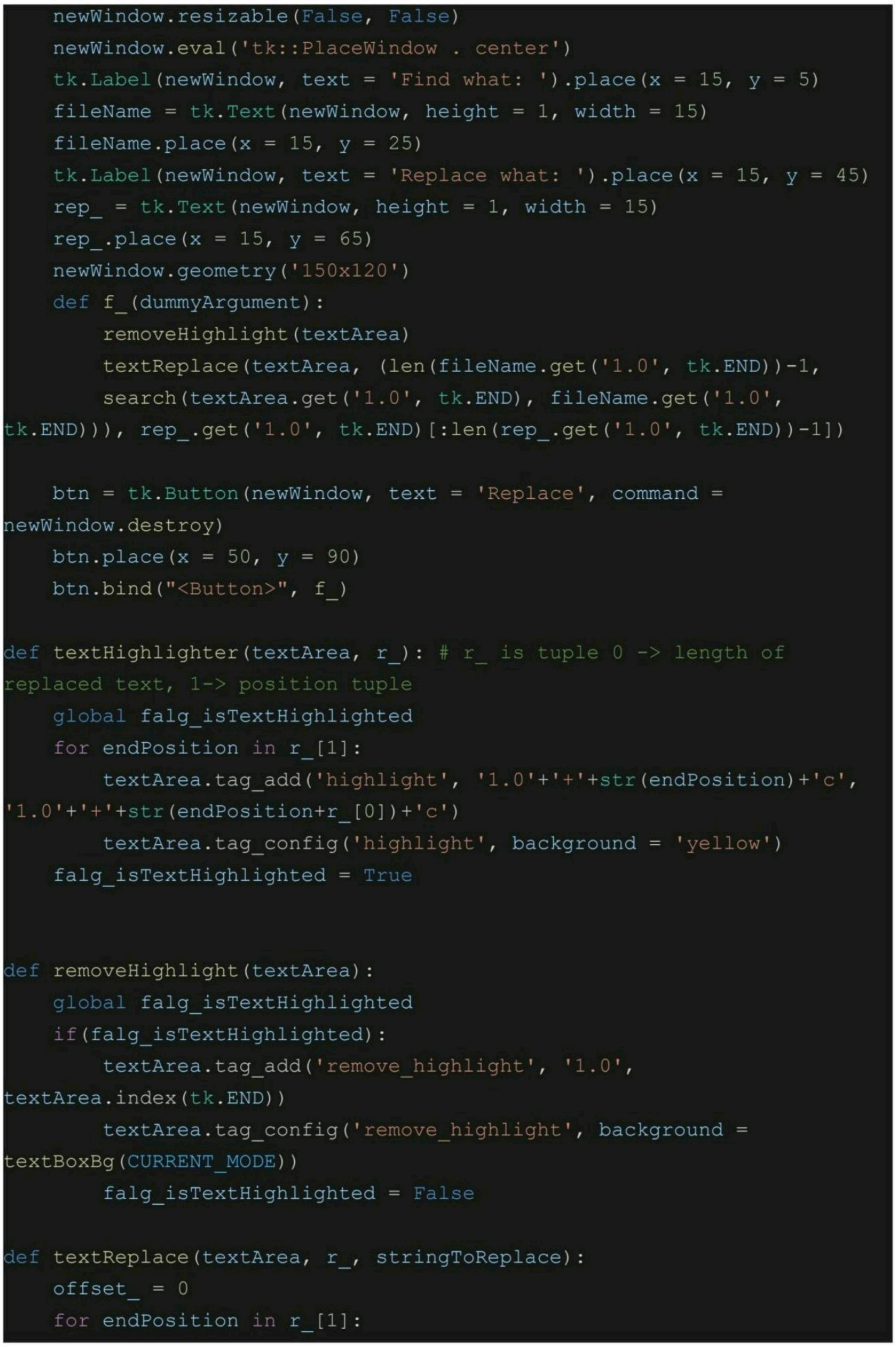
```
import tkinter as tk
from tkinter.filedialog import askopenfilename
from configuration import *

dictionary_words = {}

class CustomText(tk.Text):
    def __init__(self, *args, **kwargs):
        tk.Text.__init__(self, *args, **kwargs)
        self._orig = self._w + "_orig"
        self.tk.call("rename", self._w, self._orig)
        self.tk.createcommand(self._w, self._proxy)
    def _proxy(self, *args):
        cmd = (self._orig,) + args
        result = self.tk.call(cmd)
        if (args[0] in ("insert", "delete") or
            args[0:3] == ("mark", "set", "insert")):
            self.event_generate("<<CursorChange>>", when="tail")
        return result

class RollingHash:
    def __init__(self, text, sizeWord):
        self.text = text
        self.hash = 0
        self.sizeWord = sizeWord
        for i in range(0, sizeWord):
            self.hash = self.hash + (ord(self.text[i]) -
ord('a')+1)*(26**(sizeWord-i-1))
        self.window_start = 0
        self.window_end = sizeWord
    def move_window(self):
        if(self.window_end <= len(self.text) - 1):
            self.hash = self.hash - (ord(self.text[self.window_start])
- ord("a")+1)*26**(self.sizeWord-1)
            self.hash = self.hash * 26
            self.hash = self.hash + ord(self.text[self.window_end])-
ord("a")+1
            self.window_start = self.window_start + 1
            self.window_end = self.window_end + 1
    def window_text(self):
```



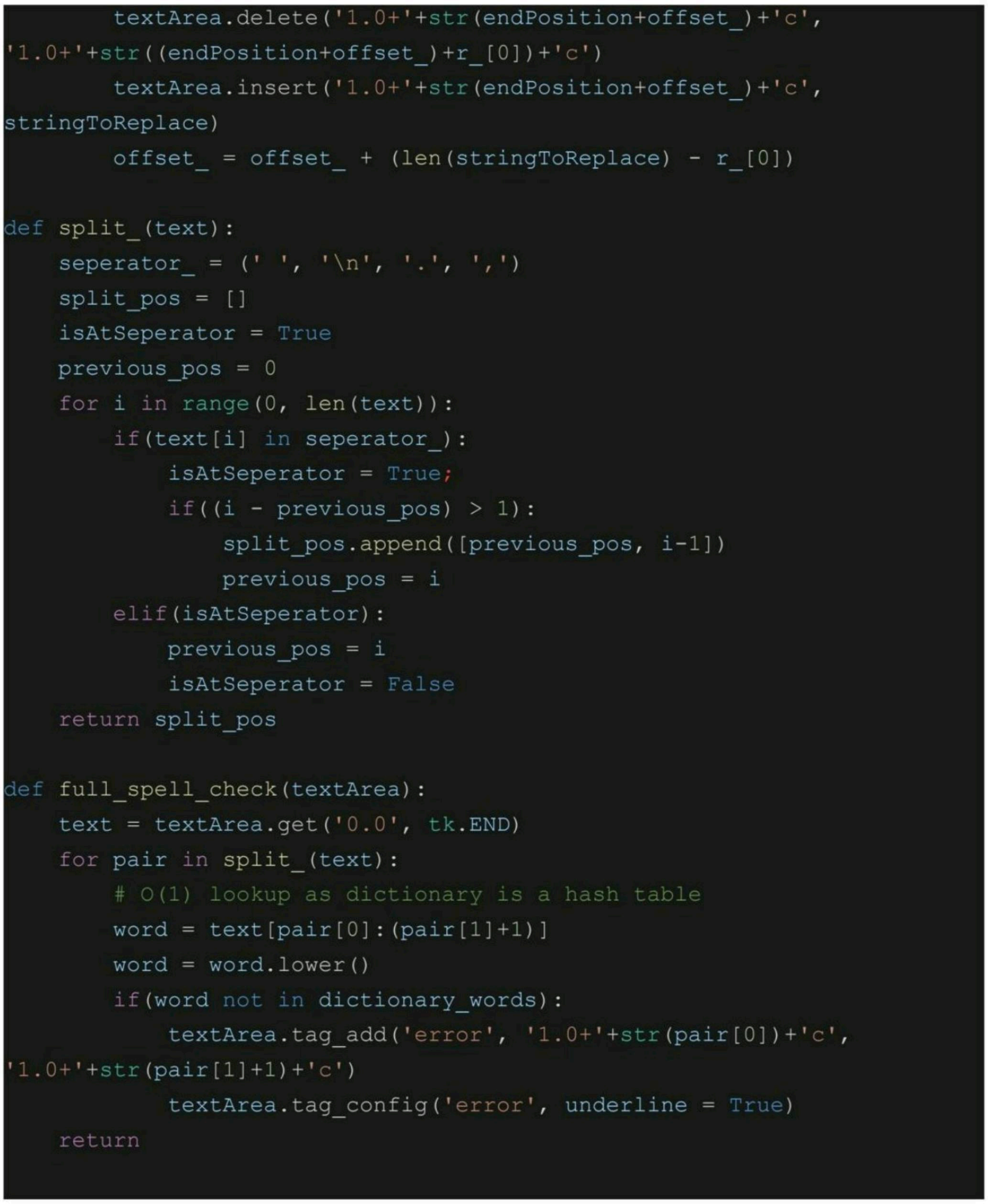
```python
        return self.text[self.window_start:self.window_end]

# dictionary
def load_words():
    global dictionary_words
    with open('words_alpha.txt') as word_file:
        valid_words = set(word_file.read().split())

    dictionary_words = valid_words

def search(text, word):
    word = word[0:len(word)-1]
    positons = []
    if(word == "" or text == ""):
        return positons
    if(len(word) > len(text)):
        return positons
    rolling_hash = RollingHash(text, len(word))
    word_hash = RollingHash(word, len(word))
    for i in range(len(text) - len(word) + 1):
        if(rolling_hash.hash == word_hash.hash):
            if(rolling_hash.window_text() == word):
                positons.append(i)
        rolling_hash.move_window()
    return tuple(positons)

def menuConstructor(root, mainMenuList, functionList):
    menubar = tk.Menu(root)
    for mainMenu in mainMenuList:
        subMenuConstructor(menubar, mainMenu, functionList)
    return menubar

def subMenuConstructor(menubar, mainMenu, functionList):
    menuOption = tk.Menu(menubar, tearoff = 0)
    menubar.add_cascade(label = mainMenu[0], menu = menuOption)
    for subMenu in mainMenu[1]:
            try:
                menuOption.add_command(label = subMenu, command = 
functionList[mainMenu[0]+'_'+subMenu])
            except:
                print('Function list error')
    return
```



```python
def textBoxConstruction(root, colorMode):
    textBox = CustomText(root,
        font=(DEFAULT_FONT, DEFAULT_FONT_SIZE),
        background = textBoxBg(colorMode),
        foreground = textBoxFg(colorMode),
        insertbackground = textBoxCursor(colorMode),
        wrap = 'word')
    return textBox

def textBoxBg(colorMode):
    if(colorMode == DARKMODE):
        return DARKMODE_BG
    elif(colorMode == LIGHTMODE):
        return LIGHTMODE_BG

def textBoxFg(colorMode):
    if(colorMode == DARKMODE):
        return DARKMODE_TEXT
    elif(colorMode == LIGHTMODE):
        return LIGHTMODE_TEXT

def textBoxCursor(colorMode):
    if(colorMode == DARKMODE):
        return DARKMODE_CURSOR
    elif(colorMode == LIGHTMODE):
        return LIGHTMODE_CURSOR

def toggleModeDark(textbox):
    CURRENT_MODE = DARKMODE
    textbox.config(
    background = DARKMODE_BG,
    foreground = DARKMODE_TEXT,
    insertbackground = DARKMODE_CURSOR)

def toggleModeLight(textbox):
    CURRENT_MODE = LIGHTMODE
    textbox.config(
    background = LIGHTMODE_BG,
    foreground = LIGHTMODE_TEXT,
    insertbackground = LIGHTMODE_CURSOR)

def openTextFile(textArea):
    filename = askopenfilename()
    with open(filename, 'r') as line:
        for l in line.readlines():
            textArea.insert(tk.END, l)

def saveFileWindow(root, textArea):
    newWindow = tk.Tk()
    newWindow.resizable(False, False)
    newWindow.eval('tk::PlaceWindow . center')
    fileName = tk.Text(newWindow, height = 1, width = 15)
    fileName.place(x = 25, y = 10)
    newWindow.geometry('170x70')
    def saveFile(event_):
        if(fileName.get('1.0', tk.END)[:-1] == ''):
            return # bad call
        file = open(fileName.get("1.0", tk.END)[:-1], "w+")
        file.write(textArea.get("1.0", tk.END))
        file.close()

    btn = tk.Button(newWindow, text = 'Save', command =
newWindow.destroy)
    btn.place(x = 65, y = 35)
    btn.bind("<Button>", saveFile)

def findWindow(root, textArea):
    newWindow = tk.Tk()
    newWindow.resizable(False, False)
    newWindow.eval('tk::PlaceWindow . center')
    fileName = tk.Text(newWindow, height = 1, width = 15)
    fileName.place(x = 25, y = 10)
    newWindow.geometry('170x70')
    def f_(event_):
        textHighlighter(textArea, (len(fileName.get('1.0', tk.END))-1,
        search(textArea.get('1.0', tk.END), fileName.get('1.0',
tk.END))
        ))
        newWindow.destroy
    btn = tk.Button(newWindow, text = 'Find', command =
newWindow.destroy)
    btn.place(x = 65, y = 35)
    btn.bind("<Button>", f_)

def replaceWindow(root, textArea):
    newWindow = tk.Tk()
```



```
    newWindow.resizable(False, False)
    newWindow.eval('tk::PlaceWindow . center')
    tk.Label(newWindow, text = 'Find what: ').place(x = 15, y = 5)
    fileName = tk.Text(newWindow, height = 1, width = 15)
    fileName.place(x = 15, y = 25)
    tk.Label(newWindow, text = 'Replace what: ').place(x = 15, y = 45)
    rep_ = tk.Text(newWindow, height = 1, width = 15)
    rep_.place(x = 15, y = 65)
    newWindow.geometry('150x120')
    def f_(dummyArgument):
        removeHighlight(textArea)
        textReplace(textArea, (len(fileName.get('1.0', tk.END))-1,
        search(textArea.get('1.0', tk.END), fileName.get('1.0',
tk.END))), rep_.get('1.0', tk.END)[:len(rep_.get('1.0', tk.END))-1])

    btn = tk.Button(newWindow, text = 'Replace', command =
newWindow.destroy)
    btn.place(x = 50, y = 90)
    btn.bind("<Button>", f_)

def textHighlighter(textArea, r_): # r_ is tuple 0 -> length of
replaced text, 1-> position tuple
    global falg_isTextHighlighted
    for endPosition in r_[1]:
        textArea.tag_add('highlight', '1.0'+'+'+str(endPosition)+'c',
'1.0'+'+'+str(endPosition+r_[0])+'c')
        textArea.tag_config('highlight', background = 'yellow')
    falg_isTextHighlighted = True


def removeHighlight(textArea):
    global falg_isTextHighlighted
    if(falg_isTextHighlighted):
        textArea.tag_add('remove_highlight', '1.0',
textArea.index(tk.END))
        textArea.tag_config('remove_highlight', background =
textBoxBg(CURRENT_MODE))
        falg_isTextHighlighted = False

def textReplace(textArea, r_, stringToReplace):
    offset_ = 0
    for endPosition in r_[1]:
```



```
        textArea.delete('1.0+'+str(endPosition+offset_)+'c',
'1.0+'+str((endPosition+offset_)+r_[0])+'c')
        textArea.insert('1.0+'+str(endPosition+offset_)+'c',
stringToReplace)
        offset_ = offset_ + (len(stringToReplace) - r_[0])

def split_(text):
    seperator_ = (' ', '\n', '.', ',')
    split_pos = []
    isAtSeperator = True
    previous_pos = 0
    for i in range(0, len(text)):
        if(text[i] in seperator_):
            isAtSeperator = True;
            if((i - previous_pos) > 1):
                split_pos.append([previous_pos, i-1])
                previous_pos = i
        elif(isAtSeperator):
            previous_pos = i
            isAtSeperator = False
    return split_pos

def full_spell_check(textArea):
    text = textArea.get('0.0', tk.END)
    for pair in split_(text):
        # O(1) lookup as dictionary is a hash table
        word = text[pair[0]:(pair[1]+1)]
        word = word.lower()
        if(word not in dictionary_words):
            textArea.tag_add('error', '1.0+'+str(pair[0])+'c',
'1.0+'+str(pair[1]+1)+'c')
            textArea.tag_config('error', underline = True)
    return
```

CONCLUSION:

We have successfully developed a text editor / notepad using python. Along with a simple notepad we have developed file, edit, and help menu. In this python project, we have used tkinter and basic python concepy.

References

1. GitHub - dwyl/english-words: A text file containing 479k English words for all your dictionary/word-based projects e.g: auto-completion / autosuggestion (https://github.com/dwyl/english-words)

2. Rolling hash - Wikipedia (https://en.wikipedia.org/wiki/Rolling_hash)

3. Rabin–Karp algorithm - Wikipedia (https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_algorithm)