Presentation by

**Group 6**

VIT Bhopal

Computational Linguistics

# Part-of-Speech (POS) Identification and Tagging

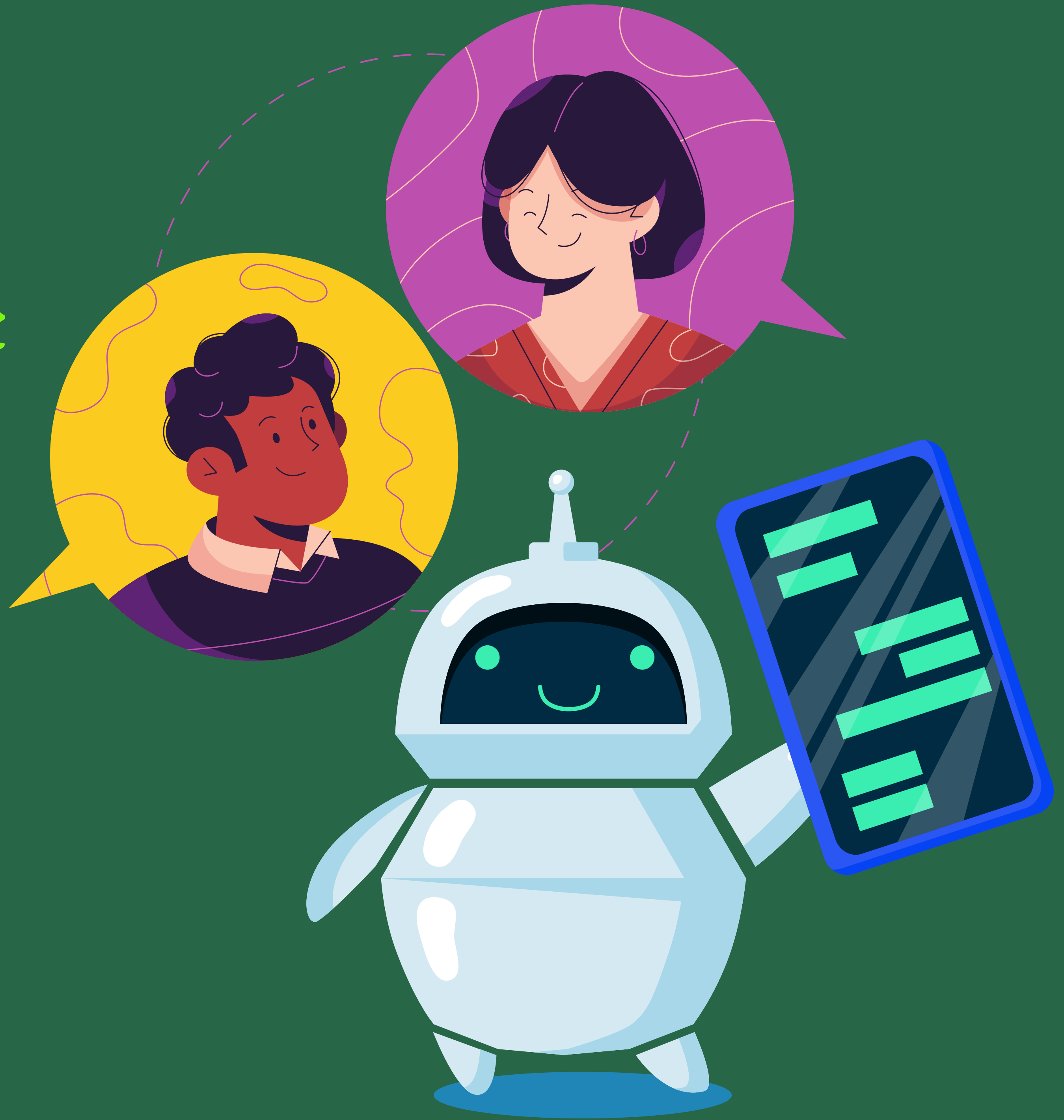
- 21BAI10023 Harshit Singh
- 21BAI10031 Sarath Rajendran
- 21BAI10067 Vansh Raja
- 21BAI10095 Ashish Sutar
- 21BAI10306 Mohini Tiwari
- 21BAI10393 Jigyashu Saxena
- 21BAI10394 Kunal Paliwal
- 21BCE10421 Ayush Johari

## Intro duction

NLP is an automatic approach to analyzing texts using a different set of technologies and theories with the help of a computer. It is also defined as a computerized approach to process and understand natural language. Thus, it improves human-to-human communication, enables human-to-machine communication by doing useful processing of texts or speeches.

Part-of-speech (POS) tagging is one of the most important addressed areas and main building block and application in the natural language processing discipline.

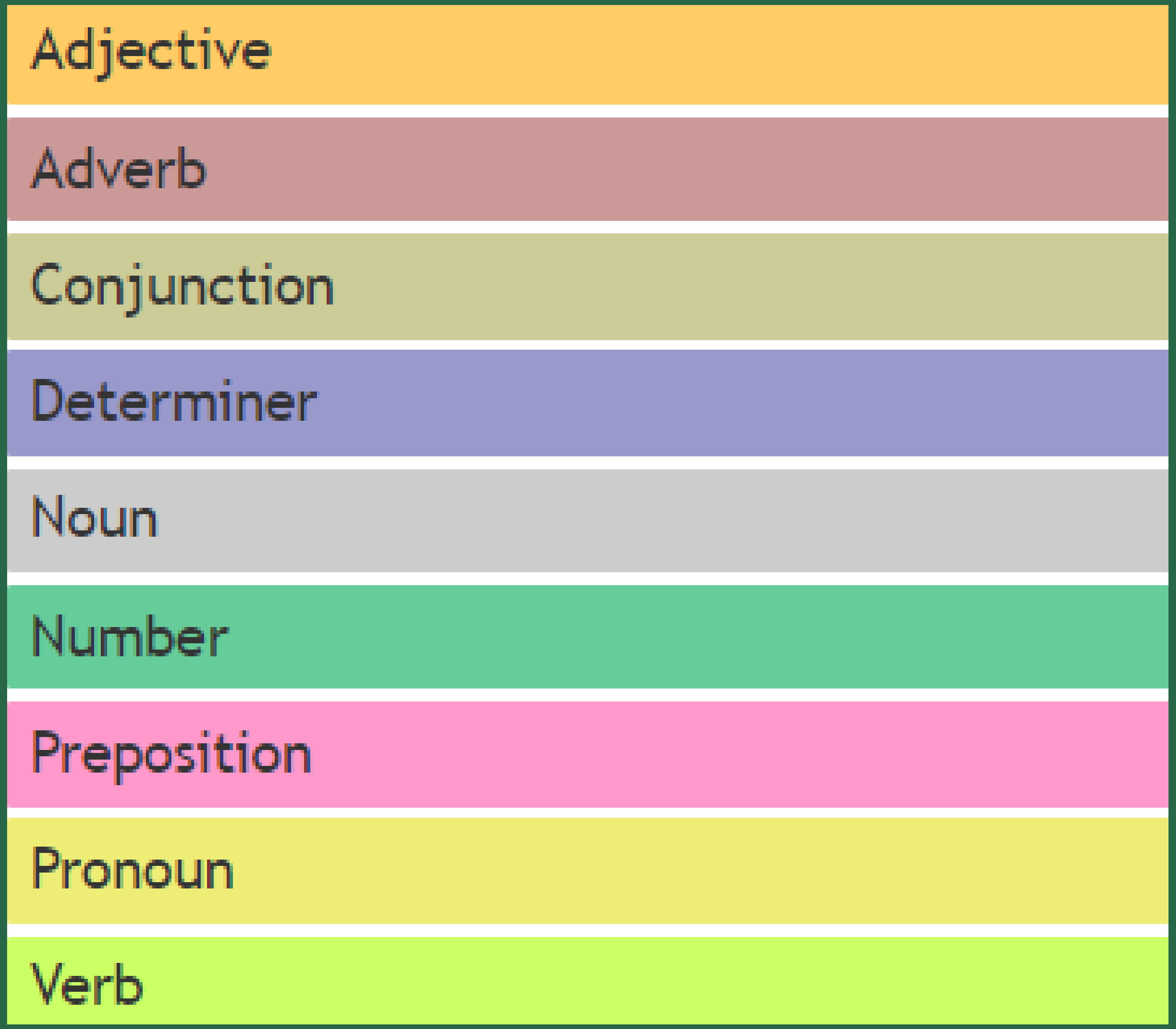
# POS Tagging



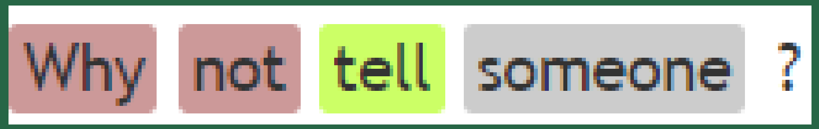
Part of Speech (POS) Tagging is a notable NLP topic that aims in assigning each word of a text the proper syntactic tag in its context of appearance. A POS is a grammatical classification that commonly includes verbs, adjectives, adverbs, nouns, etc. POS tagging is an important natural language processing application used in machine translation, word sense disambiguation, question answering parsing, and so on. The genesis of POS tagging is based on the ambiguity of many words in terms of their part of speech in a context.

# Tag sets

A set of all POS tags used in a corpus is called a tagset. Commonly, there are 9 parts of speech in English:
However, there are clearly many more categories and sub-categories. For nouns, the plural, possessive, and singular forms can be distinguished.
In part-of-speech tagging by computer, it is typical to distinguish from 50 to 150 separate parts of speech for English.
The most popular "tag set" for POS tagging for American English is probably the Penn tag set.
In our case, as our mother tongue is Hindi, the tag set we have picked up is "Hindi" from "Indian corpus"



Adjective
Adverb
Conjunction
Determiner
Noun
Number
Preposition
Pronoun
Verb



Why not tell someone ?

# Benefits of POS Tagging

01

**Improved natural language understanding:** POS tagging helps to identify the role of each word in a sentence, which can improve the overall understanding of the meaning of the sentence.

02

**Text classification:** POS tagging can be used as a feature for text classification tasks, such as sentiment analysis and topic classification.

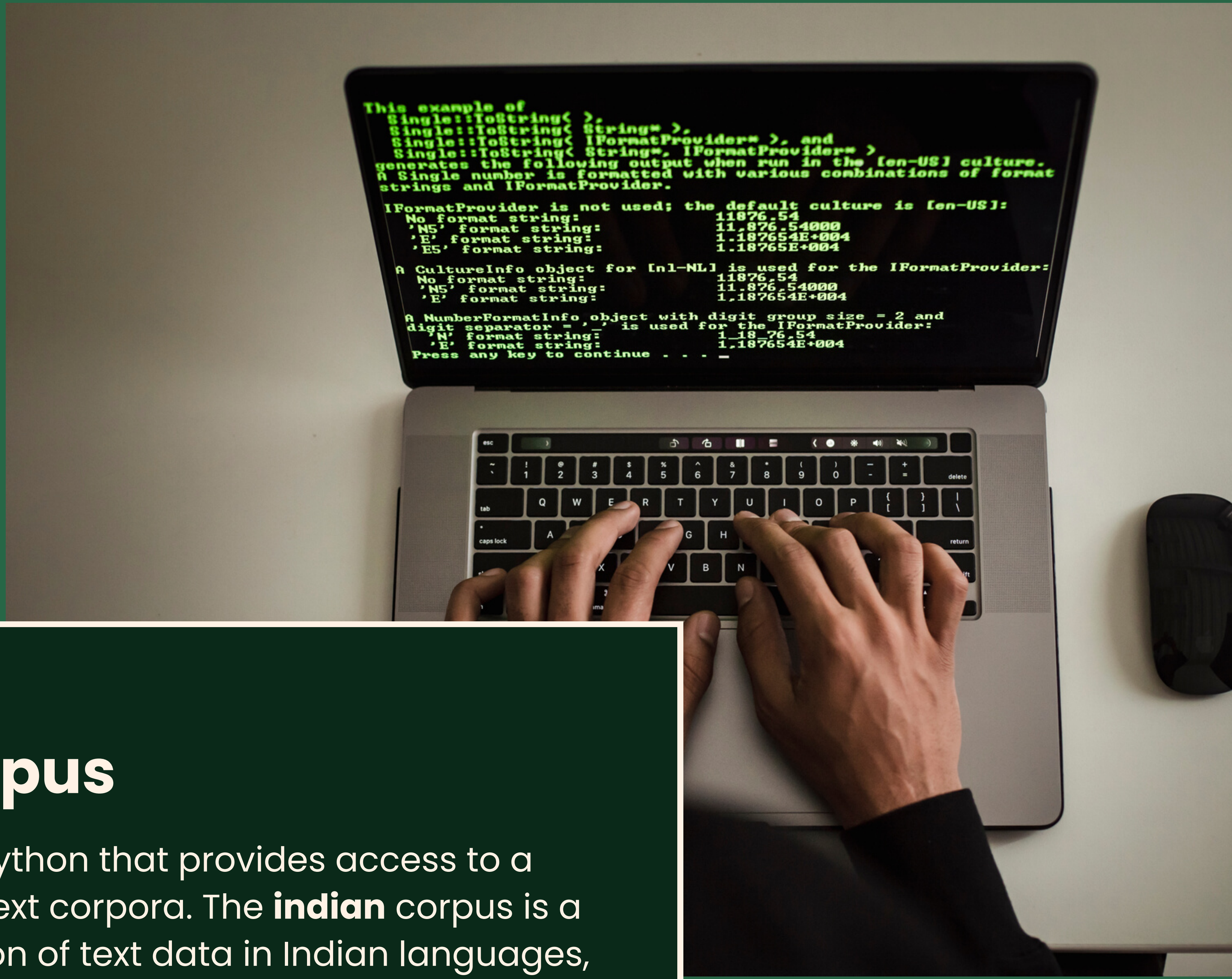
03

**Named entity recognition:** POS tagging can be used to identify proper nouns and other entities, which can be used for tasks such as named entity recognition and information extraction.

04

**Language generation:** POS tagging can be used as a guide for language generation tasks, such as text summarization and text completion, by identifying the important words and phrases in a sentence.

# Libraries Used



```
This example of
  Single::ToString( ),
  Single::ToString( String* ),
  Single::ToString( IFormatProvider* ), and
  Single::ToString( String*, IFormatProvider* )
generates the following output when run in the [en-US] culture.
A Single number is formatted with various combinations of format
strings and IFormatProvider.

IFormatProvider is not used; the default culture is [en-US]:
  No format string:                  11876.54
  'N5' format string:                11,876.54000
  'E' format string:                 1.187654E+004
  'E5' format string:                1.18765E+004

A CultureInfo object for [nl-NL] is used for the IFormatProvider:
  No format string:                  11876,54
  'N5' format string:                11.876,54000
  'E' format string:                 1,187654E+004

A NumberFormatInfo object with digit group size = 2 and
digit separator = '_' is used for the IFormatProvider:
  'N' format string:                 1_18_76.54
  'E' format string:                 1.187654E+004
Press any key to continue . . . _
```

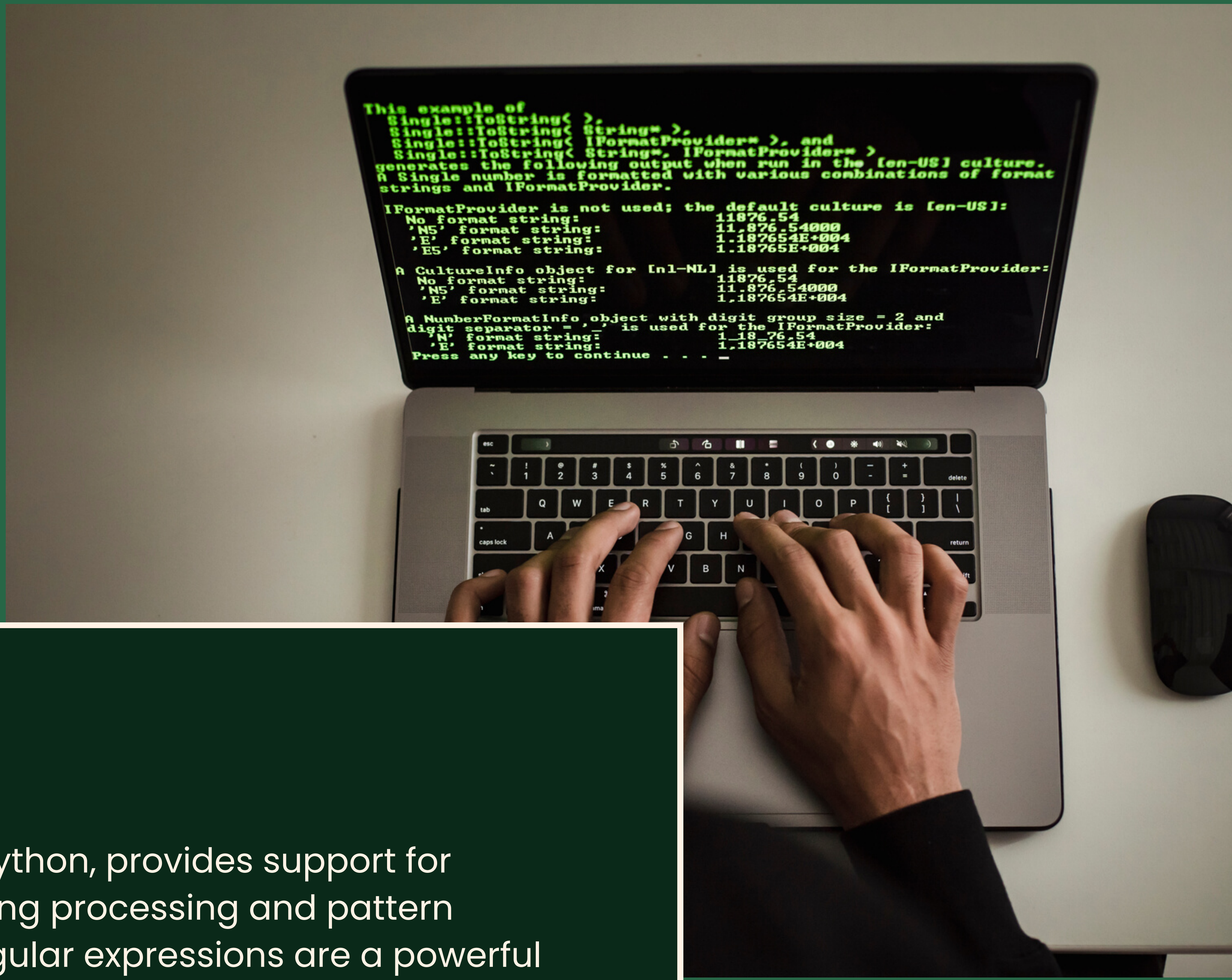
## nltk.tag

A module in Python. It is part of the Natural Language Toolkit (NLTK). It provides functions for tagging words in a text with their part of speech (POS). The **Tnt** tagger is a memory-based tagger that uses a trigram-based approach to POS tagging.

## nltk.corpus

A module in Python that provides access to a collection of text corpora. The **indian** corpus is a small collection of text data in Indian languages, including Hindi, Marathi, and Bengali.

# Libraries Used



```
This example of
  Single::ToString( ),
  Single::ToString( String* ),
  Single::ToString( IFormatProvider* ), and
  Single::ToString( String*, IFormatProvider* )
generates the following output when run in the [en-US] culture.
A Single number is formatted with various combinations of format
strings and IFormatProvider.

IFormatProvider is not used; the default culture is [en-US]:
  No format string:                     11876.54
  'N5' format string:                   11,876.54000
  'E' format string:                    1.187654E+004
  'E5' format string:                   1.18765E+004

A CultureInfo object for [nl-NL] is used for the IFormatProvider:
  No format string:                     11876,54
  'N5' format string:                   11.876,54000
  'E' format string:                    1,187654E+004

A NumberFormatInfo object with digit group size = 2 and
digit separator = '_' is used for the IFormatProvider:
  'N' format string:                    1_18_76,54
  'E' format string:                    1,187654E+004
Press any key to continue . . . _
```

## GoogleTrans

A free and unlimited Python library that implements Google Translate API. It allows developers to translate text from one language to another language.

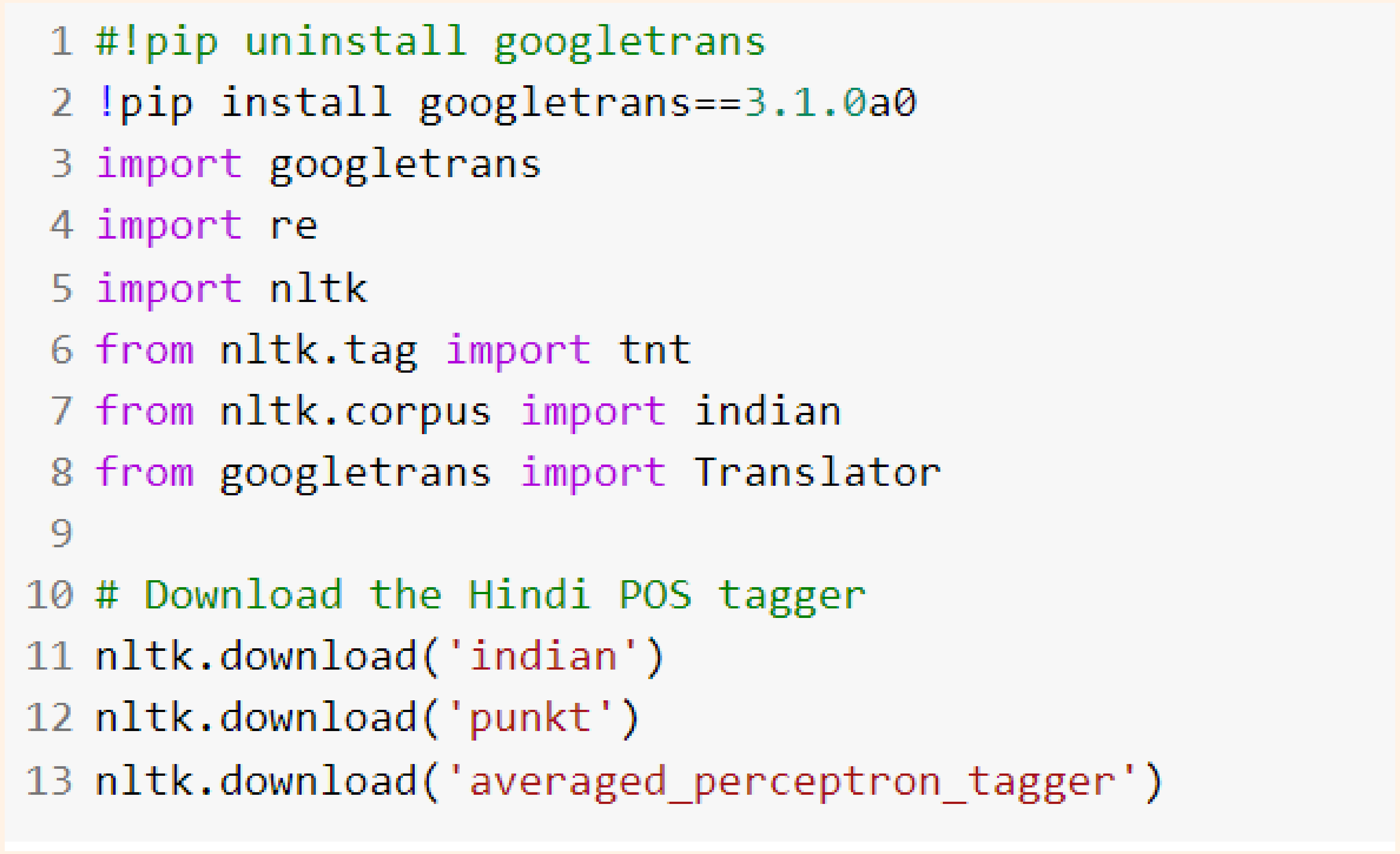
## Re

A module in Python, provides support for advanced string processing and pattern matching. Regular expressions are a powerful tool for text manipulation and can be used to search, extract, and modify text based on specific patterns.

# The Code

## Libraries in our Code

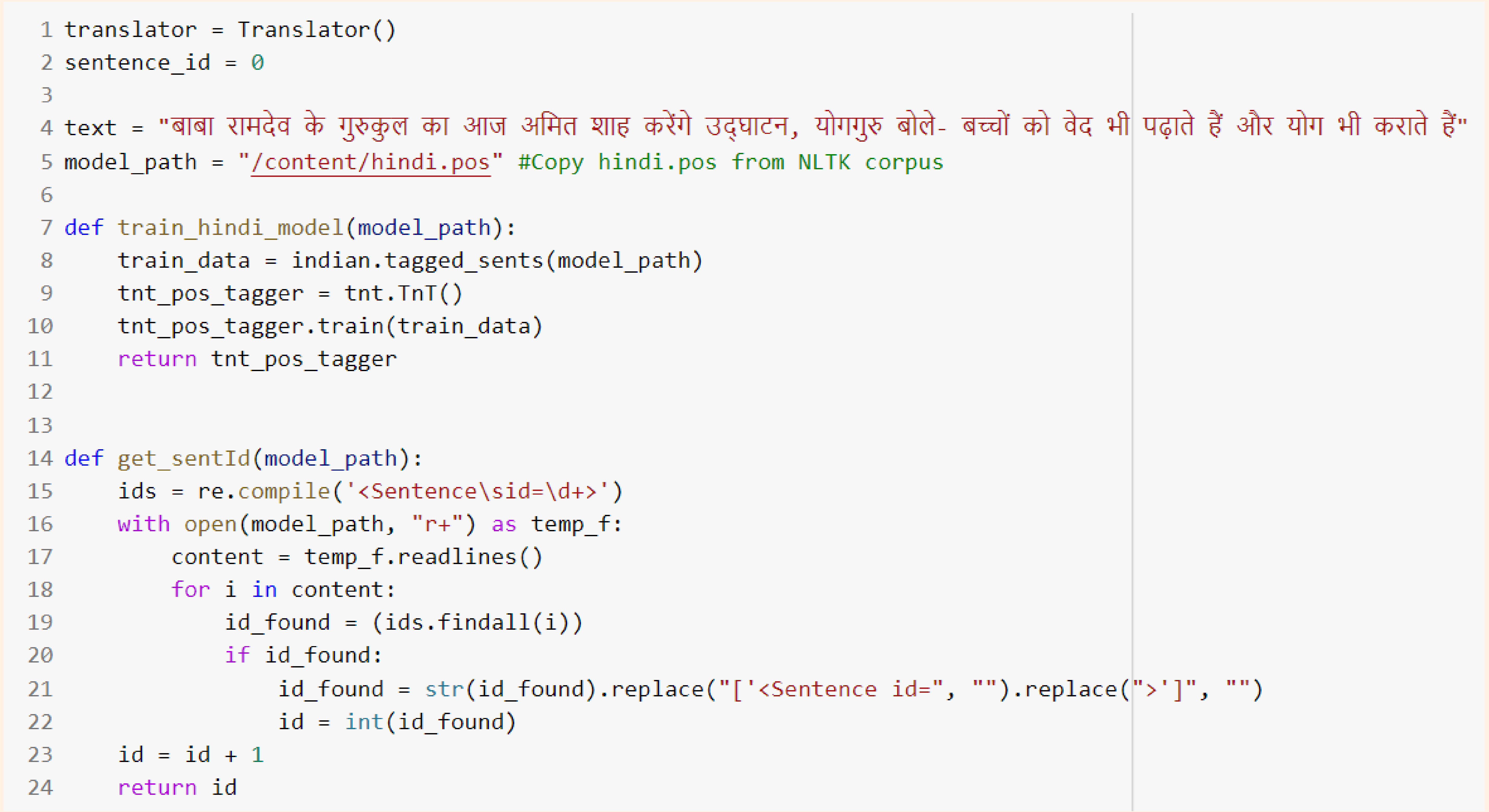
The beginning of the code is to install and download the necessary packages for part-of-speech tagging in Hindi. Firstly, the googletrans library is installed and imported. Then, the Indian, Punkt, and Averaged Perceptron Tagger packages are downloaded from the NLTK library. This will allow the program to use the Hindi POS tagger.



```python
#!pip uninstall googletrans
!pip install googletrans==3.1.0a0
import googletrans
import re
import nltk
from nltk.tag import tnt
from nltk.corpus import indian
from googletrans import Translator

# Download the Hindi POS tagger
nltk.download('indian')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

## Code

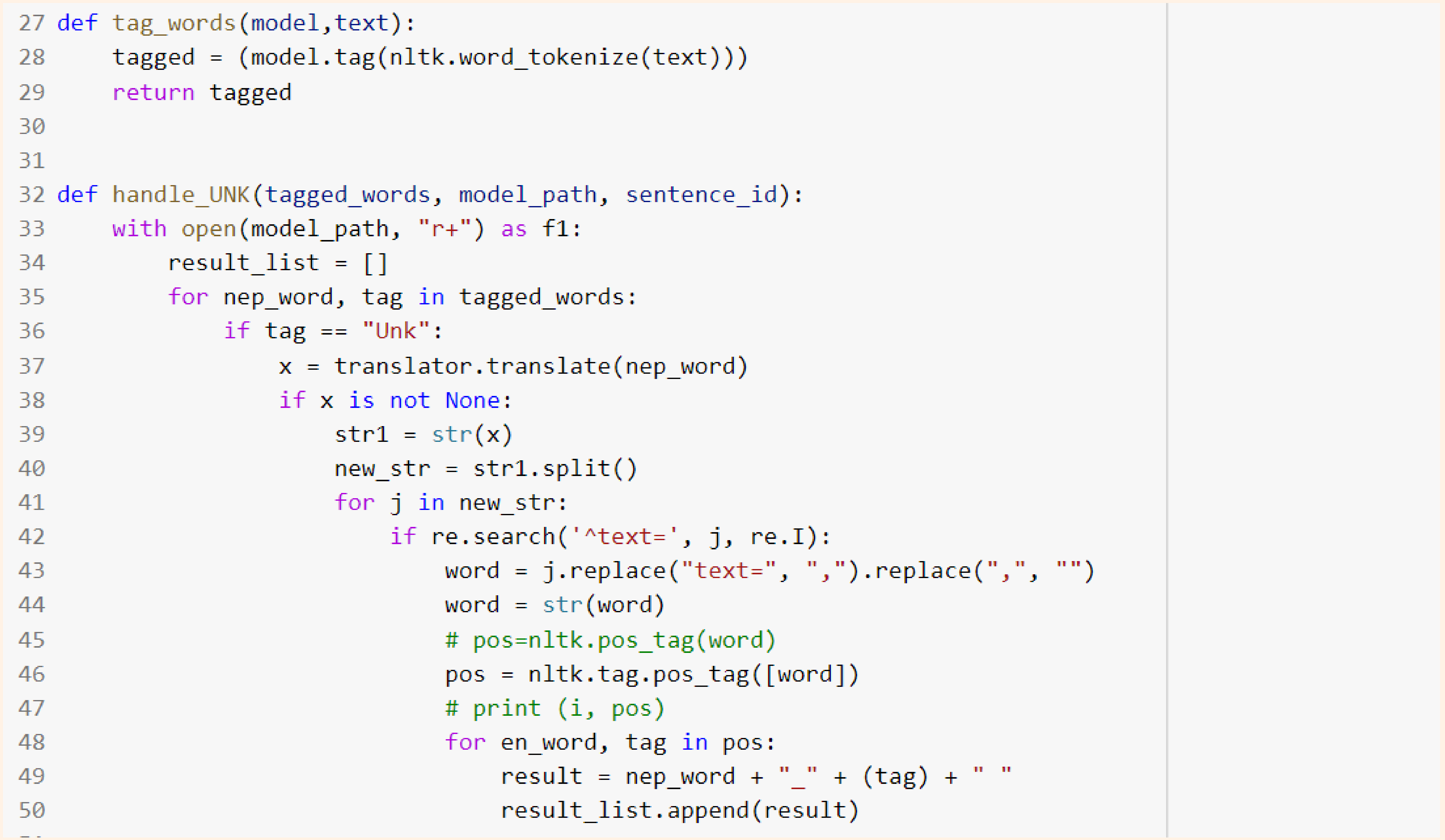
The code has used the googletrans and nltk libraries to download and install necessary packages. It utilizes functions such as indian.tagged_sents() and tnt.TnT() to train the tagger and extract sentence IDs from the collection. This will help the program identify the parts of speech of words in Hindi sentences.



```python
translator = Translator()
sentence_id = 0

text = "बाबा रामदेव के गुरुकुल का आज अमित शाह करेंगे उद्घाटन, योगगुरु बोले- बच्चों को वेद भी पढ़ाते हैं और योग भी कराते हैं"
model_path = "/content/hindi.pos" #Copy hindi.pos from NLTK corpus

def train_hindi_model(model_path):
    train_data = indian.tagged_sents(model_path)
    tnt_pos_tagger = tnt.TnT()
    tnt_pos_tagger.train(train_data)
    return tnt_pos_tagger


def get_sentId(model_path):
    ids = re.compile('<Sentence\sid=\d+>')
    with open(model_path, "r+") as temp_f:
        content = temp_f.readlines()
        for i in content:
            id_found = (ids.findall(i))
            if id_found:
                id_found = str(id_found).replace("['<Sentence id=", "").replace(">']", "")
                id = int(id_found)
    id = id + 1
    return id
```

## Code

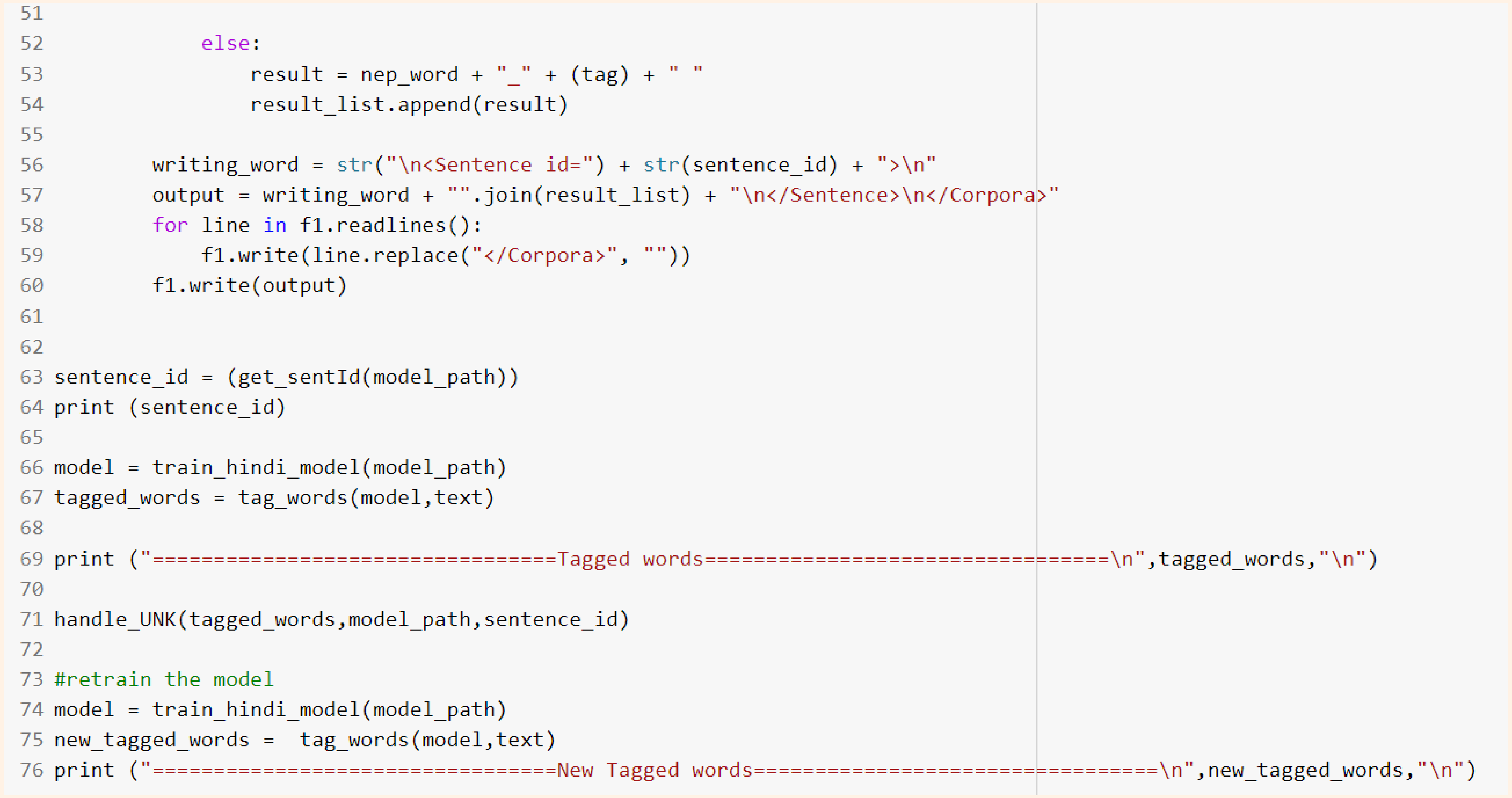
This code is used to tag words in a given text with their part of speech (POS) tags. It takes a model, text, and sentence ID as input, and handles unknown words by translating them and then tagging them with the POS tag. It then stores the results in a list and returns the list. It uses functions tag_words and handle_UNK for it.



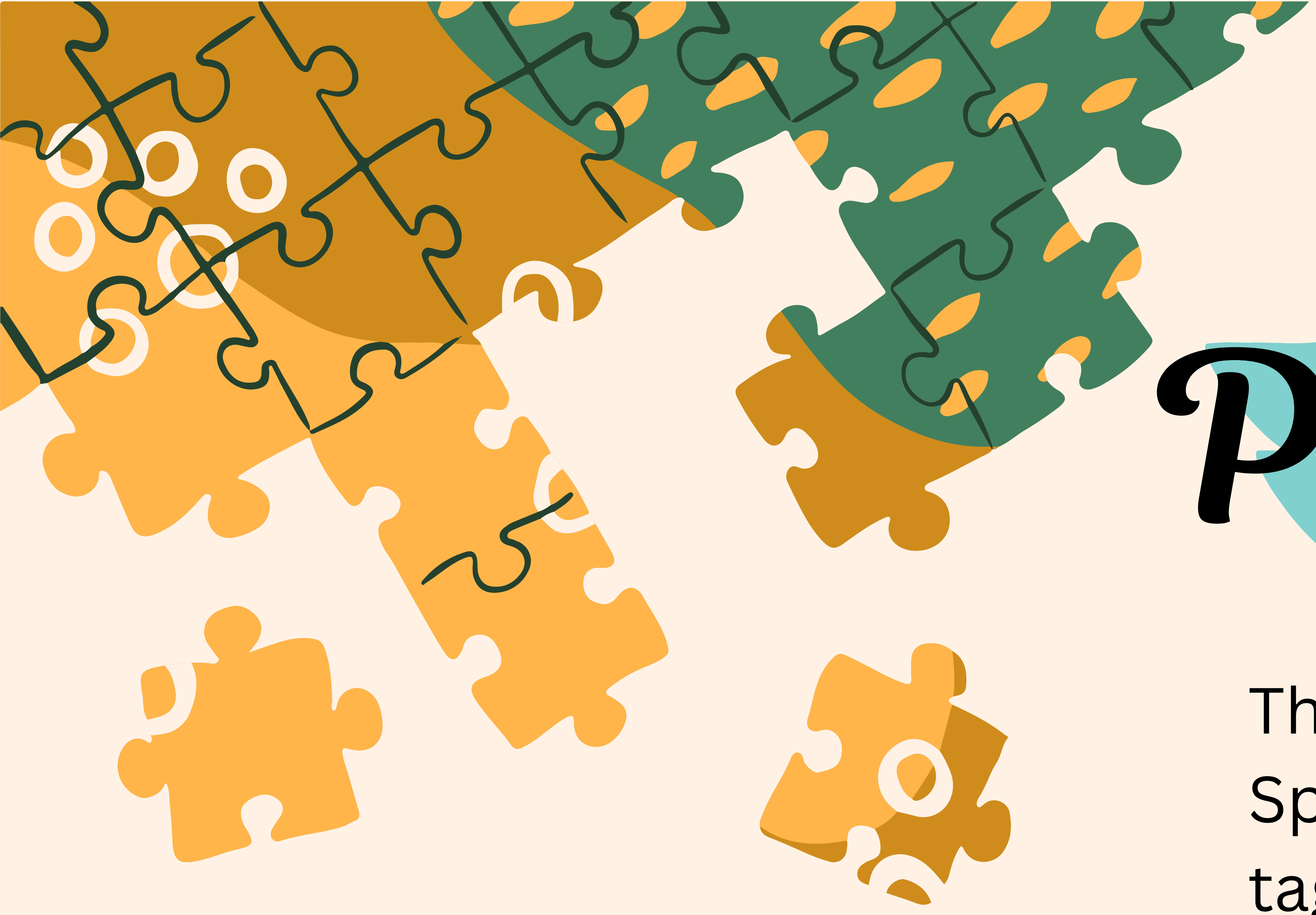
```python
def tag_words(model,text):
    tagged = (model.tag(nltk.word_tokenize(text)))
    return tagged


def handle_UNK(tagged_words, model_path, sentence_id):
    with open(model_path, "r+") as f1:
        result_list = []
        for nep_word, tag in tagged_words:
            if tag == "Unk":
                x = translator.translate(nep_word)
                if x is not None:
                    str1 = str(x)
                    new_str = str1.split()
                    for j in new_str:
                        if re.search('^text=', j, re.I):
                            word = j.replace("text=", ",").replace(",", "")
                            word = str(word)
                            # pos=nltk.pos_tag(word)
                            pos = nltk.tag.pos_tag([word])
                            # print (i, pos)
                            for en_word, tag in pos:
                                result = nep_word + "_" + (tag) + " "
                                result_list.append(result)
```

## Code

Then, the code handles unknown words in a given text. It does that by translating them and then tagging them with the POS tag. It takes a model path, text, and sentence ID as input. It retrains the model and tags the words again, returning the new list of tagged words.



```

            else:
                result = nep_word + "_" + (tag) + " "
                result_list.append(result)

        writing_word = str("\n<Sentence id=") + str(sentence_id) + ">\n"
        output = writing_word + "".join(result_list) + "\n</Sentence>\n</Corpora>"
        for line in f1.readlines():
            f1.write(line.replace("</Corpora>", ""))
        f1.write(output)


sentence_id = (get_sentId(model_path))
print (sentence_id)

model = train_hindi_model(model_path)
tagged_words = tag_words(model,text)

print ("=================================Tagged words=================================\n",tagged_words,"\n")

handle_UNK(tagged_words,model_path,sentence_id)

#retrain the model
model = train_hindi_model(model_path)
new_tagged_words =  tag_words(model,text)
print ("=================================New Tagged words=================================\n",new_tagged_words,"\n")
```



## Process Flow

The flow of processing for Part-of-Speech (POS) identification and tagging can be summarized as follows:

1

Train the POS Tagger on the POS Model

Tokenize every word and print their output along with POS Tag

2

3

Translate to handle unknown tags

Train the tagger ; tokenize again on the new model and show its output

4

# Tags used in Demo

The tagset that we have see in class i.e Penn TreeBank Tagset cannot be used to properly tag indian languages, so the tagset we have is the Indian/Hindi Tagset from the NLTK library

- **Noun (NN)**
- **Proper Noun (NNP)**
- **Pronoun (PRP)**
- **Adjective (JJ)**
- **Verb (VB)**
- **Adverb (RB)**
- **Preposition (IN)**
- **Numeral (CD)**
- **Determiner (DT)**
- **Adjectival Noun (JJ NN)**
- **Infinitive (VBN)**
- **Gerund (VBG)**
- **Participle (VBD)**
- **Proper Noun Singular (NNPS)**
- **Proper Noun Plural (NNS)**
- **Conjunct (CONJ)**
- **Expletive (INTJ)**
- **Adverb-Comparative (RBR)**
- **Adverb-Superlative (RBS)**
- **Wh-Determiner (WDT)**
- **Wh-Pronoun (WP)**
- **Wh-Adverb (WRB)**
- **Conjunction (CC)**
- **Interjection (UH)**
- **Particle (RP)**
- **Quantifier (QF)**
- **Quantifier-Fractional Number (QFNUM)**
- **Symbol (SYM)**
- **Postposition (POS)**

## Demonstration

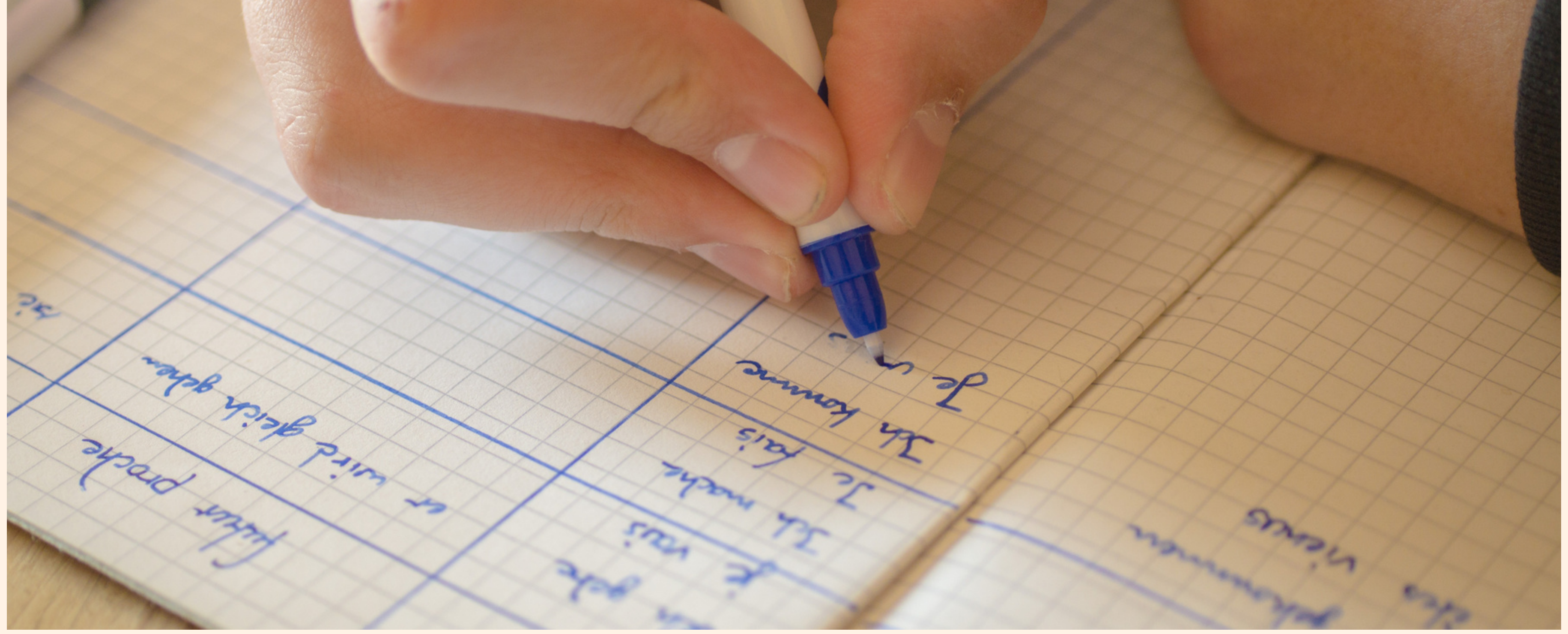
The team will be conducting a live demonstration of the functioning code.

### Statement 1

गर्मी के एक दिन में, जंगल के एक शेर को बहुत जोरों से भूख लगी. इसलिए वो इधर उधर खाने की तलाश करने लगा. कुछ देर खोजने के बाद उसे एक खरगोश मिला, लेकिन उसे खाने के बदले में उसे उसने छोड़ दिया क्यूंकि उसे वो बहुत ही छोटा लगा

### Statement 2

बहुत पुराने समय की बात है, एक राजा ने जानबूझकर एक बड़ा सा चट्टान रास्ते के बीचों बीच में रखवा दिया। वहीं वो पास के एक बड़े से झाड़ी में छुप गया। वो ये देखना चाहता था की आख़िर कौन वो चट्टान रास्ते से हटाता है



# Conclusion

## The Importance of POS Tagging in NLP

POS tagging is an essential technique for natural language processing, with wide-ranging applications in various fields such as natural language understanding, text classification, named entity recognition, and language generation.

## The Future of POS Tagging in Understanding Text Data

As the field of NLP continues to grow, the use of POS tagging will become increasingly important in understanding and making sense of the vast amounts of text data generated every day, making it a fundamental step in the process of comprehending human language.

Presentation by

**Group 6**

VIT Bhopal

Computational Linguistics

# Thank You!