**MINI PROJECT 1**

IDS 566: Advance Text Analytics

Prof. Moontae Lee

College of Business Administration, the University of Illinois at Chicago

by Team 7

Anvesh Rao Nadipelli (651408842)

**Question 2: Open-Ended Project:**

Spellchecker Algorithm:

**The necessary libraries we had imported for the job:**

Graphical user interface, text

Description automatically generated

* Re-package to consider the following to perform [[a-z] will match any lowercase ASCII letter].
* Collections package to use the counter to tokenize the words.
* String to do string operations

**Data and Cleaning:**

We used the “Data.text” file to load our data.

It includes 6488666 characters and 1105285 words.

Normalized all the capitals to small letter words, ignored the numbers, symbols, and punctuations present in the Data, and created all the set strings containing alphabets using the following function.

Logo, company name

Description automatically generated

Output for Vocab Example

Graphical user interface, text

Description automatically generated

**Tokenization of the strings:**

Loaded our data set into input variable using vocab function.

The following code can obtain the number of words in the file:

A picture containing graphical user interface

Description automatically generated

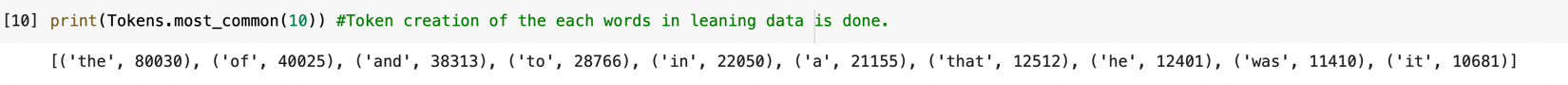
Now from NLP, tokenization of all the words present in the vocab done by the following function:

Graphical user interface, text, application, chat or text message

Description automatically generated

Resulting in Dictionary with words as keys and tokens (number of times a word appeared in the learning data) as items.

The most common words in the data:



**Autocorrect and Spell Correction Model:**

Our Autocorrect model takes the text data file to construct a dictionary of words, assuming those words exist in English language.

It also determines the likelihood of the word occurrence in English text.

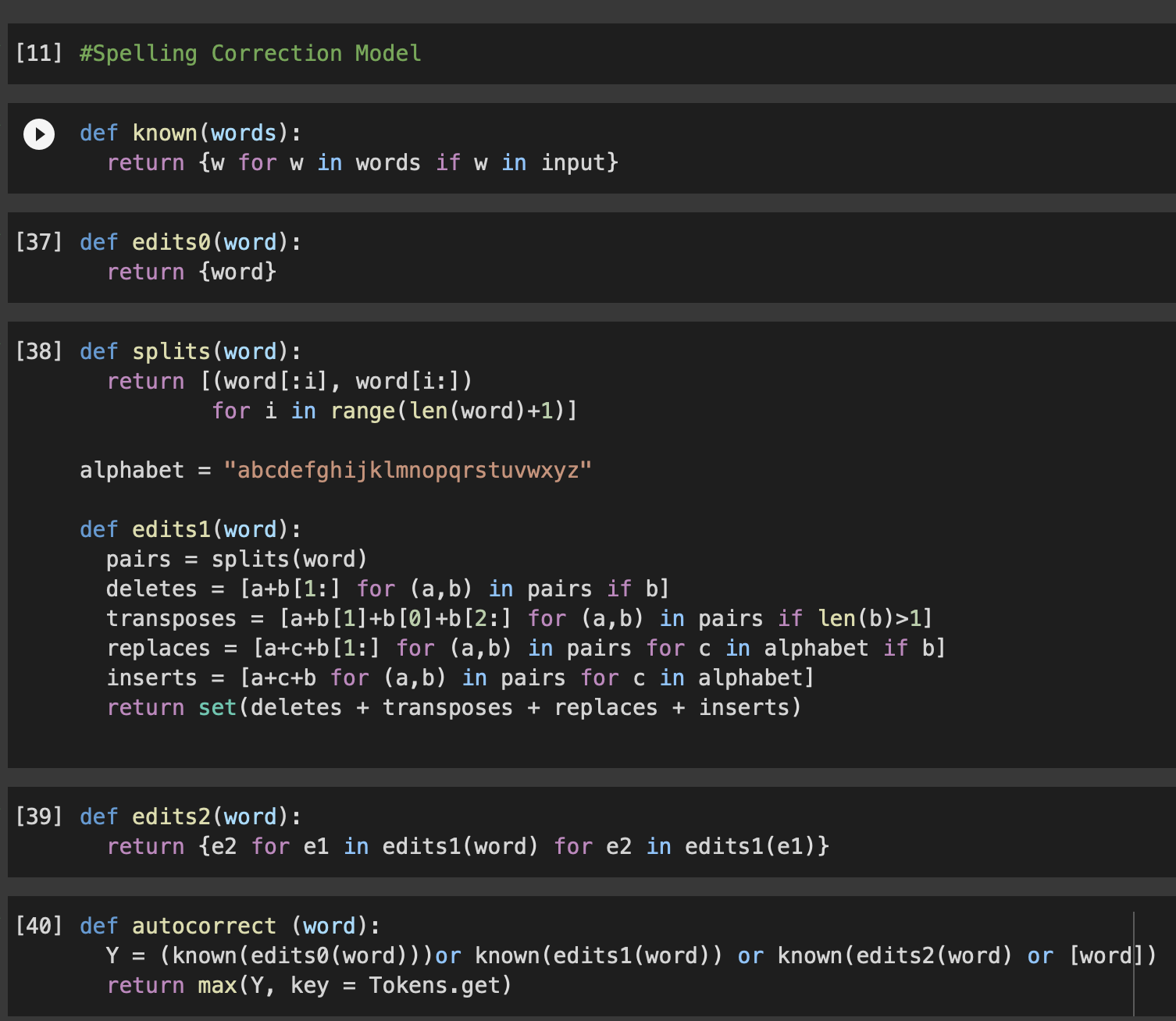
When an inaccurately spelled word is passed to the autocorrection function, the algorithm generates an edit distance 1 version of it by doing one of the following operations:

Delete, transpose, replace, or insert.

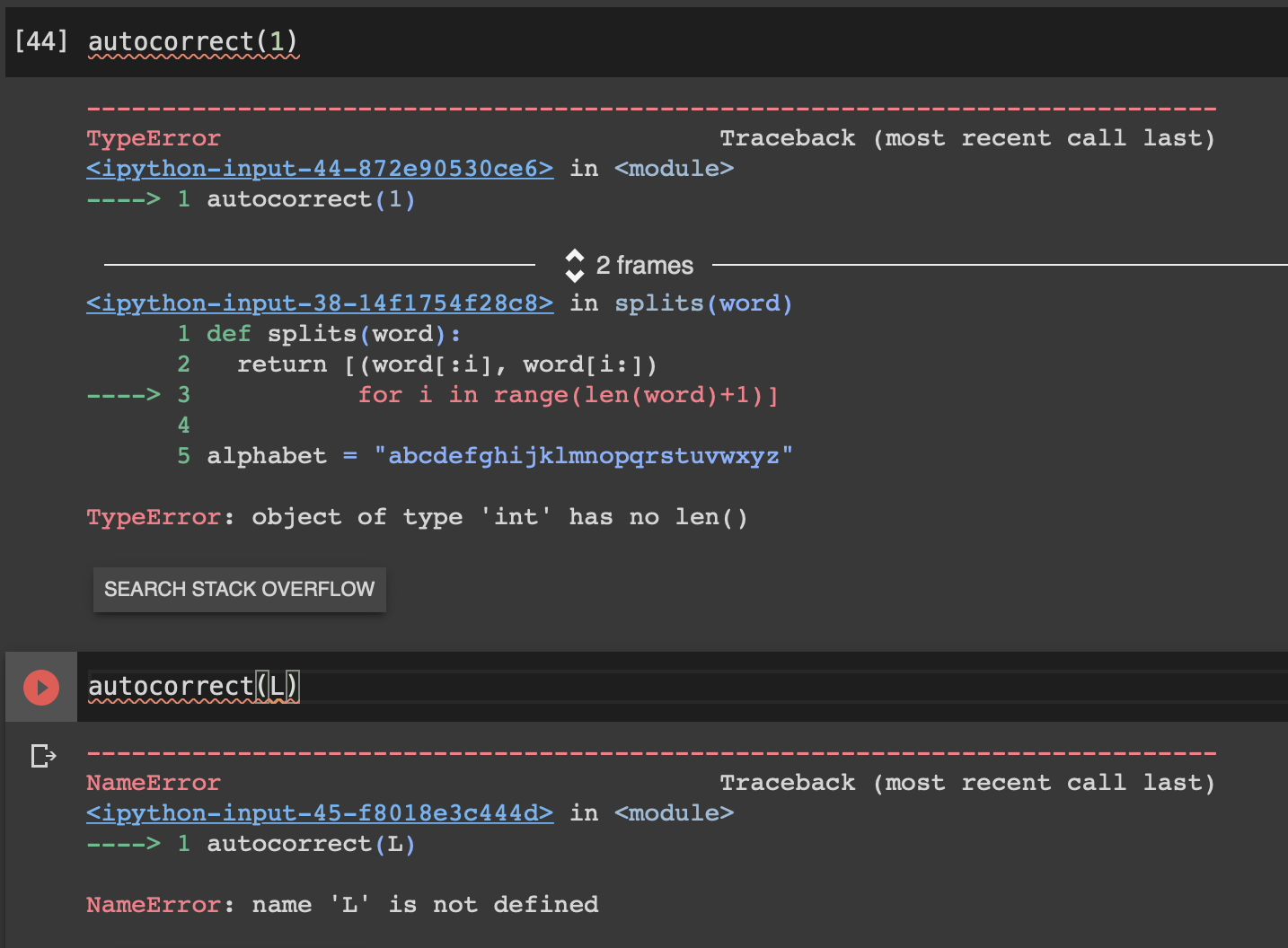
Similarly, it just uses the function on the created words again to get a list of words with an edit distance of 2.

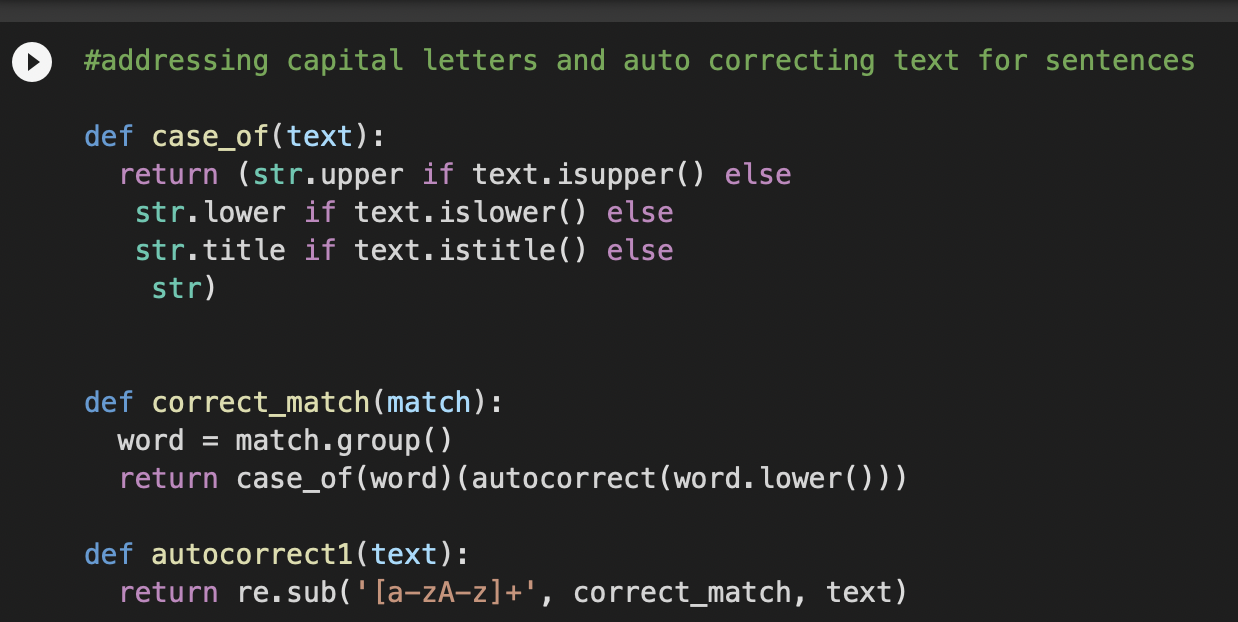
The next step is to select the most appropriate term from the created list of corrected words; this is done by using the probability of each word occurring from the previously saved probabilities.

For example, if it had to choose between 'the' and 'then,' it would prefer 'the' because it occurs more frequently in text.

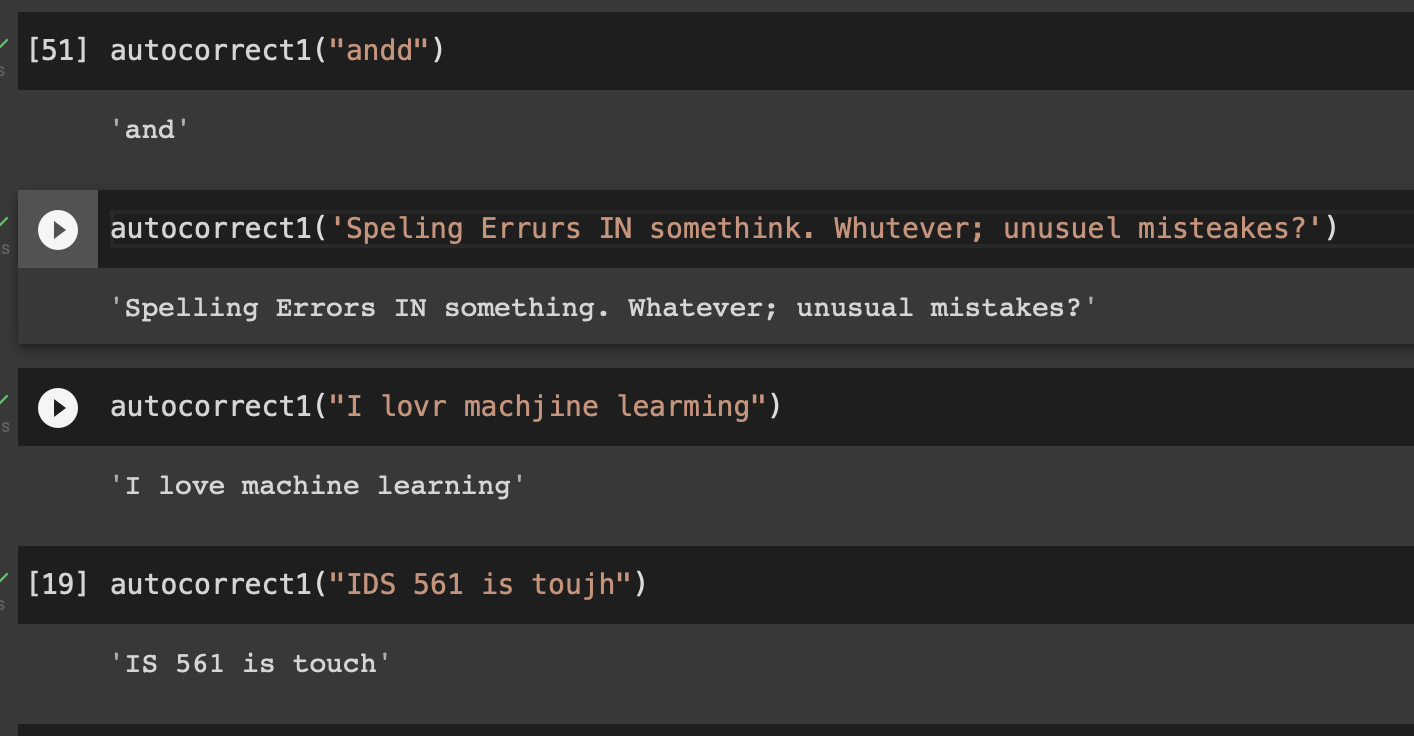


The problem with the above autocorrect function is that it will show an error whenever we use capital letter words, special characters, and numbers.



To handle this problem, we defined the following functions, which handled special characters, all types of alphabets (both capital and small), and numbers.

**Results for autocorrect1 Function:**



The one problem with this model will be, it’s output will always be from the learnt data. Words outside of the learnt data, will be corrected to words present in the learnt data.

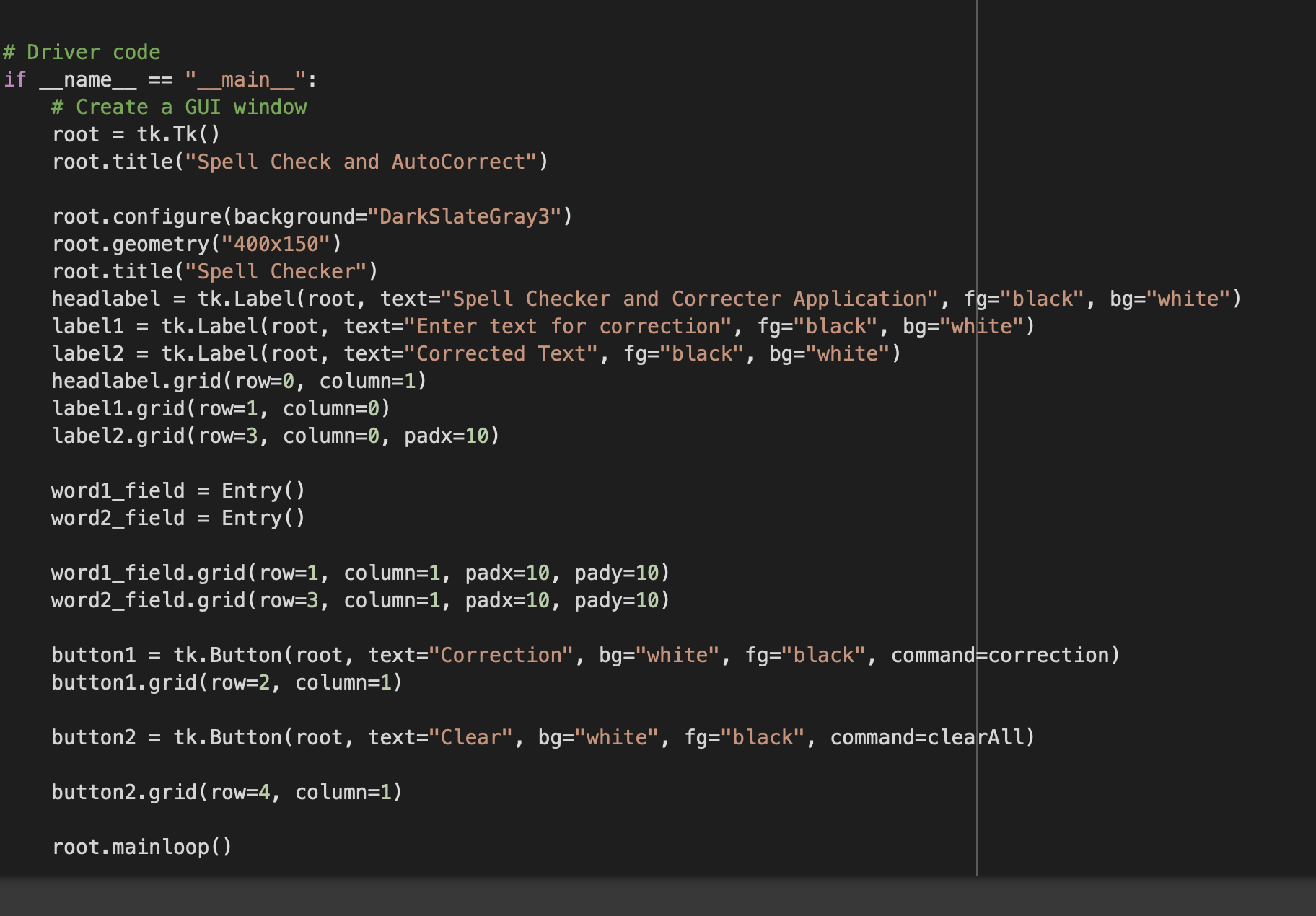
Eg [19] IDS is converted to IS. Tough is converted to touch.

**User interface**

We had used below python packages to generate the GUI Interface from python.

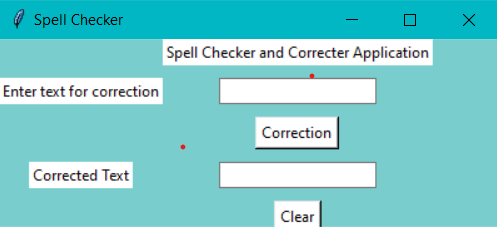
**“The below code needs to be runned on PyCharm to a local machine to develop an GUI”.**

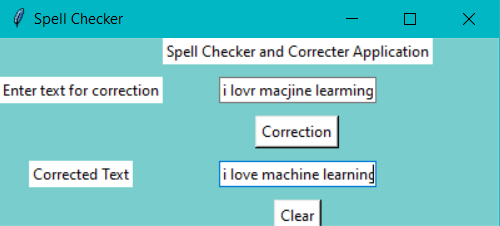
We had used tinkter packages to develop the GUI. Main driver code for the GUI Window:

****

**W**

Output:

****

**