SYSTEMS PROGRAMMING PROJECT

TEXT EDITOR

BY

Gaurav Mokhasi, 11CO37

Jay Priyadarshi, 11CO43

Kadam Aditya, 11CO45

Karan Sabhani, 11CO46

SUBMITTED TO

Mr. Deepak Prabhu

OBJECTIVE AND SCOPE

The objective of our project is to create and develop a text editor. We have used various tools and programming applications in the development of this text editor. Our primary scope is to develop a tool which can perform most of the basic functions of any other text editor and some other advanced features.

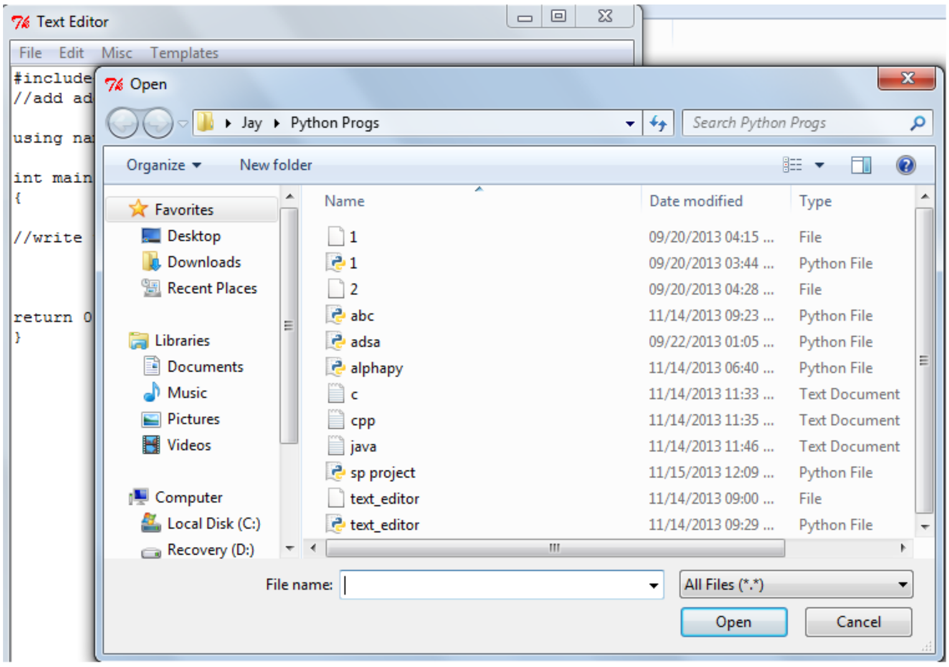
WHAT IS A TEXT EDITOR

Text editors generally are tools provided along with Operating Systems to complete basic text editing functions, change configuration files and basic programming languages. Some text editors are small and simple, while others offer broad and complex functions. For example, UNIX and Unix-like operating systems have the VI editor (or a variant), but many also include the Emacs editor. Microsoft Windows systems come with the simple Notepad, though many people—especially programmers—prefer another Windows text editor with more features. Under Apple Macintosh's classic Mac OS there was the native SimpleText, which was replaced under Mac OS X by TextEdit, which merges features of a text editor with those of a word processor such as rulers, margins and multiple font selection. Some editors, such as WordStar, have dual operating modes allowing them to be either a text editor or a word processor. A text editor written or customized for a specific use can sense what the user is editing and assist the user, often by providing simple ways to retrieve related information. Many text editors for software developers include source code syntax highlighting and automatic completion to make programs easier to read and write. Programming editors often let the user select the name of a subprogram or variable, and then jump to its definition and back.

SALIENT FEATURES AND DESIGN OF OUR TEXT EDITOR

- **Open**

This is used to open a saved file in the file-system.



```
#include
//add ad

using na

int main
{

//write



return 0
}
```

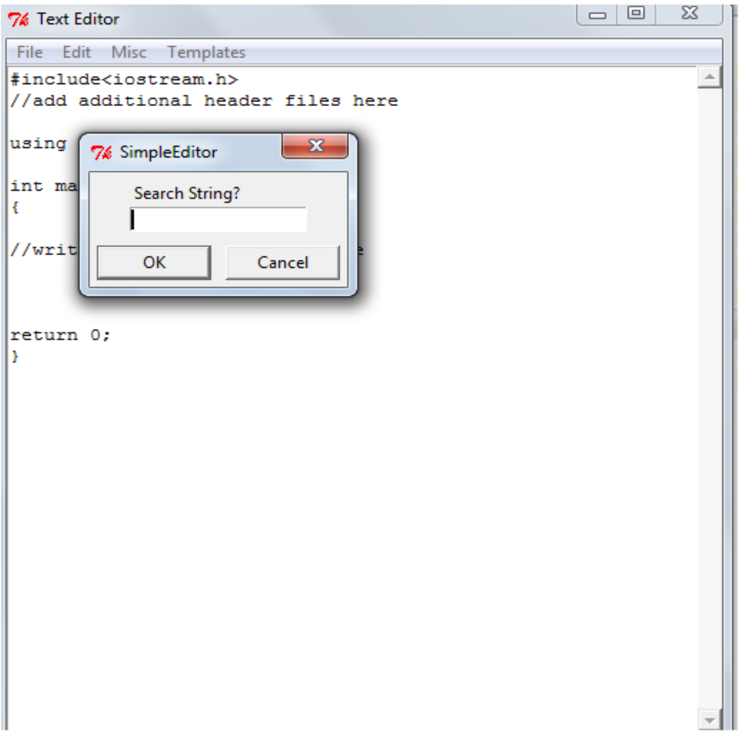
Name	Date modified	Type
1	09/20/2013 04:15 ...	File
1	09/20/2013 03:44 ...	Python File
2	09/20/2013 04:28 ...	File
abc	11/14/2013 09:23 ...	Python File
adsa	09/22/2013 01:05 ...	Python File
alphapy	11/14/2013 06:40 ...	Python File
c	11/14/2013 11:33 ...	Text Document
cpp	11/14/2013 11:35 ...	Text Document
java	11/14/2013 11:46 ...	Text Document
sp project	11/15/2013 12:09 ...	Python File
text_editor	11/14/2013 09:00 ...	File
text_editor	11/14/2013 09:29 ...	Python File

- **Save as**

This feature can be used to save any open file.

- **Find**

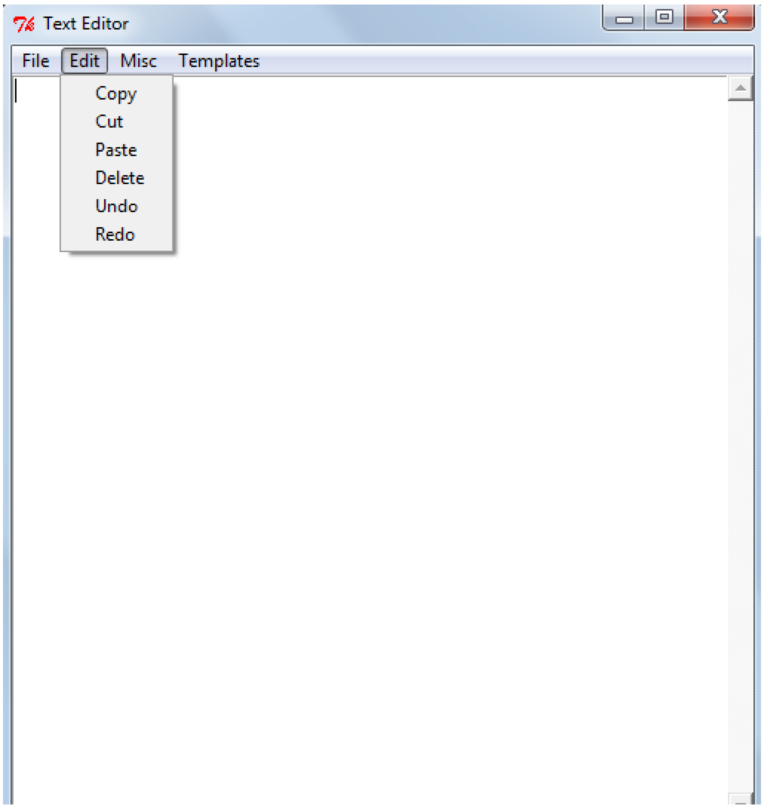
This feature is used to find any string from the text currently in the text box.



```
#include<iostream.h>
//add additional header files here

using

int ma
{

//writ


return 0;
}
```

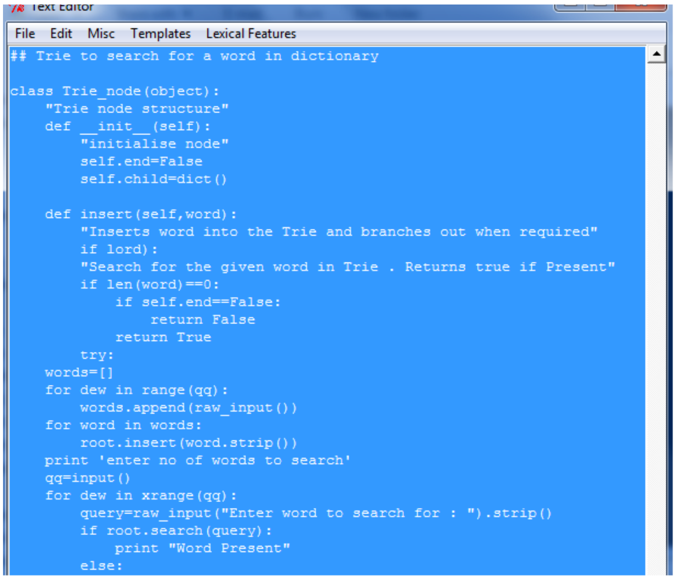
- **Copy, Cut, Paste**

These features can be used to copy or cut any text on the screen. The paste features pastes any copied text onto the screen. There are screen shortcuts for all these three functions.



- **Select All**

This function selects all the text in the text box.



```
## Trie to search for a word in dictionary

class Trie_node(object):
    "Trie node structure"
    def __init__(self):
        "initialise node"
        self.end=False
        self.child=dict()

    def insert(self,word):
        "Inserts word into the Trie and branches out when required"
        if lord):
        "Search for the given word in Trie . Returns true if Present"
        if len(word)==0:
            if self.end==False:
                return False
            return True
        try:
    words=[]
    for dew in range(qq):
        words.append(raw_input())
    for word in words:
        root.insert(word.strip())
    print 'enter no of words to search'
    qq=input()
    for dew in xrange(qq):
        query=raw_input("Enter word to search for : ").strip()
        if root.search(query):
            print "Word Present"
        else:
```

- **Delete**

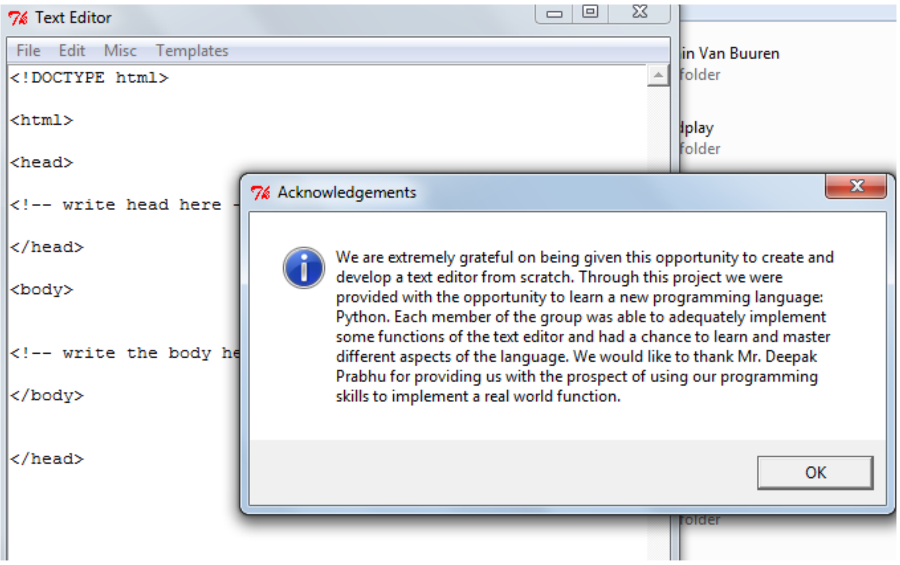
The delete function deletes all selected text from the text box.

- **Undo, Redo**

The undo and redo functions rollback to the previous made changes. There is a screen shortcut for undo.

- **Team Members, Acknowledgements, References**

These functions are all in the miscellaneous tab in the menu bar. The team members tab shows the team members who worked on the project. The acknowledgement tab opens up a dialog box showing the acknowledgements. The references tab shows all the references.

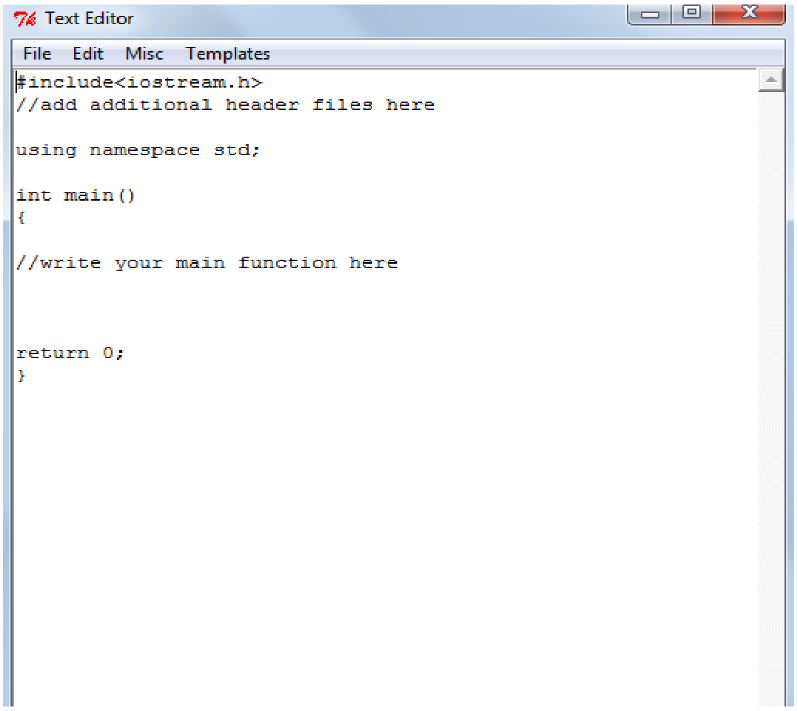


```
<!DOCTYPE html>

<html>

<head>

<!-- write head here

</head>

<body>


<!-- write the body he

</body>


</head>
```

We are extremely grateful on being given this opportunity to create and develop a text editor from scratch. Through this project we were provided with the opportunity to learn a new programming language: Python. Each member of the group was able to adequately implement some functions of the text editor and had a chance to learn and master different aspects of the language. We would like to thank Mr. Deepak Prabhu for providing us with the prospect of using our programming skills to implement a real world function.

- **Templates**

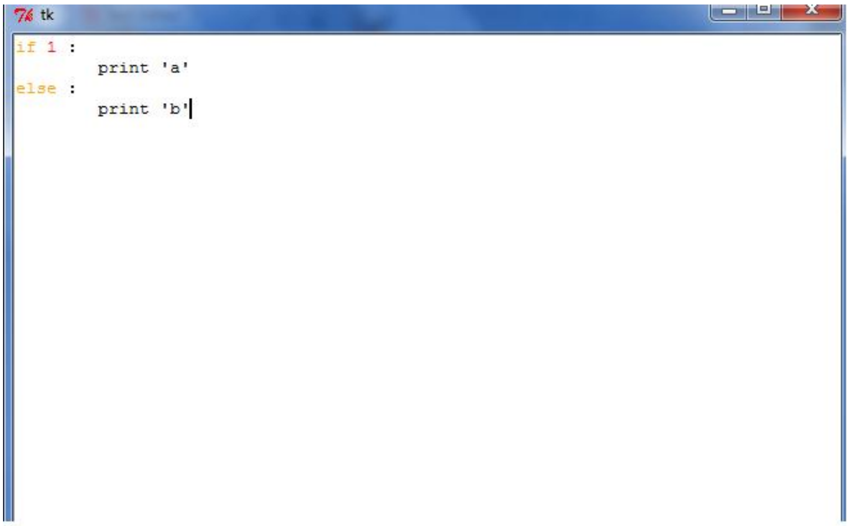
The templates tab contains preloaded templates for four languages C, C++, JAVA, and HTML. Selecting any one of the templates opens up the text box containing a preloaded template for the selected language.



```
#include<iostream.h>
//add additional header files here

using namespace std;

int main()
{

//write your main function here



return 0;
}
```

- **Python Syntax Highlighter**

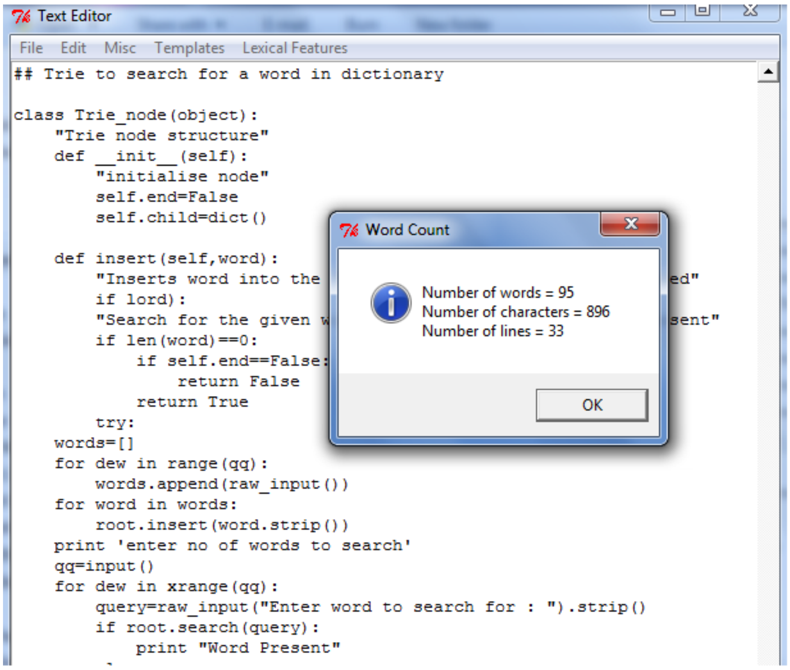
This function opens up a new text box where inputted text is compared with all the keywords in python and integers and syntax is highlighted.



```
if 1 :
        print 'a'
else :
        print 'b'
```

- ## Word Count

This function returns a count of all the words, lines, and characters in the text box.

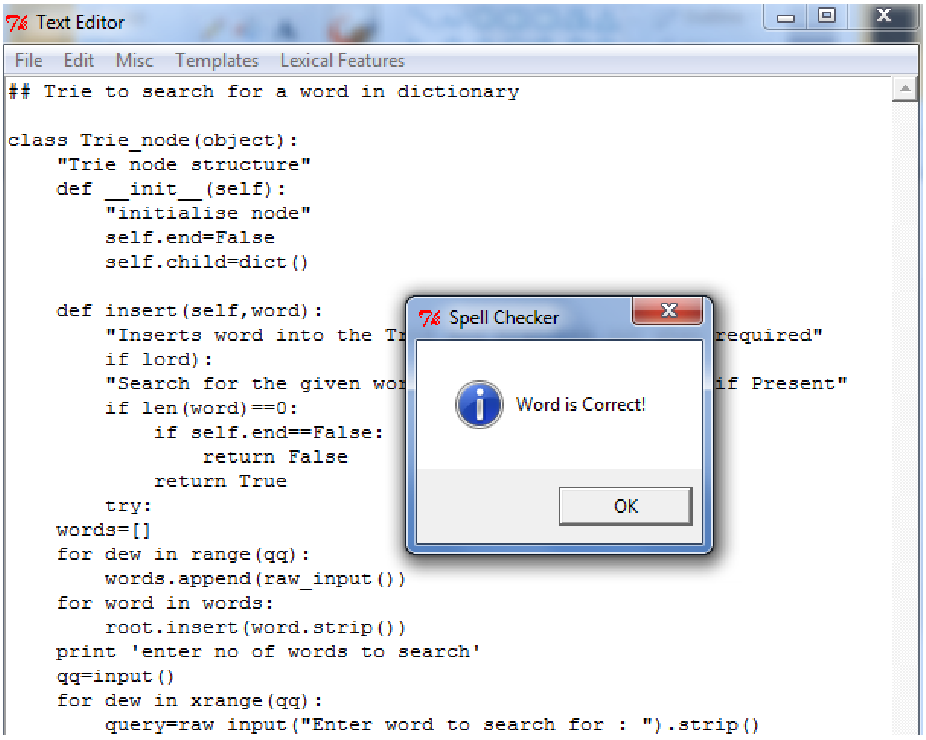


```
## Trie to search for a word in dictionary

class Trie_node(object):
    "Trie node structure"
    def __init__(self):
        "initialise node"
        self.end=False
        self.child=dict()

    def insert(self,word):
        "Inserts word into the                               ed"
        if lord):
        "Search for the given w                              sent"
        if len(word)==0:
            if self.end==False:
                return False
            return True
        try:
    words=[]
    for dew in range(qq):
        words.append(raw_input())
    for word in words:
        root.insert(word.strip())
    print 'enter no of words to search'
    qq=input()
    for dew in xrange(qq):
        query=raw_input("Enter word to search for : ").strip()
        if root.search(query):
            print "Word Present"
```

Number of words = 95
Number of characters = 896
Number of lines = 33

- **Word Check**

The word check function goes through a highlighted word in the text box and returns if the word is a valid word or not.

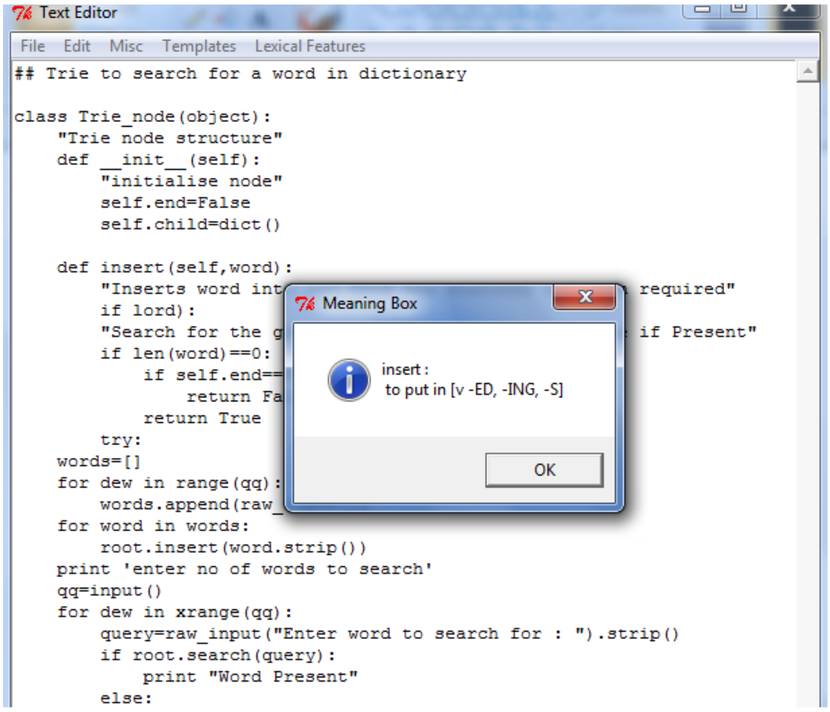


Word is Correct!

```
## Trie to search for a word in dictionary

class Trie_node(object):
    "Trie node structure"
    def __init__(self):
        "initialise node"
        self.end=False
        self.child=dict()

    def insert(self,word):
        "Inserts word into the Tr                       required"
        if lord):
        "Search for the given wor                       if Present"
        if len(word)==0:
            if self.end==False:
                return False
            return True
        try:
    words=[]
    for dew in range(qq):
        words.append(raw_input())
    for word in words:
        root.insert(word.strip())
    print 'enter no of words to search'
    qq=input()
    for dew in xrange(qq):
        query=raw input("Enter word to search for : ").strip()
```

- **Meaning Check**

The meaning check function goes through a highlighted word in the text box and returns the meaning of the word from a dictionary.



```
## Trie to search for a word in dictionary

class Trie_node(object):
    "Trie node structure"
    def __init__(self):
        "initialise node"
        self.end=False
        self.child=dict()

    def insert(self,word):
        "Inserts word int                              required"
        if lord):
        "Search for the g                              if Present"
        if len(word)==0:
            if self.end==
                return Fa
            return True
        try:
    words=[]
    for dew in range(qq):
        words.append(raw_
    for word in words:
        root.insert(word.strip())
    print 'enter no of words to search'
    qq=input()
    for dew in xrange(qq):
        query=raw_input("Enter word to search for : ").strip()
        if root.search(query):
            print "Word Present"
        else:
```

insert :
to put in [v -ED, -ING, -S]

SOFTWARE SPECIFICATIONS AND TOOLS

These are some of the tools we have used during the implementation of our project.

- Python 2.7
- IDLE Interface
- Various tkinter API functionalities like asksaveasfilename, askopenfilename, askstring, settext, gettext, pack, etc.

CHALLENGES FACED

During the course of this text editor project, we faced a lot of challenges, especially due to encapsulating our entire text editor by making use of the tkinter text widget. The choice of the GUI was another issue faced as it was the first step to starting the entire editor from scratch. Syntax highlighting was extremely difficult to implement as it was a new concept to us. Online documentation and coding forums helped us find many solutions to these problems.

CONCLUSION

We are extremely grateful on being given this opportunity to create and develop a text editor from scratch. Through this project we were provided with the opportunity to learn a new programming language: Python. Each member of the group was able to adequately implement some functions and had a chance to learn and master different aspects of the language. We would like to thank MR. Deepak Prabhu for providing us with the prospect of using our programming skills to implement a real world function. The different functions we have implemented are:

- Open
- Save as
- Find
- Cut, Copy, Paste
- Select All
- Delete
- Undo, Redo
- Team Members, Acknowledgements, References
- Templates
- Python Syntax Highlighter
- Word Check
- Meaning Check