**TEXT ANALYSIS**

**A REPORT ON PROJECT BASED LEARNING**

**(SEMESTER -II)**

*Submitted by*

AKASH REDDY KUMMETA (10407)

MILAN KUMAR PARIDA (10409)

SOHAM RAVINDRAN (10411)

RAJ MAVANI (10417)

DODIA RONAK (10420)

**FIRST YEAR ENGINEERING**

## Society for Computer Technology and Research’s

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

## DHANKAWADI, PUNE – 43

## A.Y. 2020-21

**- CERTIFICATE-**

This is to certify that the work incorporated in the report entitled **“TEXT ANALYSIS”,** is carried out by Mr./Ms. (Surname- First name- Middle

name) (Roll No (five digits)), who is part a group of students

with Project Id 04A42021 under the subject ***Project Based Learning*** during A.Y.

## 2010-2021.

Such material has not been submitted to any other University/ Institute for any financial support. The literature related to the problem investigated has been appropriately cited and duly acknowledged wherever facilities and suggestions have been availed of.

## Date: Name & Sign of Project Guide

**Place:** PUNE

Prof. A. A. Chavan

**Name & Sign of PBL Coordinator Name & Sign of Head of Department**

**Prof. N. P. Sapkal Prof. E. M. Reddy**

**Abstract**

Many of us face grammar issues and mistakes in our daily day to day life. That can be while writing an E-mail, text, application writing or even during a presentation.

We intend to solve this problem by using python programming language. The python programming language consists of a various libraries such as TextBlob which is mainly used for text analysis using various inbuilt algorithms and classifiers. So we have used this library in our programme to help solve spelling errors, translate a given word or sentence from one language to another language and used to singularize or pluralize a given word PyDictionary for the meaning and Gingerit Library for checking grammatical errors.

So the primary task of our project is to solve the most common issues in grammar debris using coding and programming. Our intent is to reduce time and reduce the communication gap between people.

In this way our programme solves the issues and mistakes done by people in their daily day to day life while writing any E-mail or text message.

**ACKNOWLEDGEMENT**

This project consumed huge amount of work, research and dedication. Still, implementation would not have been possible if we did not have a support of many individuals. Therefore, we would like to extend our sincere gratitude to all of them.

First of all we are thankful to SCTR Pune Institute of Computer Technology for their logistical support and for providing necessary guidance concerning projects implementation.

We are also grateful to Prof. A. A. Chavan for being an expertise, and a great moral support in the implementation. Without their superior knowledge and experience, the Project would lake in quality of outcomes, and thus their support has been essential.

We would like to express our sincere thanks towards all researchers who devoted their time and knowledge in the implementation of this project.

Nevertheless, we express our gratitude toward our families and colleagues for their kind co-operation and encouragement which help us in completion of this project

Place: Name of the Student &Sign

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Title** | **Page No.** |
| 1 | INTRODUCTION | 6 |
| 2 | BASIC LOGIC |  |
| 3 | BASIC GUI |  |
| 4 | ADDING MULTIPLE BUTTONS FOR FUNCTIONS |  |
| 5 | ADDING MULTIPLE SCREENS FOR FUNCTIONS |  |
| 6 | METHODOLOGY |  |
| 7 | CONCULSION |  |

**CHAPTER 1**

**INTRODUCTION**

**CHAPTER 2**

**BASIC LOGIC**

Once we had decided the idea we chose the language python as the base language for our project.it had the libraries required for executing the program.

Python is a programming language that lets you work quickly and integrate systems more effectively.

We started working on the main logic of the code will run functions such as word correction, meaning, grammatical correction, translation and as such.

We used nested if and while loops to and imported libraries such as textblob and nltk.

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](http://nltk.org/nltk_data/) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries

------------------------------------------------------------------------

We made the following code in using these libraries:-

!pip install -U textblob

!pip install nltk

import nltk

nltk.download('wordnet')

from nltk.corpus import wordnet

nltk.download('punkt')

from textblob import TextBlob

a=int(input("welcome to PICT\_grammerly \n 1.do you want to correct a spelling?\n 2.Do you want to check the number of particular word/noun in a text\n 3.do you want to singularize or pluralize a word\n 4.Meaning of a word\n 5.sentiment of sentence\n 6.complete correction of a sentence.\n 7.Synonym/antonym of a word.\n 8.count the number of words in a text.\n"))

if a==1:

wrd=str(input("Please tell the word to be checked."))

b = TextBlob(wrd)

print(b.correct())

elif a==2:

sntnc=TextBlob(input("Enter the sentence "))

wrd1=str(input("Enter the word whose frequency is to be found."))

k=sntnc.word\_counts[wrd1]

print(k)

elif a==3:

b=str(input("enter the word to be singularized or pluralized "))

b=TextBlob(b)

c=str(input("do you want to singularize or pluralized?(S/P) "))

if c=="S" or c=="s":

b.singularize()

elif c=="P" or c=="p":

print(b.pluralize)

elif a==4:

wrd2=wordnet.synsets(input("Enter the word to find its meaning "))

print(wrd2[0].definition())

elif a==5:

stc=TextBlob(input("Enter the Sentence."))

print(stc.sentiment)

if stc.sentiment.polarity < 0:

print('negative statement')

elif stc.sentiment.polarity== 0:

print("It is a neutral statement")

else:

print("positive statement")

elif a==6:

l=TextBlob(input("Enter the grammatically wrong sentence."))

print(b.correct)

elif a==7:

m=wordnet.synsets(input("Enter the word"))

y=str(input("do you want its synonym or antonym?"))

if y.upper()=="SYNONYM":

print(m[0].lemmas()[0].name())

else:

antonyms = []

for syn in m:

for l in syn.lemmas():

if l.antonyms():

antonyms.append(l.antonyms()[0].name())

print(antonyms[0])

elif a==8:

p=str(input("Enter the text"))

q=list(p.split())

for i in q:

if i =='?'or i=='.' or i=="!":

q.remove(i)

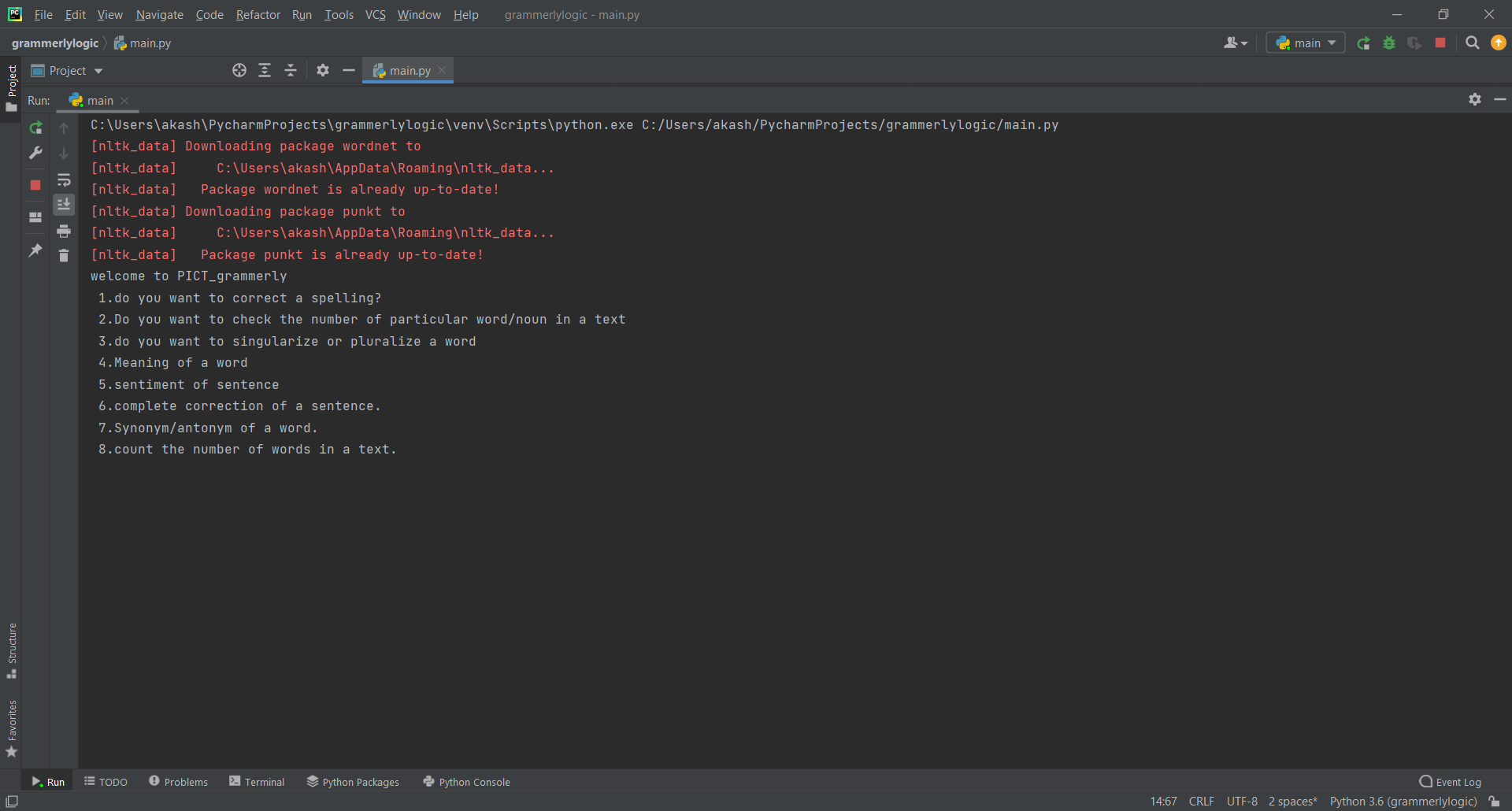
print("The total number of words are: ",len(q))

print("Thank you!!")

------------------------------------------------------------------------------------------------------------------------------------------

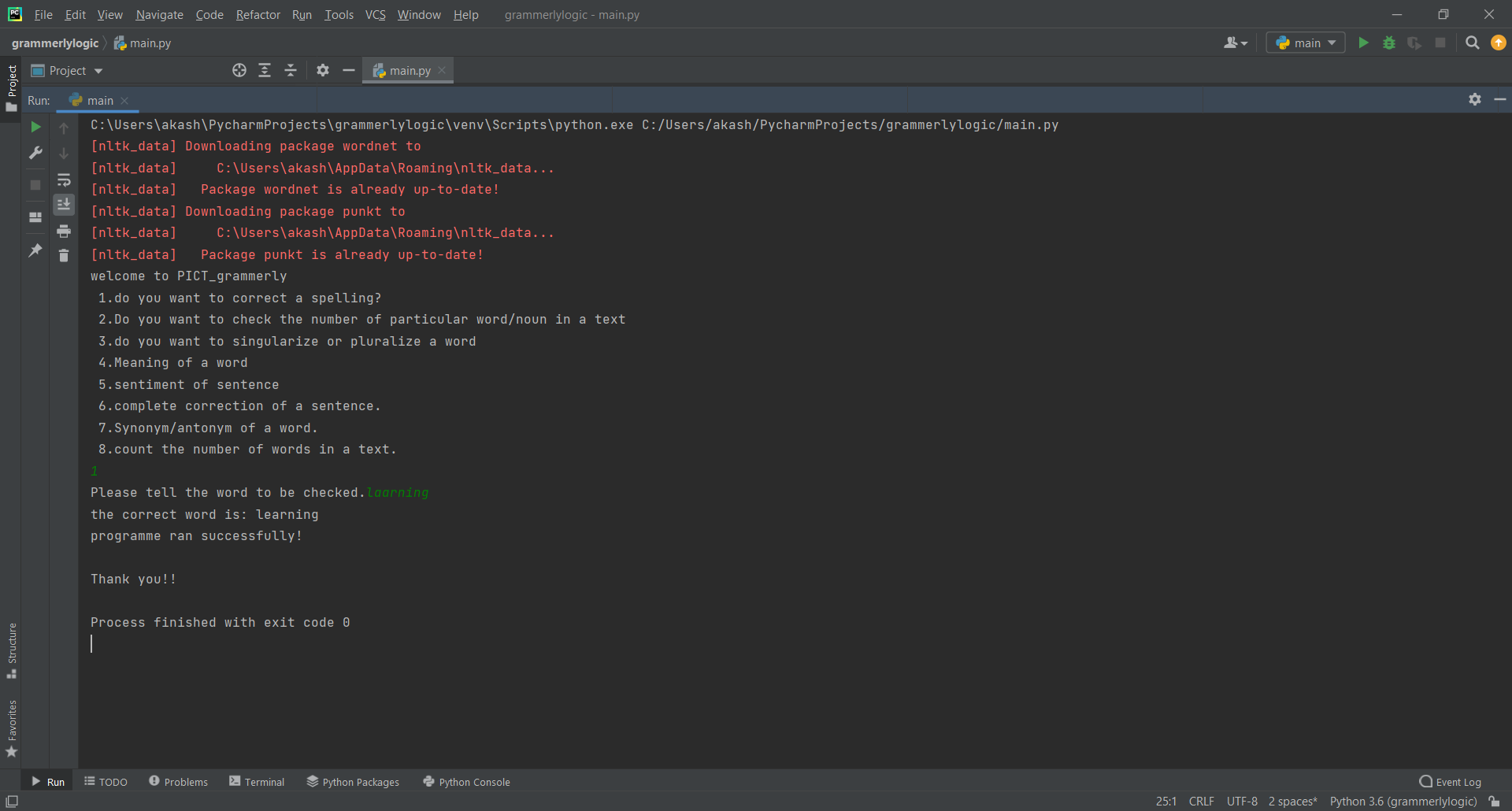
The following are the results of this code being run

We didn’t work on the GUI yet so the result was simple text in the compiler.



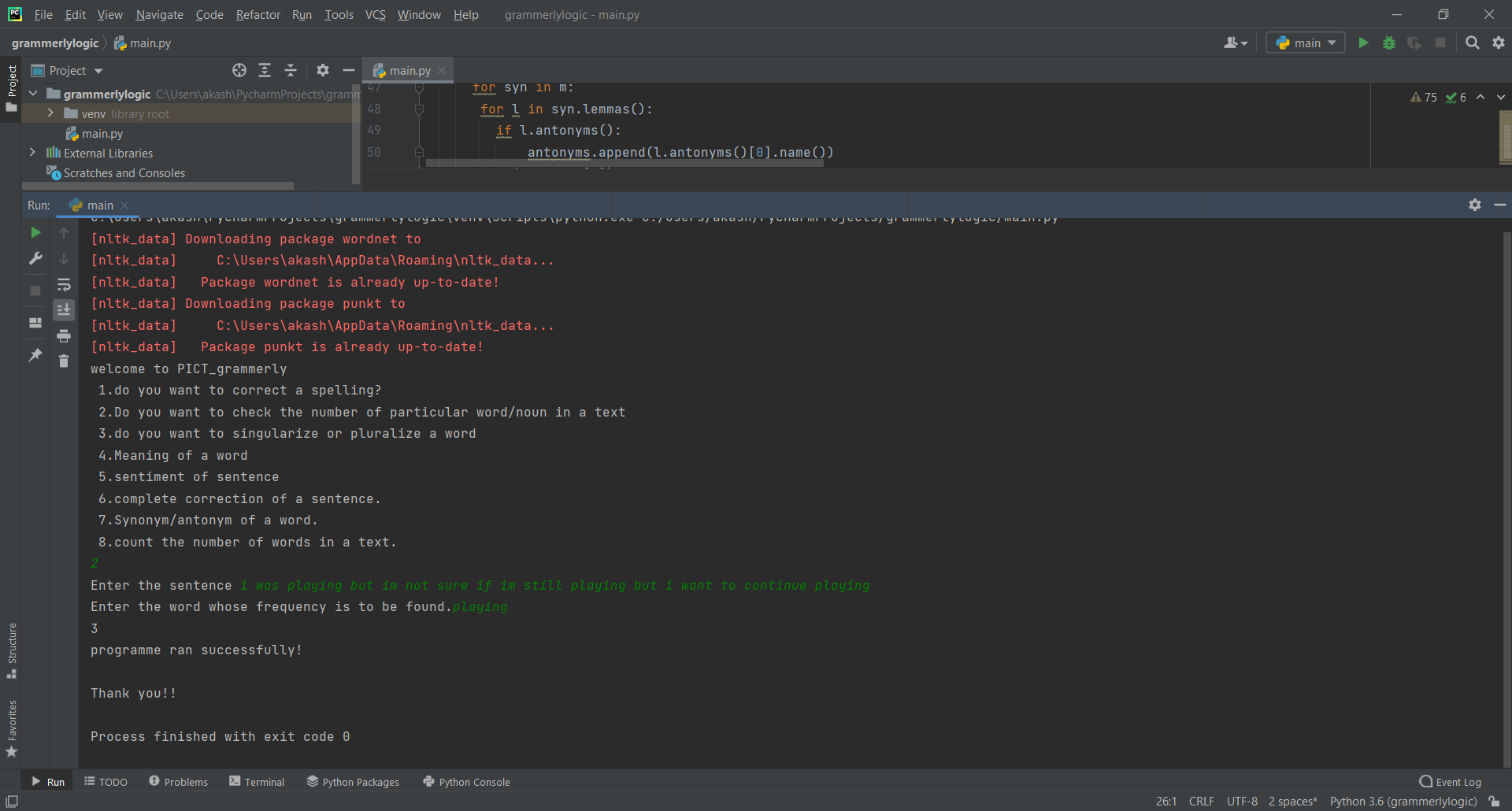
---------------------------

Word correction:-



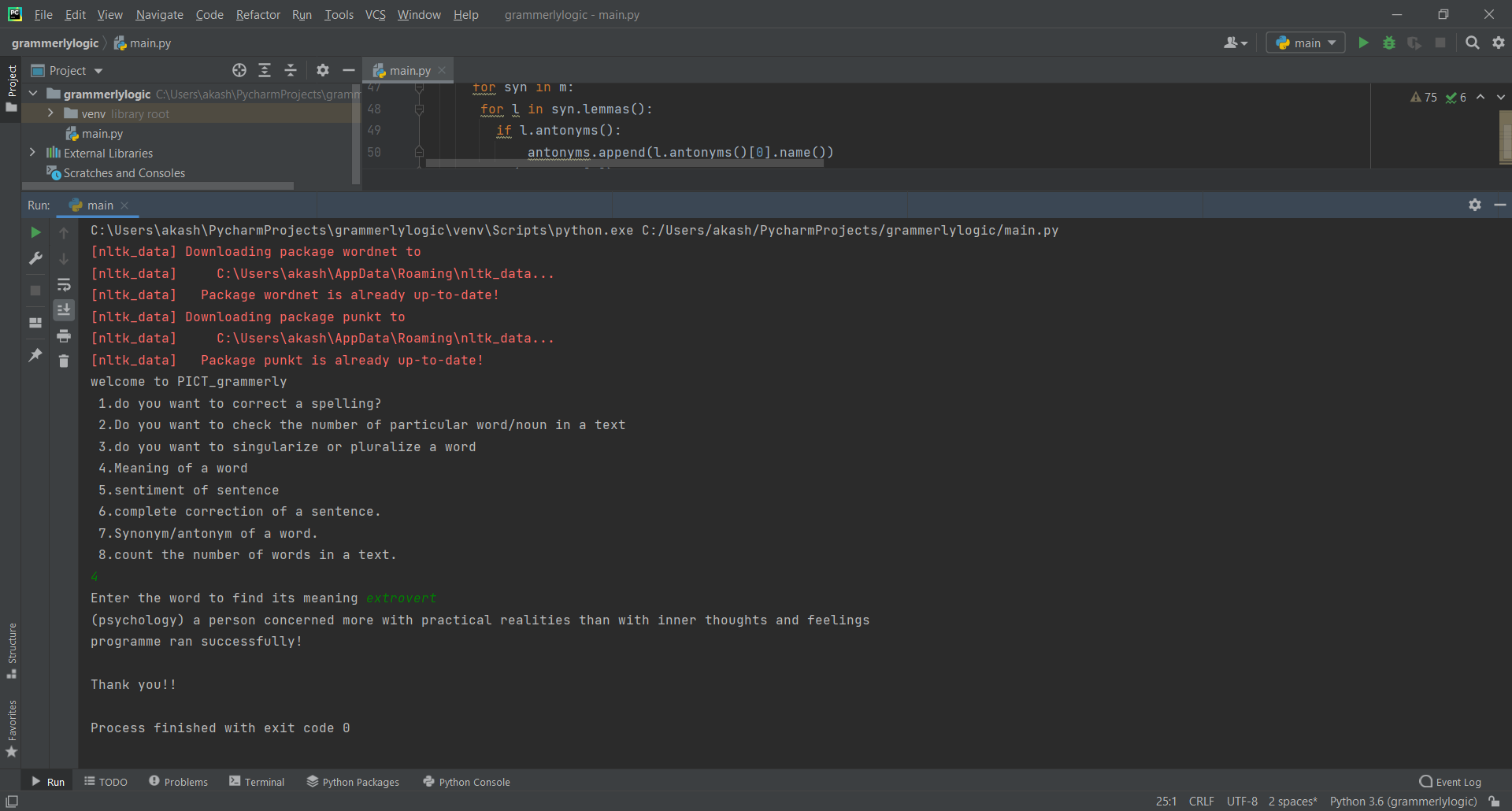
----------------------

Word frequency:-



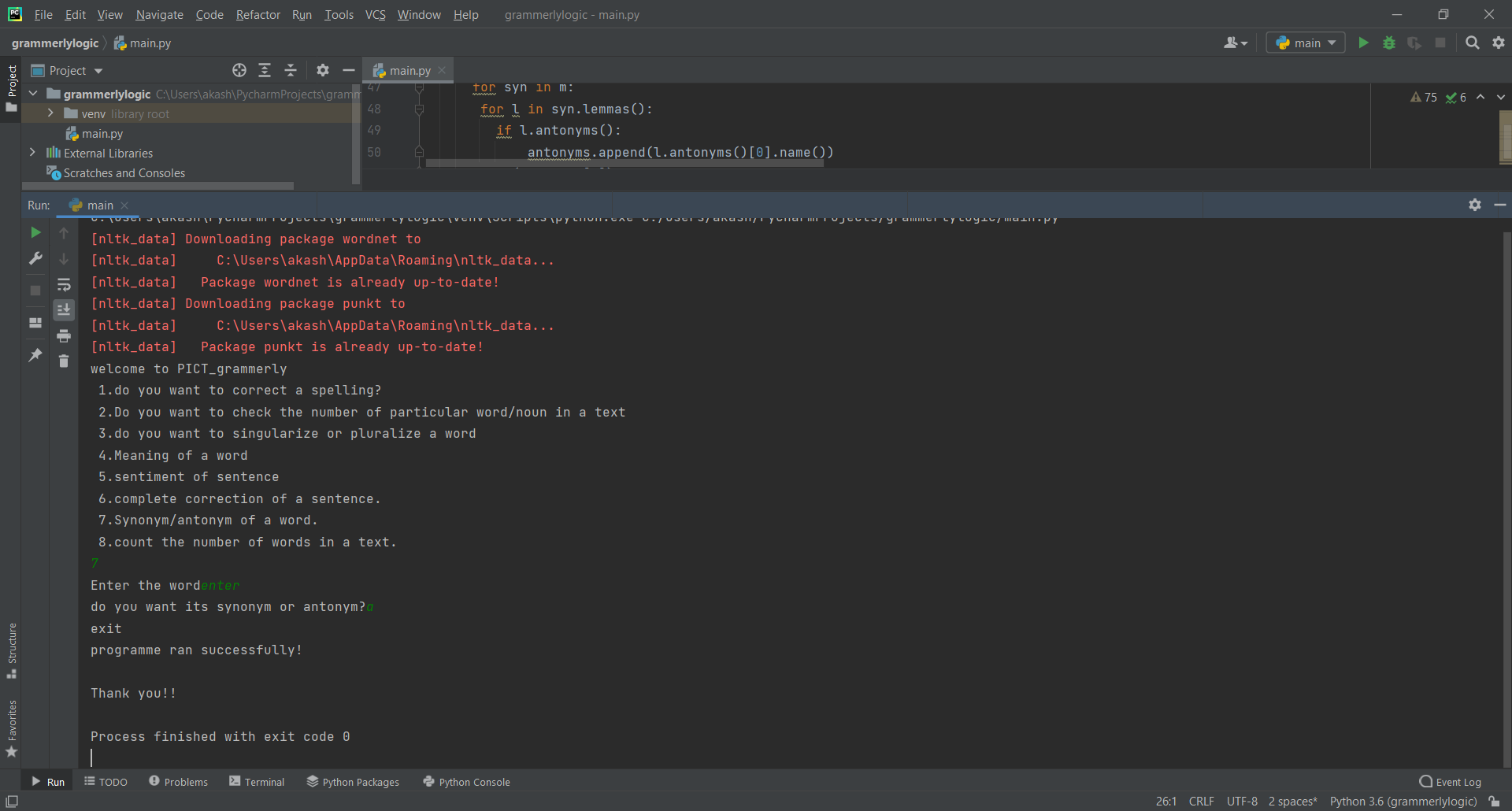
-----------------------------

Word meaning:-



----------------------------

Word antonym/synonym:-



**CHAPTER 3**

**BASIC GUI**

We worked on more functions and now it was time for us to create a proper gui which would look good in both desktop and mobile version.

Our approach to this was using python itself by the libraries kivy and kivyMD.

Kivy - Open source Python library for rapid development of applications  
that make use of innovative user interfaces, such as multi-touch apps.

KivyMD is a collection of Material Design compliant widgets for use with Kivy, a framework for cross-platform, touch-enabled graphical applications.

Kivy allows us to create a user friendly interface like changing background color, a flash screen, creating multiple pages,textbox,buttons,text field images, etc.

We created our first graphic user interface for one fuction which was word meaning and later on added more functions.

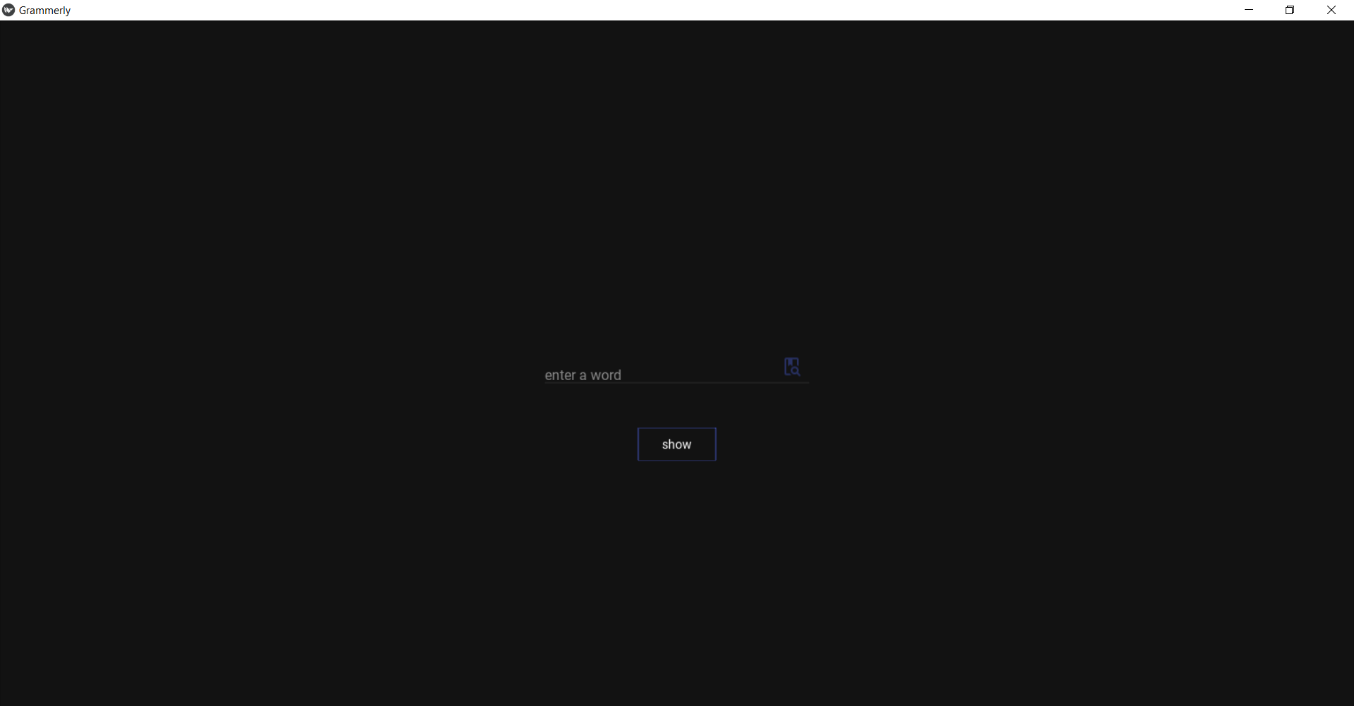
------------------------------------------------------------------------------------------------------------

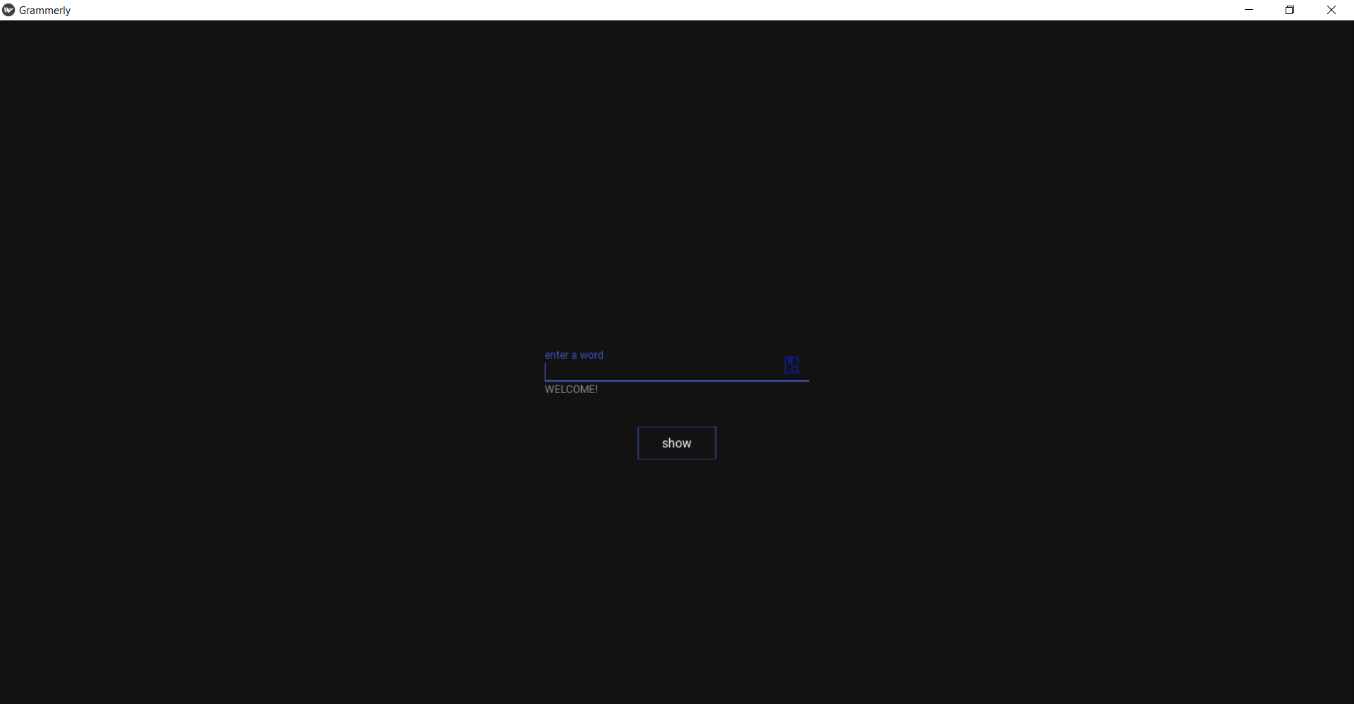
The code for this is:-

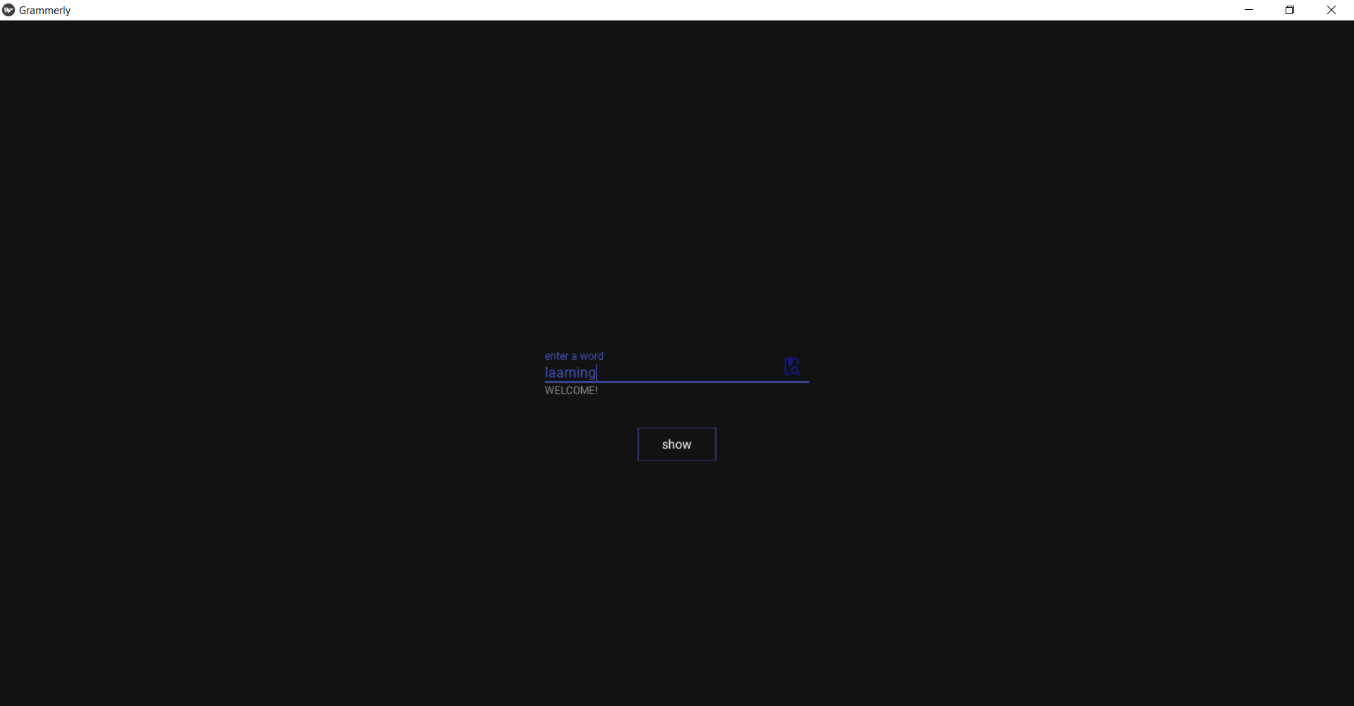
from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivymd.uix.button import MDRectangleFlatButton,MDFlatButton  
from kivymd.uix.dialog import MDDialog  
from kivy.lang import Builder  
from helper import username\_helper  
import nltk  
nltk.download('wordnet')  
  
nltk.download('punkt')  
from textblob import TextBlob  
  
class GrammerlyApp(MDApp):  
 def build(self):  
 screen=Screen()  
 self.theme\_cls.theme\_style="Dark"  
 self.theme\_cls.primary\_palette="Indigo"  
 button=MDRectangleFlatButton(text='show',pos\_hint={'center\_x':0.5,'center\_y':0.4},  
 on\_release=self.show\_data)  
 self.username=Builder.load\_string(username\_helper)  
 screen.add\_widget(self.username)  
 screen.add\_widget(button)  
 return screen  
  
 def show\_data(self,obj):  
 if self.username.text is "":  
 check\_string = 'please enter a word'  
 close\_button = MDFlatButton(text='close', on\_release=self.close\_dialog)  
 more\_button = MDFlatButton(text='more')  
 self.dialog = MDDialog(title='Error!', text=check\_string, size\_hint=(0.7, 1),  
 buttons=[close\_button, more\_button])  
 self.dialog.open()  
 else:  
  
 b= TextBlob(self.username.text)  
 b=str(b.correct())  
  
 close\_button=MDFlatButton(text='close',on\_release=self.close\_dialog)  
 more\_button=MDFlatButton(text='more')  
 self.dialog=MDDialog(title='correction',text=b,size\_hint=(0.7,1),  
 buttons=[close\_button,more\_button])  
 self.dialog.open()  
  
 def close\_dialog(self,obj):  
 self.dialog.dismiss()  
GrammerlyApp().run()

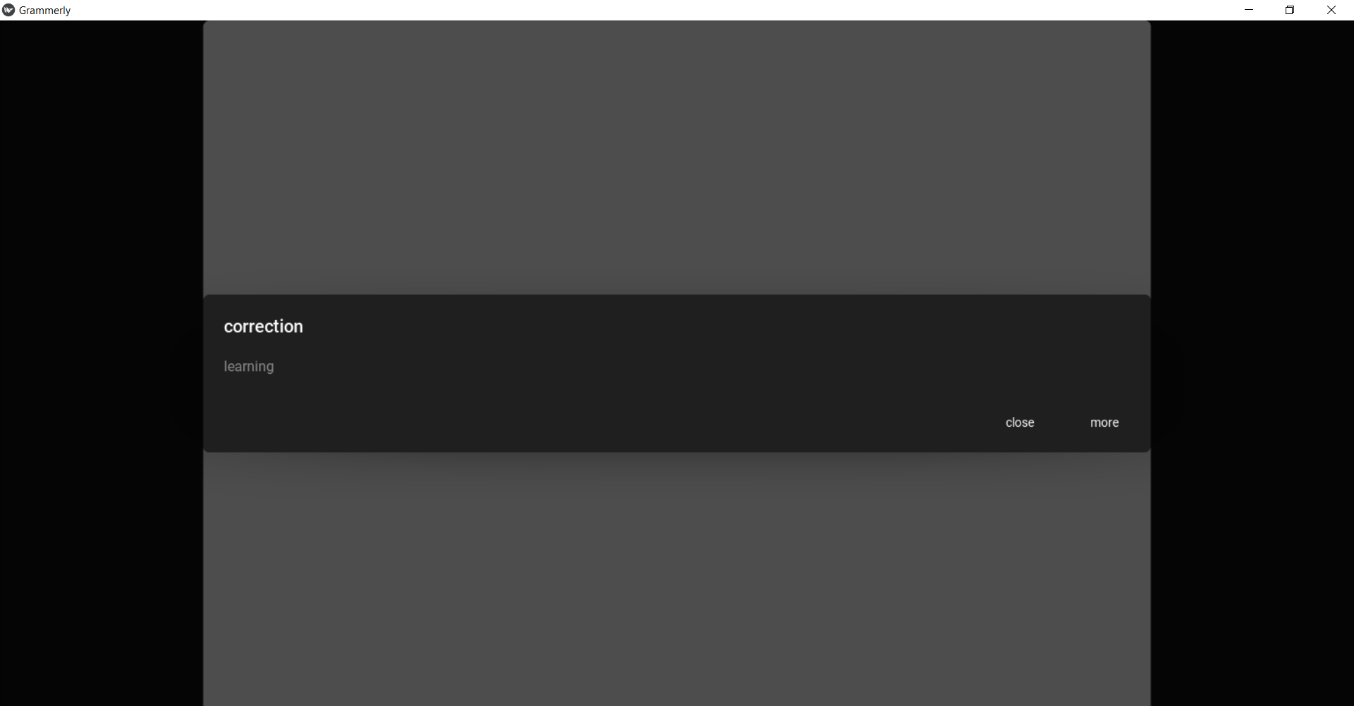
-----------------------------------------------------------

The following are the results:-









## Chapter 4

**ADDING MULTIPLE BUTTONS FOR FUNCTIONS**

To build our applications more user-friendly and easy we use different widgets such as buttons, text fields, labels,dialog boxes etc

Widgets: - Widgets are elements of a graphical user interface that form part of the User Experience.

Label: The Label widget is for rendering text. It supports ascii and unicode strings. Label is the text which we want to add on our window, give to the buttons and so on. On labels, we can apply the styling also i.e increase text, size, color and more.

Text Field: Text fields let users enter and edit text.

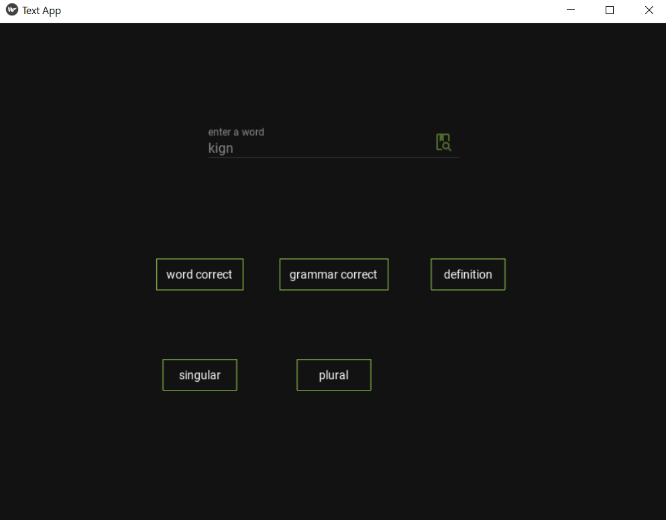
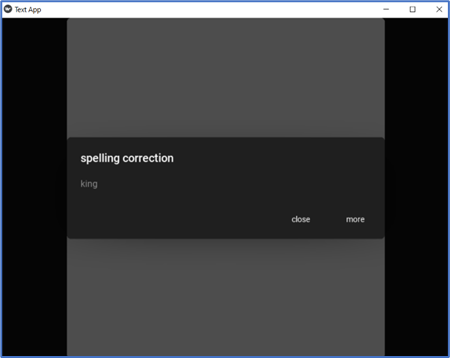
Buttons: Buttons allow users to take actions, and make choices, with a single tap.

Dialog Box: Dialogs inform users about a task and can contain critical information, require decisions, or involve multiple tasks.

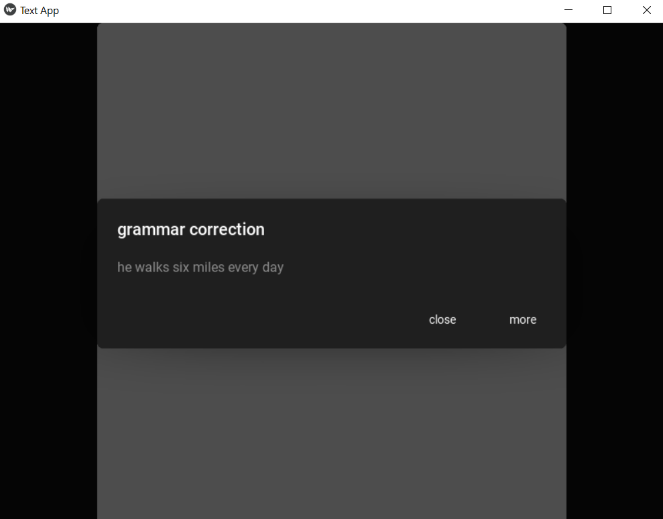
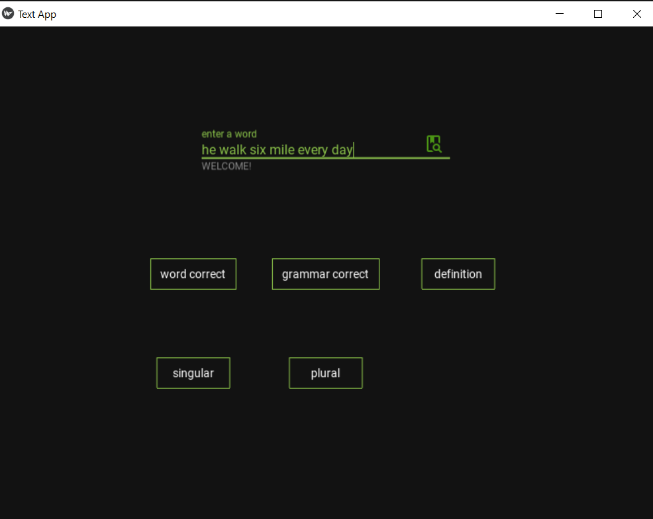
We use the similar code from previous chapter to add the functions.

To show the output, dialog boxes are used.

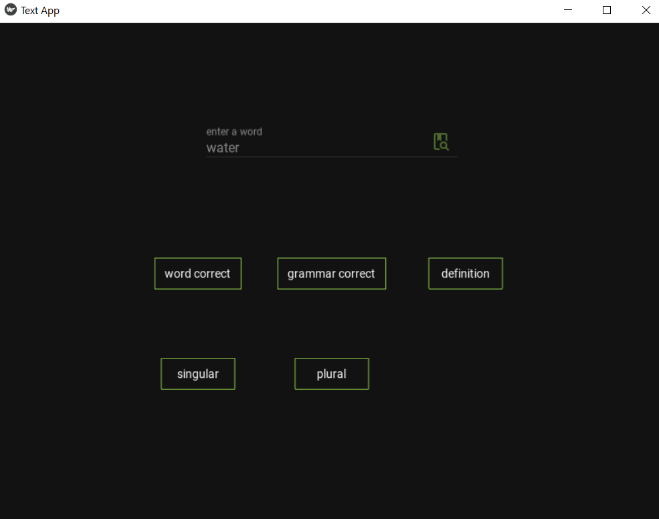
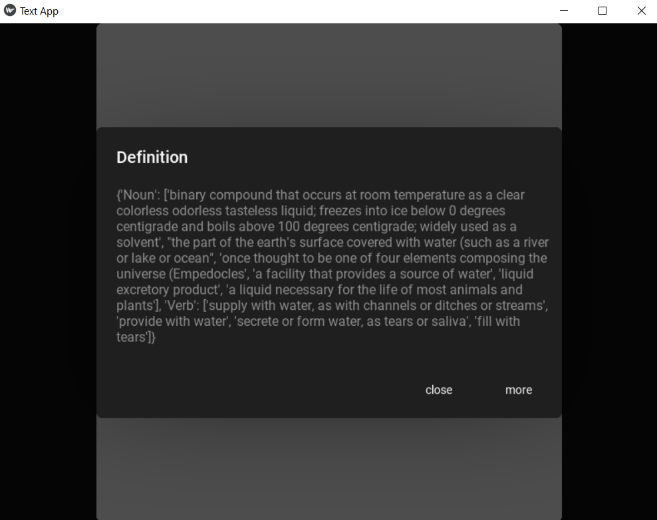
There are five buttons: -

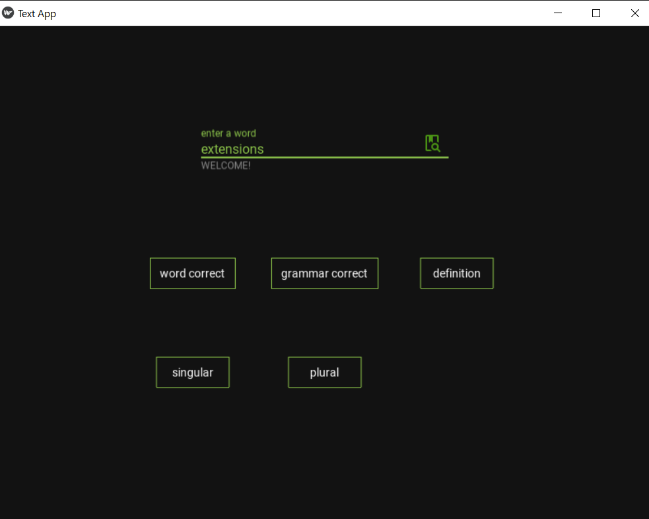
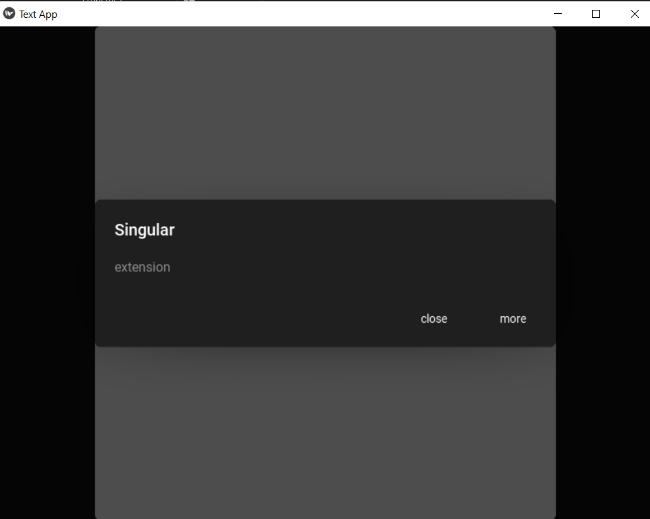
1] word correct: - to correct the spelling error in the text 

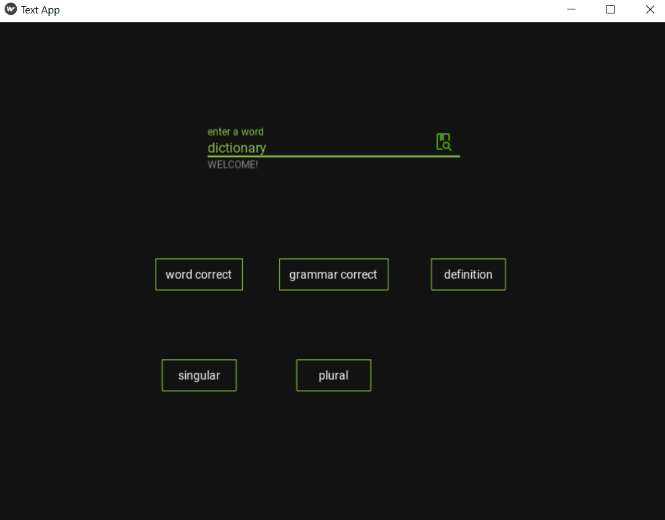
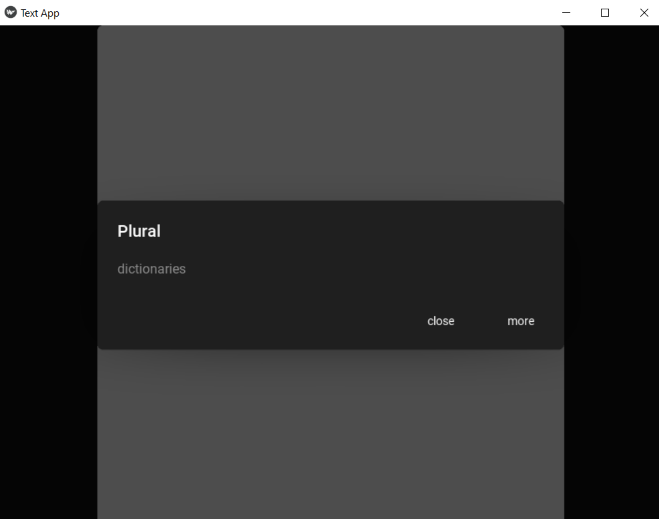
2] grammar correct: - to correct the grammatical mistakes

3] definition: - to show the meaning of the text(word)

4] singular: - to show the singular of the word 

5] plural: - to show the plural of the word  

To close the dialog boxes, press on the close button.

**Chapter 5**

**ADDING MULTIPLE SCREENS FOR FUNCTIONS**

we had to change our approach rather than using the dialog boxes for the output, we

As our application had grown more complex, the construction of widget trees and explicit declaration of bindings becomes verbose and hard to maintain. The KV Language is an attempt to overcome these shortcomings.

The KV language, sometimes called ‘kvlang’ or the ‘kivy language’, allows you to create your widget tree in a declarative way and to bind widget properties to each other or to callbacks in a natural manner. It allows for very fast prototypes and agile changes to your UI. It also facilitates separating the logic of your application and its User Interface.

By using kvlang we have added 5 functions which include: -

1] definitions/meaning, synonyms, antonyms, plural, singular

2] to correct grammatical mistakes

3] to correct spelling mistakes

4] translate the text

5] change theme

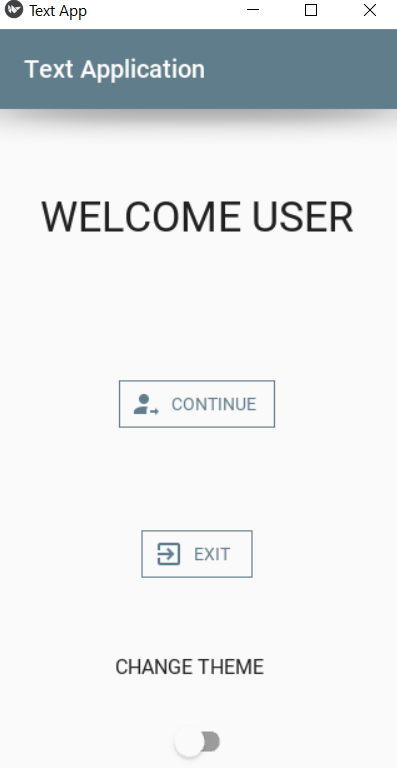
To achieve this we use different libraries such as TextBlob, PyDictionary and GingerIt.

PyDictionary: - PyDictionary is a Dictionary Module/Library for Python 2/3 to get meanings, translations, synonyms and Antonyms of words. It uses WordNet for getting meanings, Google for translations, and synonym.com for getting synonyms and antonyms.This module uses Python Requests, BeautifulSoup4 and goslate as dependencies.

GingerIt: - GingertIt module/ Library is used for correction of grammatical mistakes based on the context of complete sentences. By using the gingerit library in Python you can eliminate all the grammatical mistakes, fix your spellings and punctuation errors and at the end, it helps you to enhance your text.

We use multiple screens and to manage the screens the ScreenManager Module to achieve this.

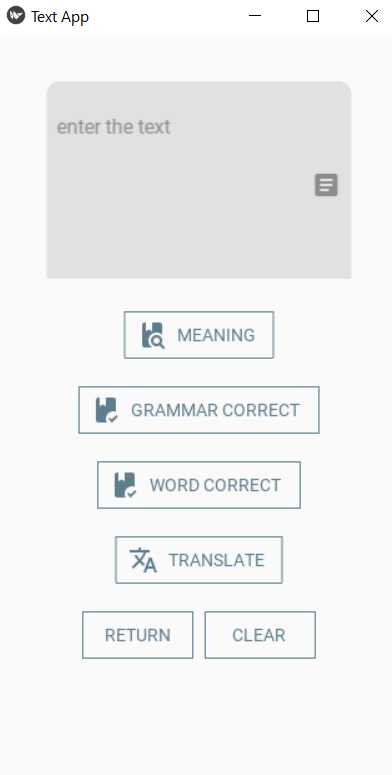
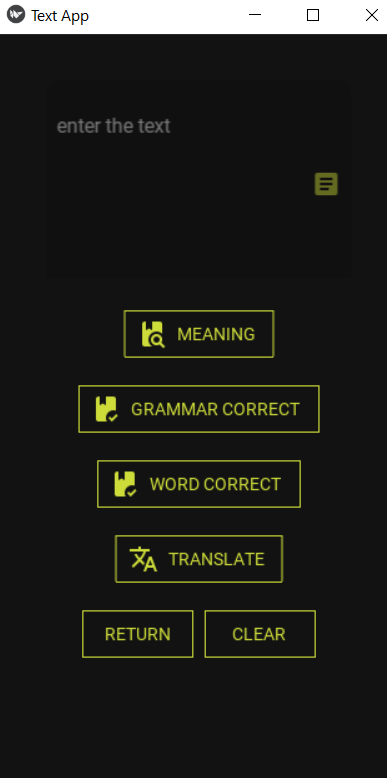
On opening the application the user is greeted with the text “WELCOME USER.



On pressing ‘CONTINUE’ button the screen is changed to the “Function screen”.

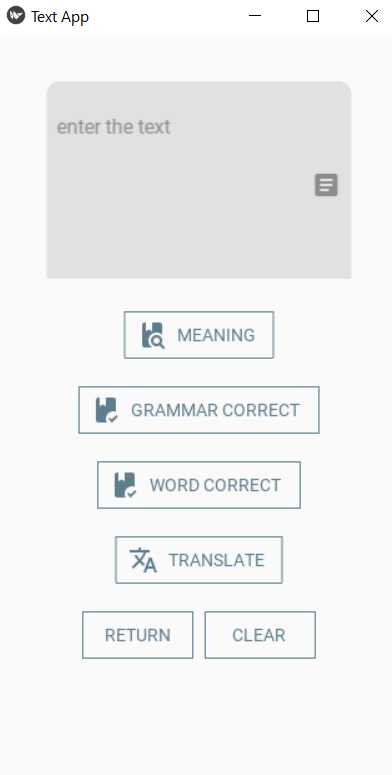
On pressing ‘EXIT’ button the application is closed.

On the bottom of the screen a switch is added to change the theme of the application

To change the theme a switch is used, when in off mode the primary theme is Light and the primary palette is BlueGray. When in on mode the primary theme is Dark and primary palette is Yellow.

Switch: - The Switch widget is operative or inoperative, as a mechanical switch. The user can swipe switch to the left/right to On/Off or activate/deactivate it.



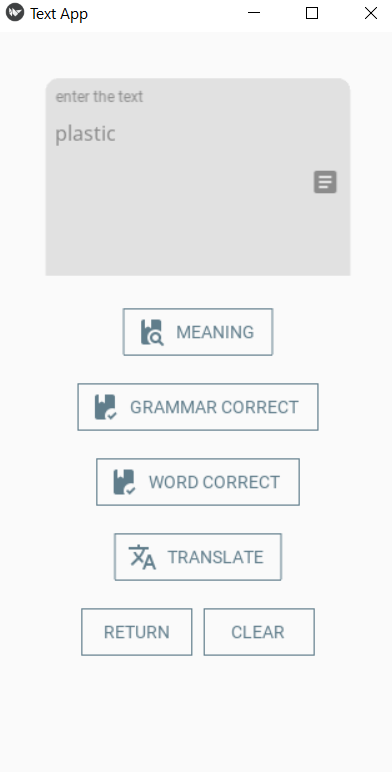
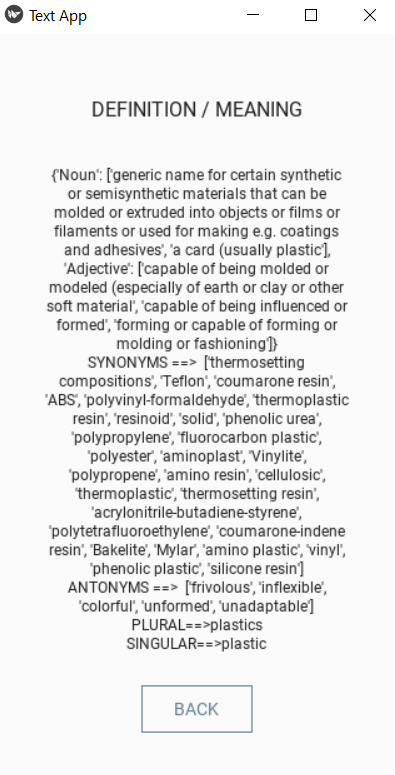
On Function screen on top of the screen there is the text filed where the user can input the data (text/word).

Buttons: -

1] MEANING: - It requires one a word has a input. On pressing the button, the screen is change to Meaning Screen. Where it shows the meaning of the word its synonyms, antonyms, plural and singular of the word. It uses the PyDictionary library for the meaning, synonym and antonym. For singular and plural TextBlob library is used.

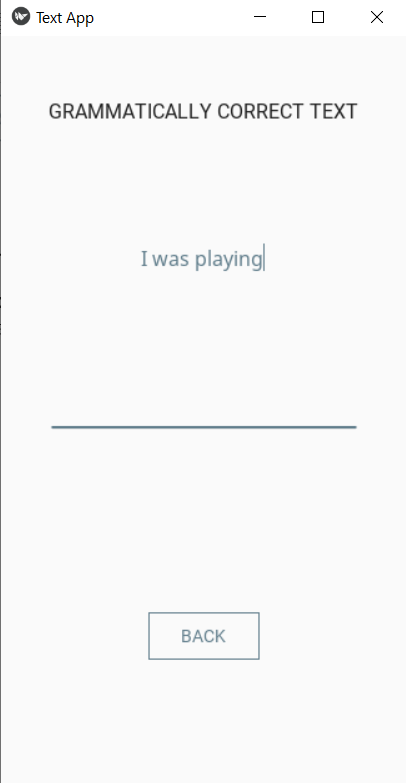
If there are no meaning, synonyms, plural etc. Found it shows the output as None.

If the textfield is empty it shows ‘ENTER THE TEXT’

2] GRAMMAR CORRECT: - It requires sentence/text as input. On pressing the button the screen is changed to Grammar Screen. Where it shows the grammatically correct sentence.

It uses GingerIt library. If the textfield is empty it shows ‘ENTER THE TEXT’

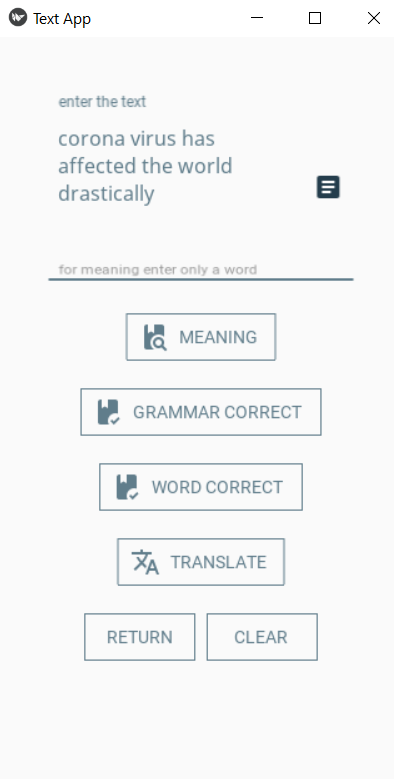
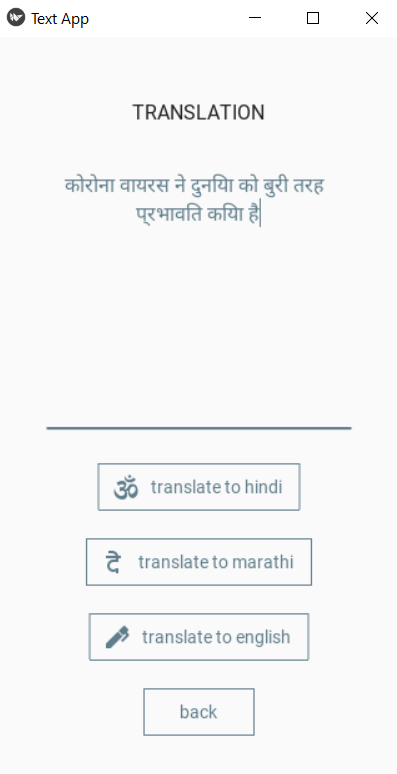
 

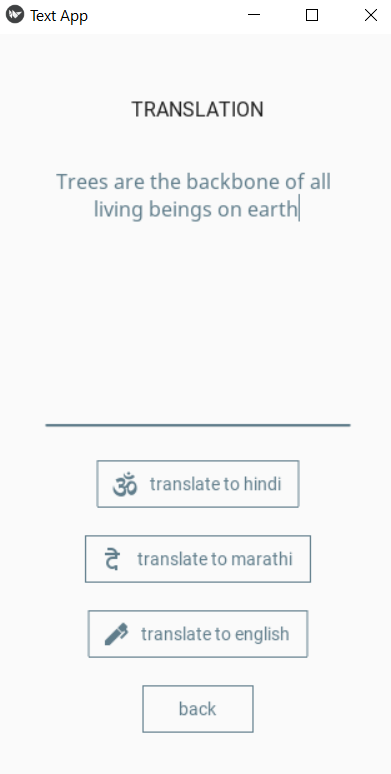
3] WORD CORRECT: - it accepts word and text as input. On pressing the button the screen is changed to Word Screen. Where it shows the correct spelling of the text.

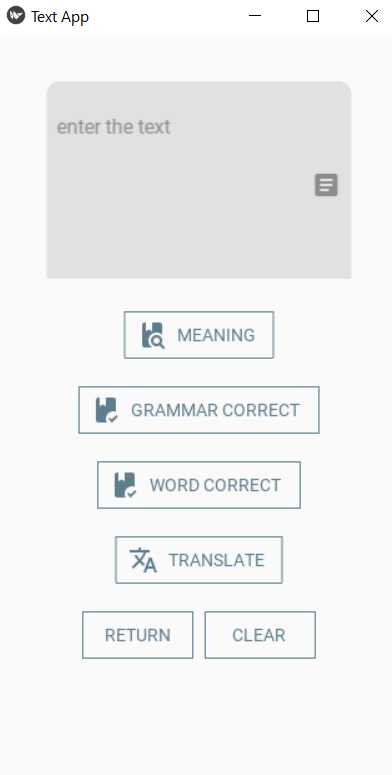
It uses TextBlob library. If the textfield is empty it shows ‘ENTER THE TEXT’

4] TRANSLATE: - It accepts word and text as input. On pressing the button the screen is changed to Translate Screen. On the screen the user can translate the text into English, Hindi and Marathi. It uses TextBlob library. For translation the font is change to Noto-Sans (Regular) has it have the necessary glyph for the translation to Hindi and Marathi. If the textfield is empty it shows ‘ENTER THE TEXT’

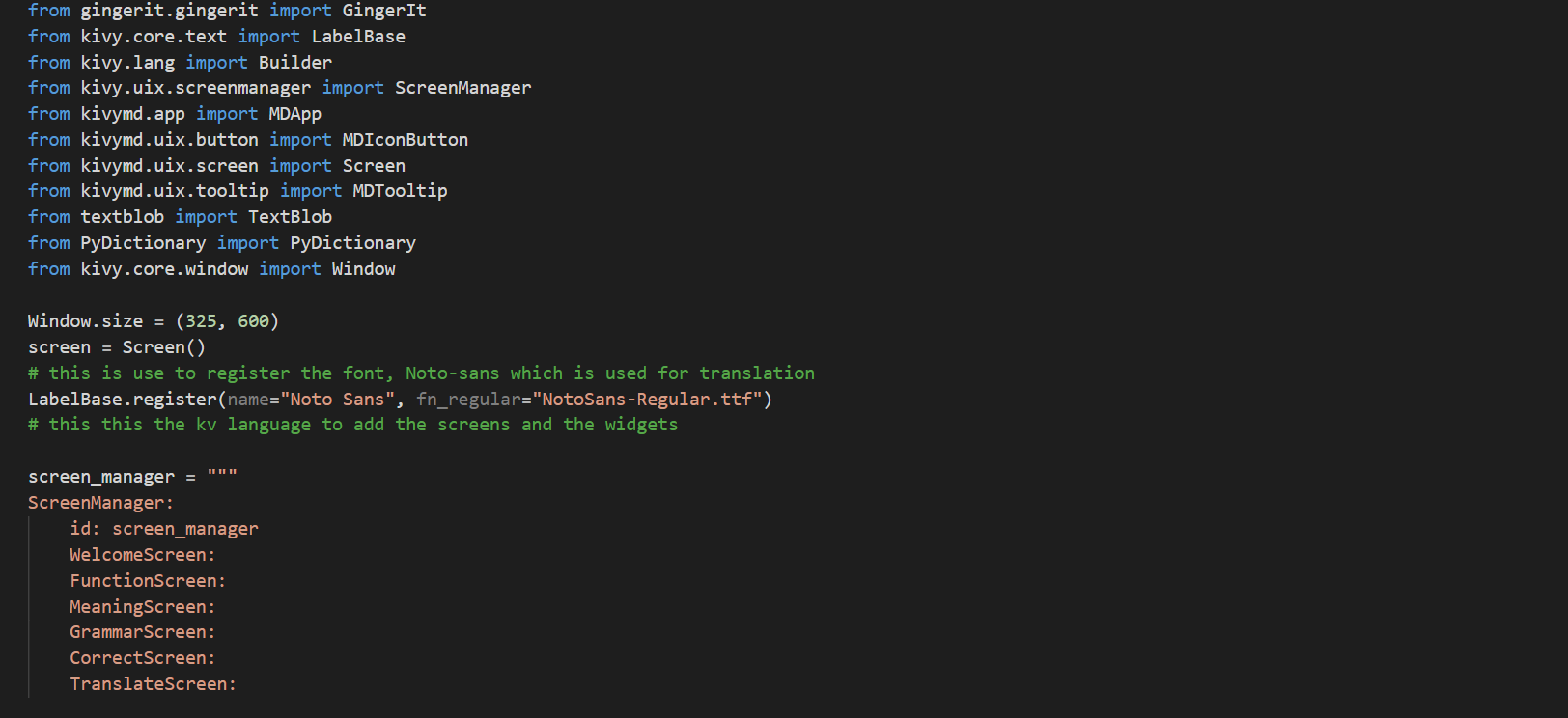
5] CLEAR: - on press it clears the textfield  

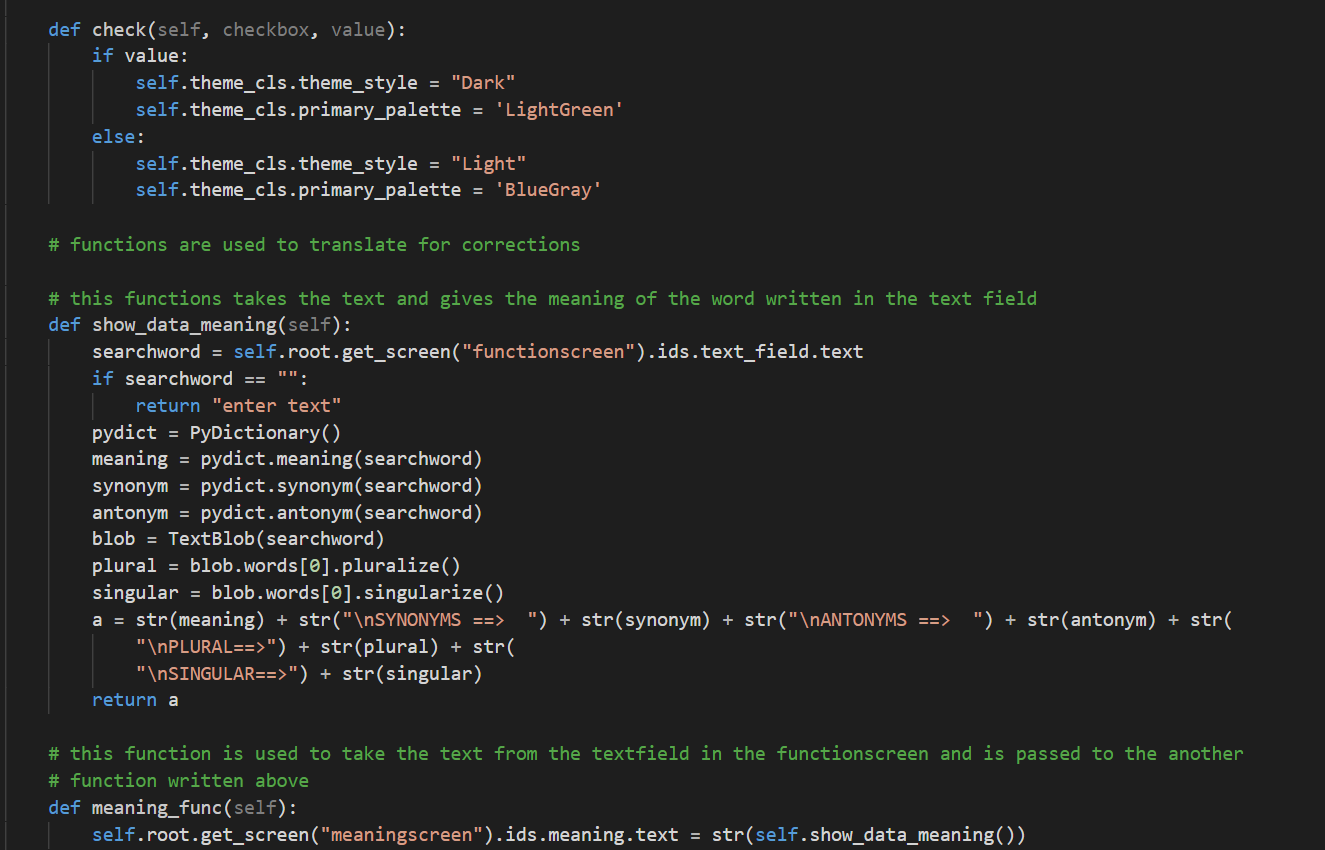
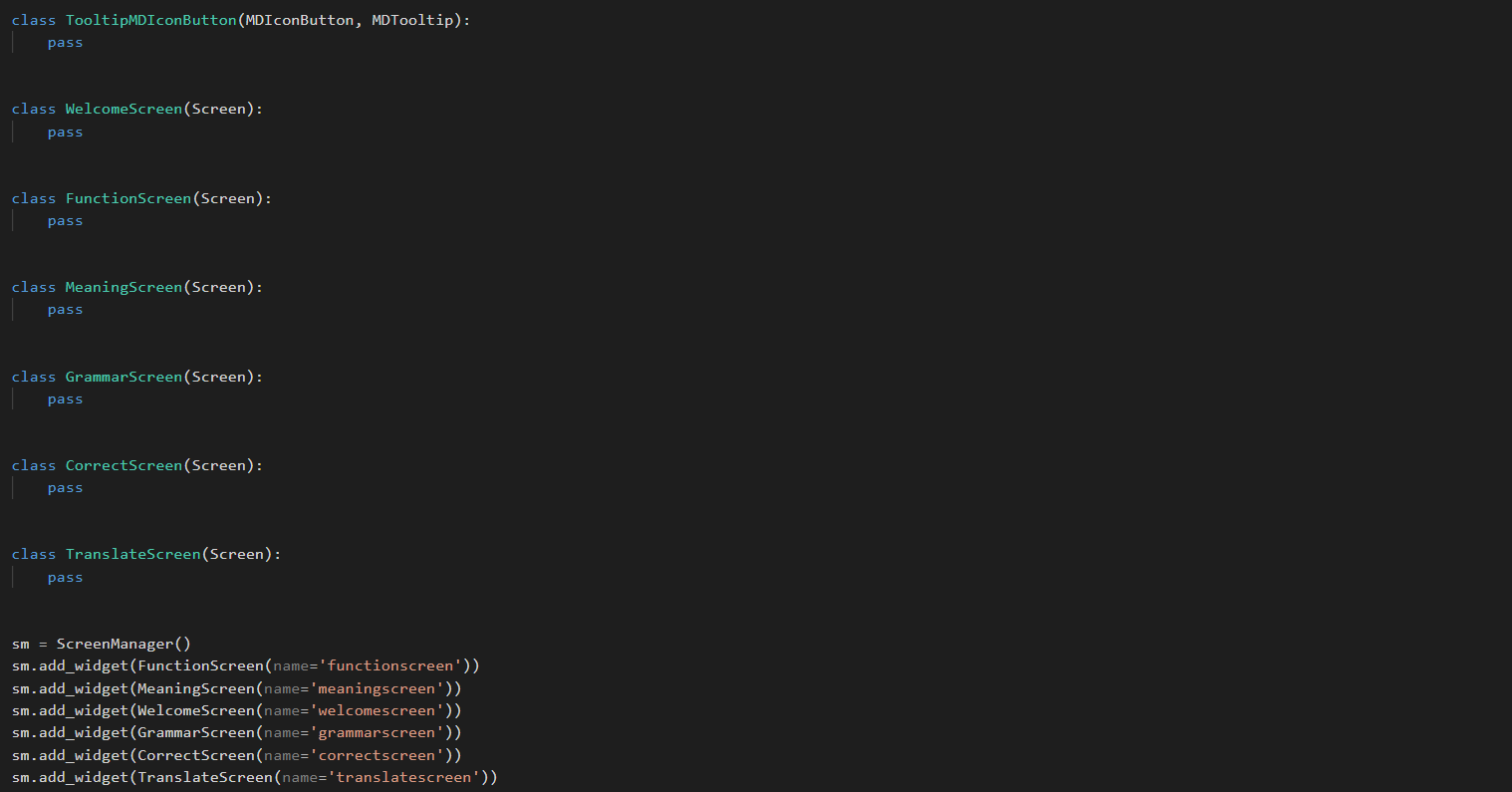
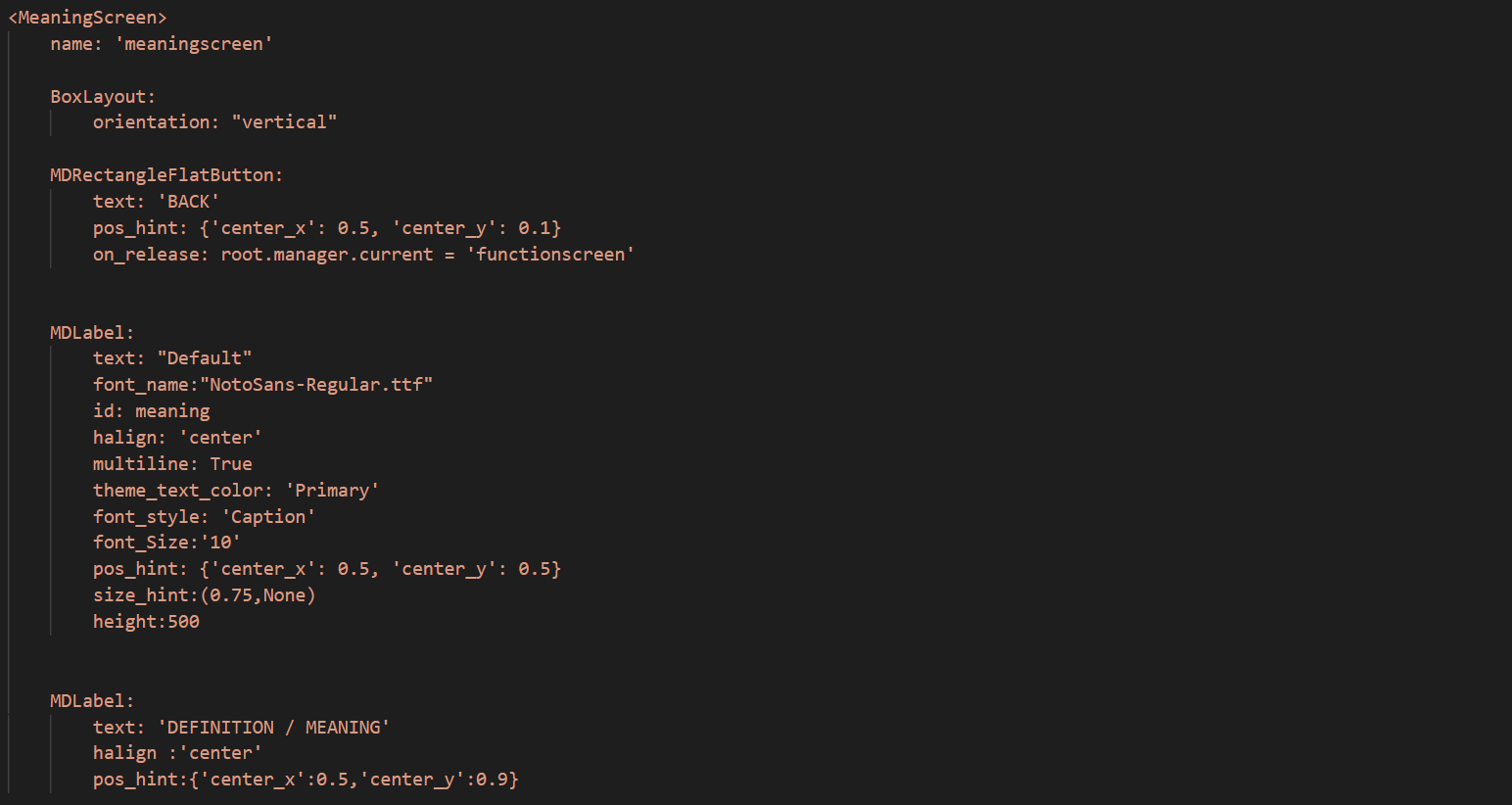
6] RETURN: - on press the screen goes back to the welcome screen.

**Chapter 6**

**METHODOLOGY**

Sample Code:-

`



In kv lang, a multiline string is used to add the screens, buttons, labels and icons.

To display the kv land builder method is used, The Builder is responsible for creating a Parser for parsing a kv file, merging the results into its internal rules, templates, etc.

To add multiple screens ScreenManager Module is used in the kv lang.

Screen Manager: - The screen manager is a widget dedicated to managing multiple screens for your application. The default ScreenManager displays only one Screen at a time and uses a TransitionBase to switch from one Screen to another.

After declaring the ScreenManager, to add screens we have to write the name of the screen with proper indentation.

After adding the name of the screen, to declare the screen you have to write

<(name of the screen)> and ‘:’ after it. To add widget on the screen, we have to use ‘:’ instead of ‘=’ sign.

Whenever a new screen is created, we have to add the class for the screen and add the screen in the screen manager.

In the sample code screen\_manager is used as the kv language. To add the meaning screen

we have to write MeaningScreen in the Screen Manager with indentation and to

declare the MeaningScreen, write <MeaningScreen> without indentation in the kv language.

For adding widgets, we use box layout.

Screen layouts are generally used to set the positions of the widgets on the screen. In the application box layout is used.

box layout: - Widgets can be added sequentially, in either vertically or horizontally.

etc.

In the bottom of the meaning screen button is added to change the screen back to the function screen where all the buttons are added for definition, grammar correction etc.

in the center of the meaning screen label is added to show the result i.e the definition, synonyms, antynonym, plural and singular of the input word.

When the button is pressed in the function screen, is calls the function meaning\_func()

Which is declare in the class App which in return calls the show\_data\_meaning function declare above to show the definition, synonyms, antonyms, plural and singular of the input word from the user. The return is in of string data type as kivy does not accept any other.

Similarly the other functions are added in the application.

In case of translation, the buttons to translate are added in the translate screen, the user has to input the text/word and press the translate button. After this the screen will change to translate screen where there are three button to translate to Hindi, Marathi, English.

In this case we have to use the Noto-Sans regular font as the default font of kivy does not support the glyph of hindi and marathi.

CHAPTER 8

CONCLUSION

**REFERENCES**

1. <https://pypi.org/project/gingerit/>
2. <https://pypi.org/project/PyDictionary/>
3. <https://pypi.org/project/textblob/>
4. <https://kivy.org/doc/stable/>
5. <https://github.com/kivymd/KivyMD/blob/master/kivymd/icon_definitions.py>
6. <https://fonts.google.com/specimen/Noto+Sans#standard-styles>
7. <https://youtu.be/LRXo0juuTrw>