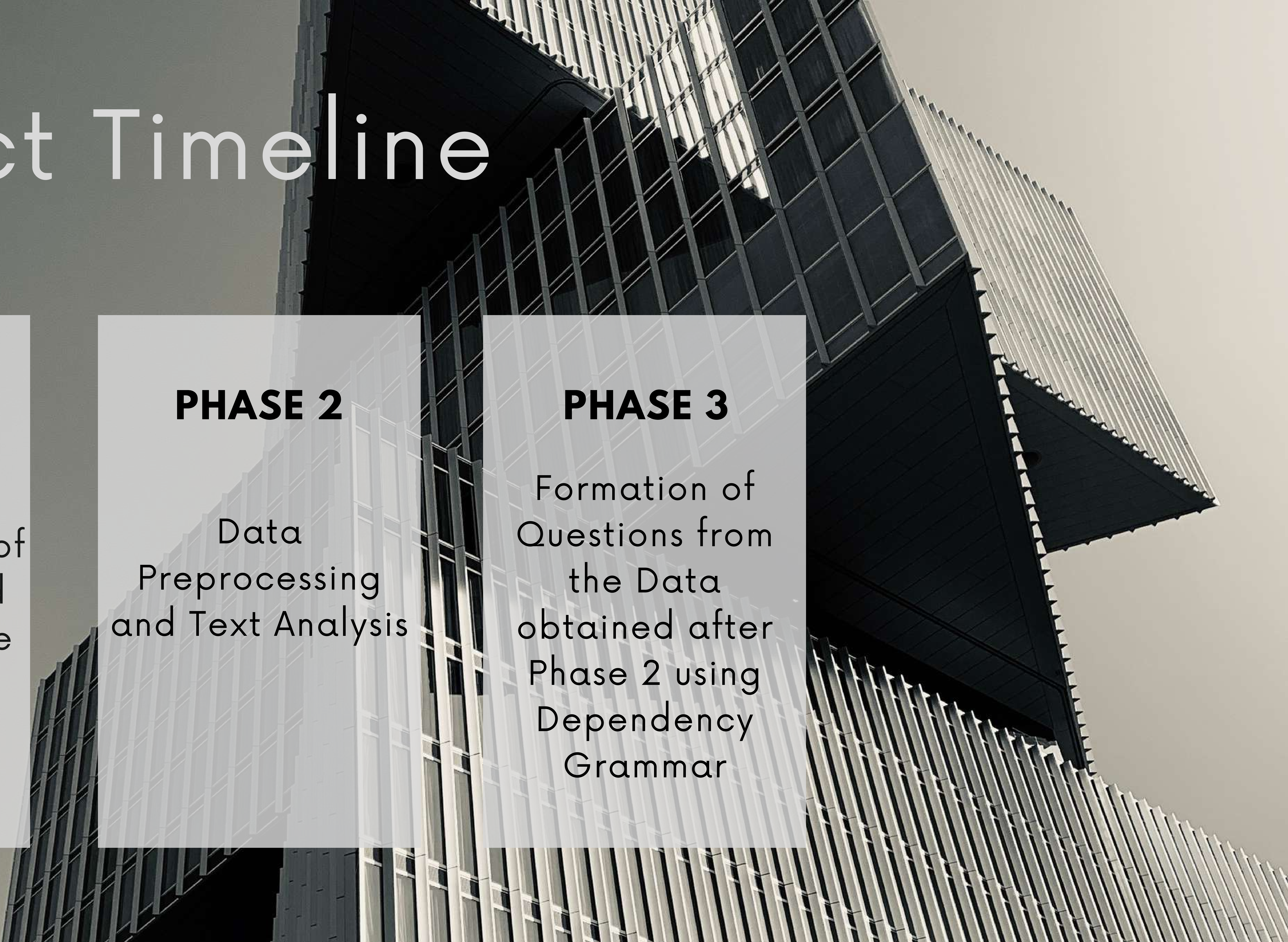
Question Generation from Fable Stories

Team - DoNut Give Up

Gargi Shroff
Swarang Joshi
Ketaki Shetye

Project Timeline



PHASE 1

Presenting the general outline of the project and procedure to be followed.

PHASE 2

Data Preprocessing and Text Analysis

PHASE 3

Formation of Questions from the Data obtained after Phase 2 using Dependency Grammar

Data Pre-Processing

Steps involved in preprocessing are as follows -

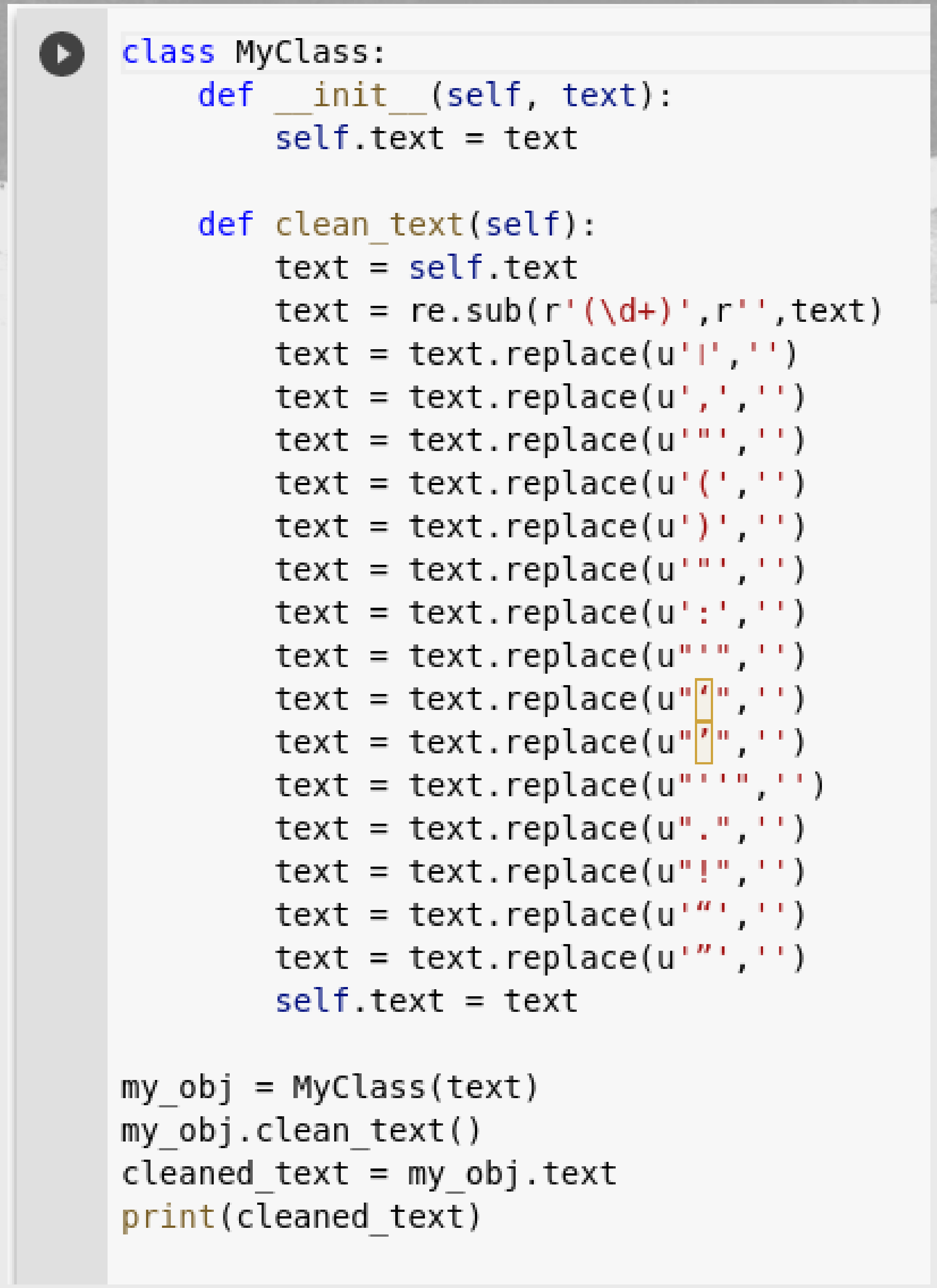
1. Scraping the required text from the collected Dataset.
   Link of the resource for the Dataset - https://www.hindisahityadarpan.in/2016/06/panchatantra-complete-stories hindi.htm

2. Removing the Punctuation Marks from the Dataset.

3. POS Tagging of the obtained Dataset

4. Dependency Parsing of the Dataset

Removing Punctuation Marks



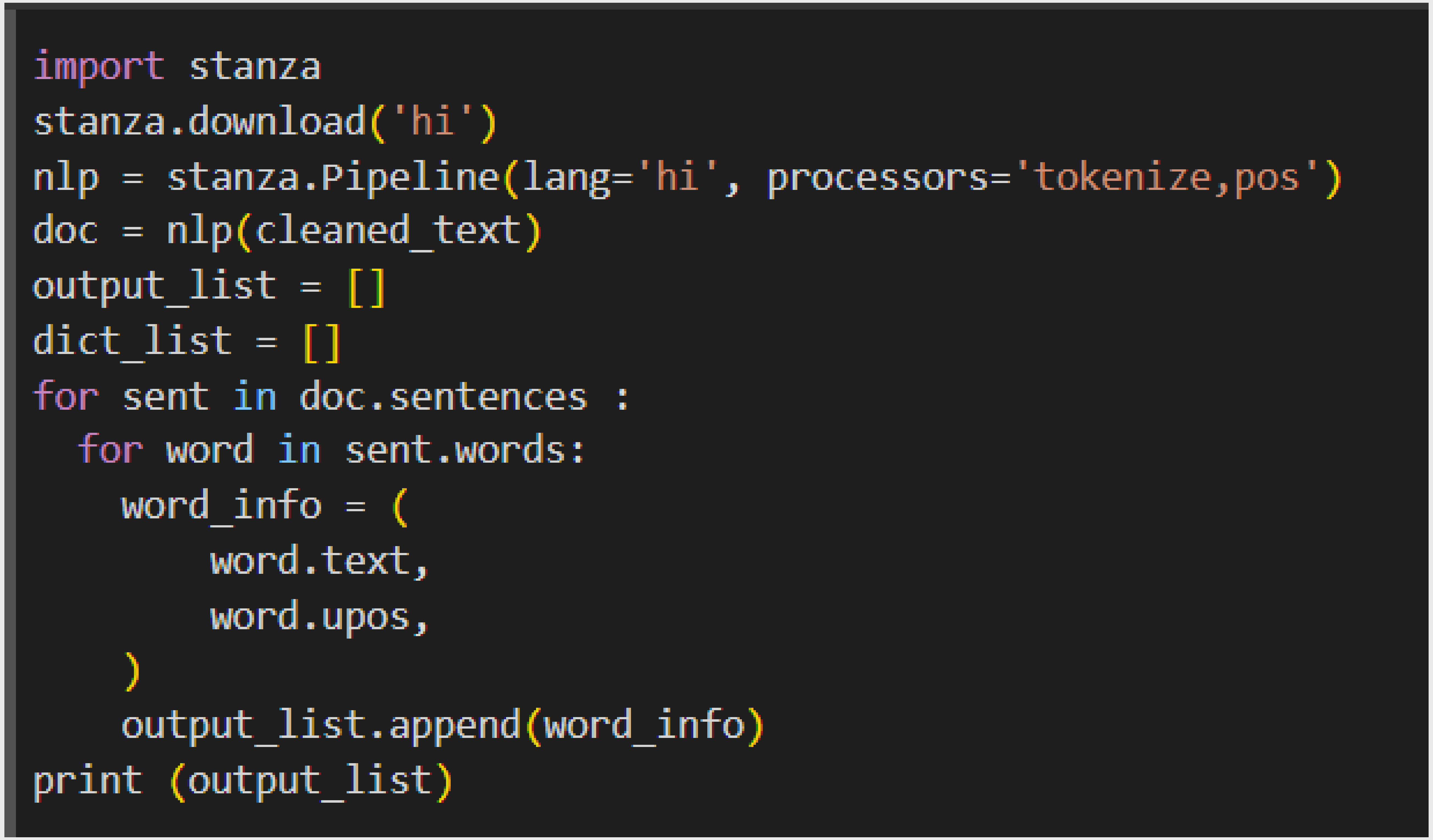
```
class MyClass:
    def __init__(self, text):
        self.text = text

    def clean_text(self):
        text = self.text
        text = re.sub(r'(\d+)',r'',text)
        text = text.replace(u'|','')
        text = text.replace(u',','')
        text = text.replace(u'"','')
        text = text.replace(u'(','')
        text = text.replace(u')','')
        text = text.replace(u'"','')
        text = text.replace(u':','')
        text = text.replace(u"'",'')
        text = text.replace(u"‘",'')
        text = text.replace(u"’",'')
        text = text.replace(u"''",'')
        text = text.replace(u".",'')
        text = text.replace(u"!",'')
        text = text.replace(u'“','')
        text = text.replace(u'”','')
        self.text = text

my_obj = MyClass(text)
my_obj.clean_text()
cleaned_text = my_obj.text
print(cleaned_text)
```

Punctuation marks are not always used consistently across different types of text data. For example, the use of commas, periods, and other punctuation marks can vary depending on the writer or the context. Removing punctuation marks helps to standardize the text data and makes it easier to compare and analyse different datasets. **However we are not removing the punctuation marks | , which marks the end of sentence in Hindi language, so as to keep the track where the sentence ends.**

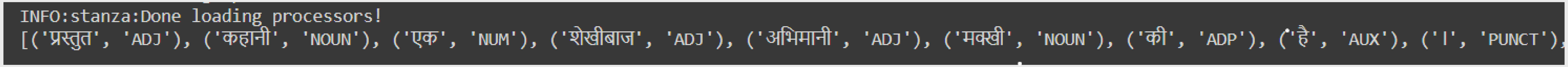
Parts of Speech Tagging

POS tagging is the process of assigning a part of speech tag to each word in the text data. This helps to identify the role of each word in a sentence, which is essential, because it helps to identify the grammatical structure of sentences and the roles played by different words within them.

We have used the stanza library of python with Hindi language for the Purpose of POS tagging. We are storing the POS tagged data thus obtained in a list.



```python
import stanza
stanza.download('hi')
nlp = stanza.Pipeline(lang='hi', processors='tokenize,pos')
doc = nlp(cleaned_text)
output_list = []
dict_list = []
for sent in doc.sentences :
  for word in sent.words:
    word_info = (
        word.text,
        word.upos,
    )
    output_list.append(word_info)
print (output_list)
```

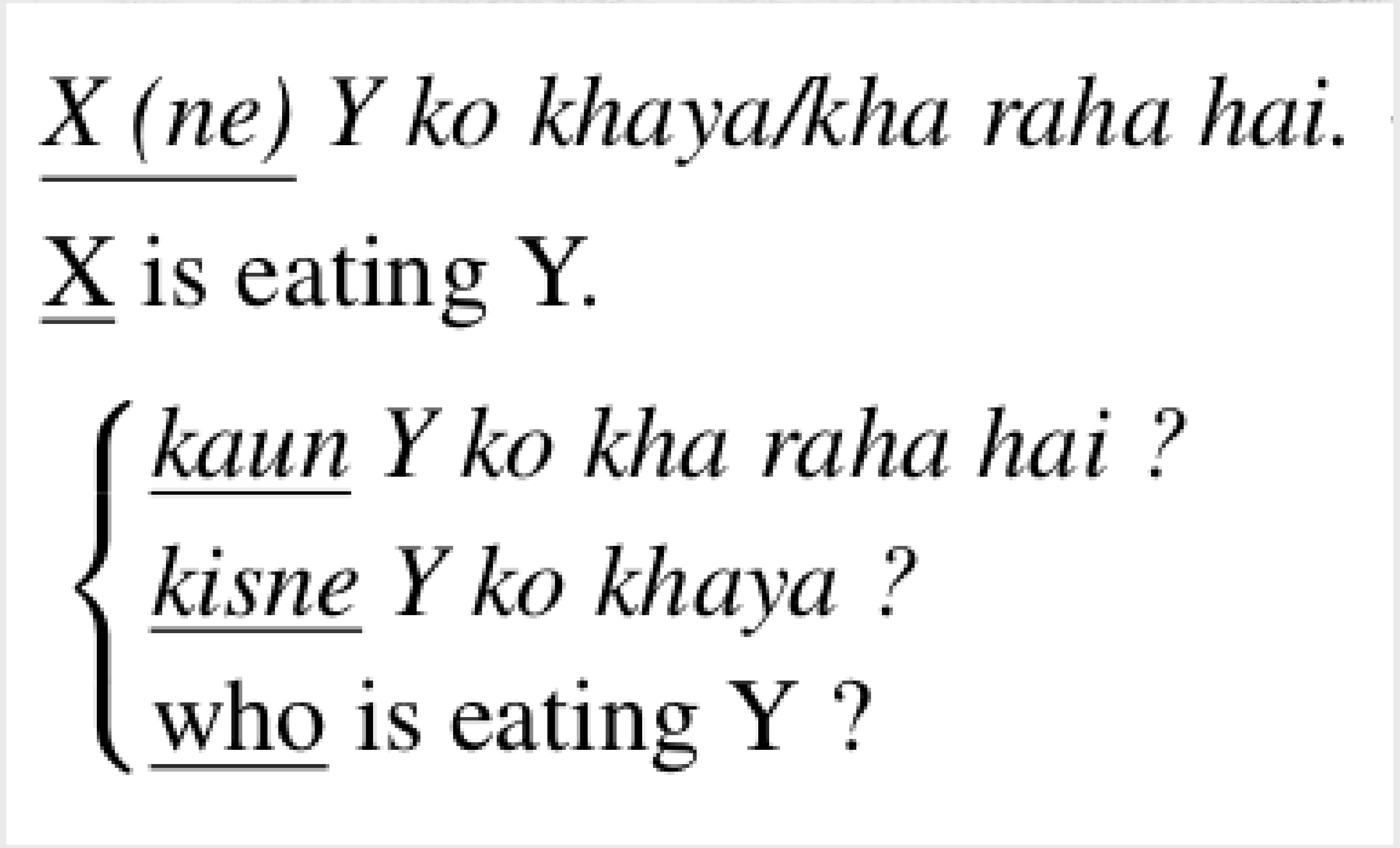


```
INFO:stanza:Done loading processors!
[('प्रस्तुत', 'ADJ'), ('कहानी', 'NOUN'), ('एक', 'NUM'), ('शेखीबाज', 'ADJ'), ('अभिमानी', 'ADJ'), ('मक्खी', 'NOUN'), ('की', 'ADP'), ('है', 'AUX'), ('।', 'PUNCT'),
```

Dependency Structures

- According to Paninian Grammar, Hindi dependency roles can be explained in terms of Karakas.

- The head of a sentence is the main verb while the rest of the phrases are children of the main verb. The role of these child nodes with respect to the main verb is the Karaka rols.


X (ne) Y ko khaya/kha raha hai.

X is eating Y.

kaun Y ko kha raha hai ?

kisne Y ko khaya ?

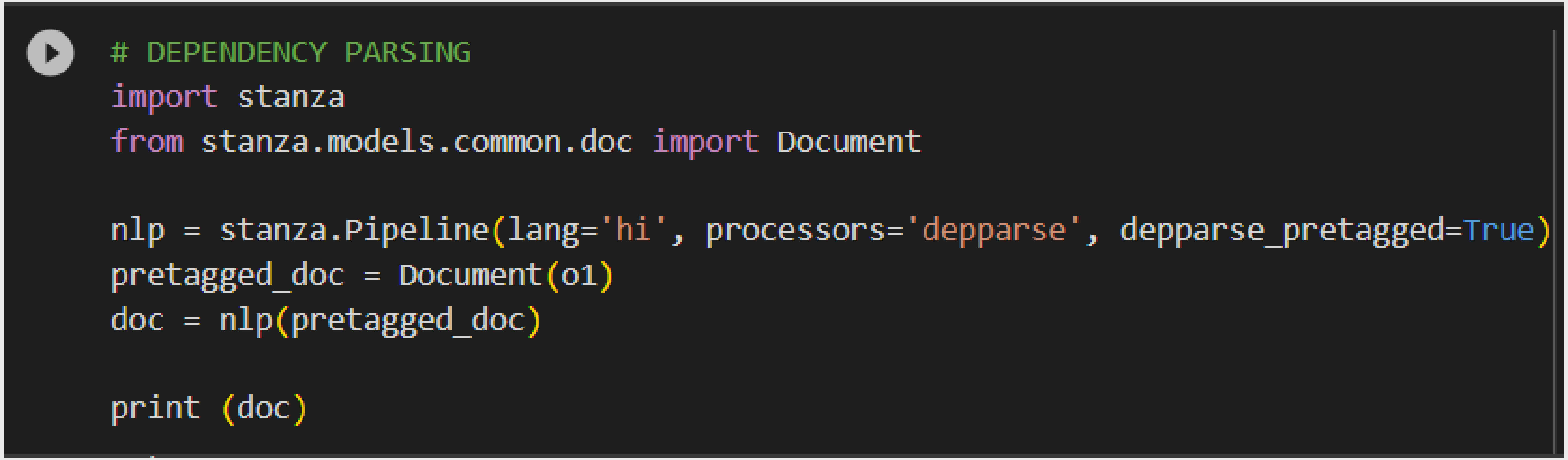
who is eating Y ?

Definition: Doer or Patient
Generally the subject of the verb. We substitute with direct or oblique interrogative words based on whether the word assigned k1 is direct or oblique (direct case does not have a postposition between the noun whereas oblique case does).

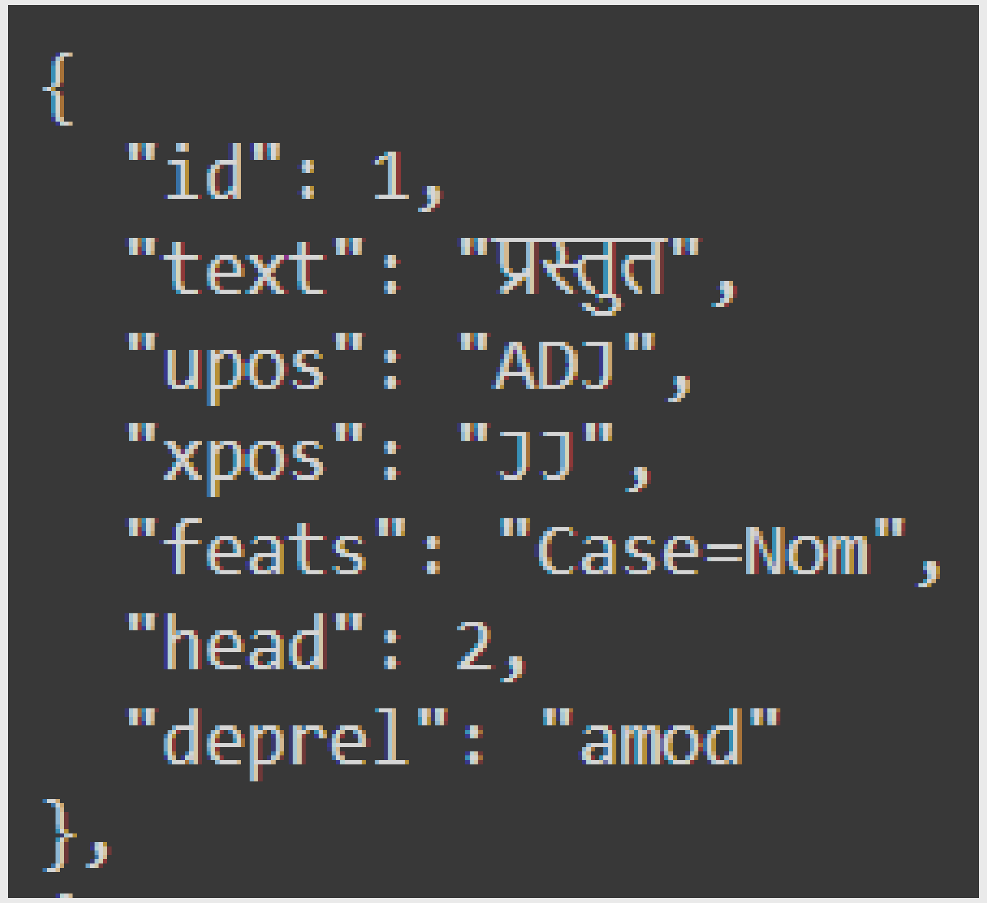
Dependency Parsing

Dependency parsing is a crucial step in question generation models because it helps in understanding the syntactic structure and relationships between words in a sentence. By analyzing the dependencies, such as subject-verb relationships, noun phrases, and modifiers, the model can generate meaningful and grammatically correct questions.

For Dependency Parsing, we are using the stanza library of python with Hindi

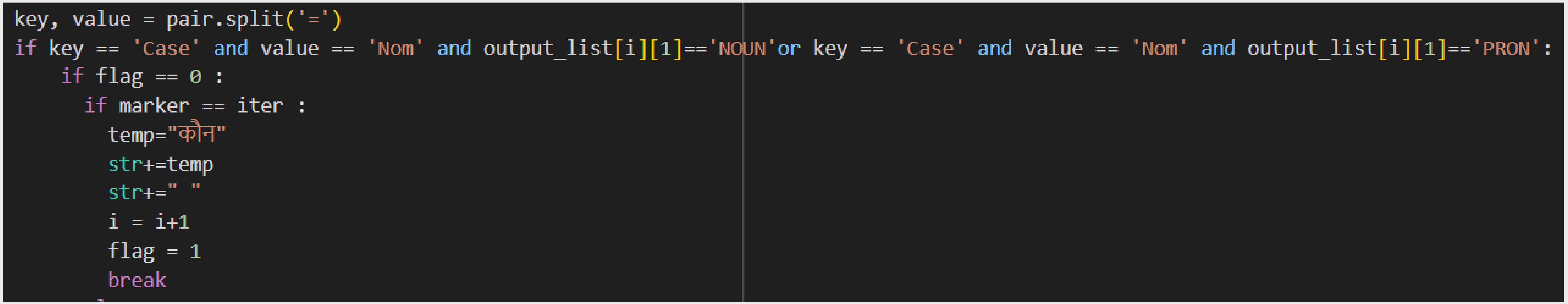


```
# DEPENDENCY PARSING
import stanza
from stanza.models.common.doc import Document

nlp = stanza.Pipeline(lang='hi', processors='depparse', depparse_pretagged=True)
pretagged_doc = Document(o1)
doc = nlp(pretagged_doc)

print (doc)
```



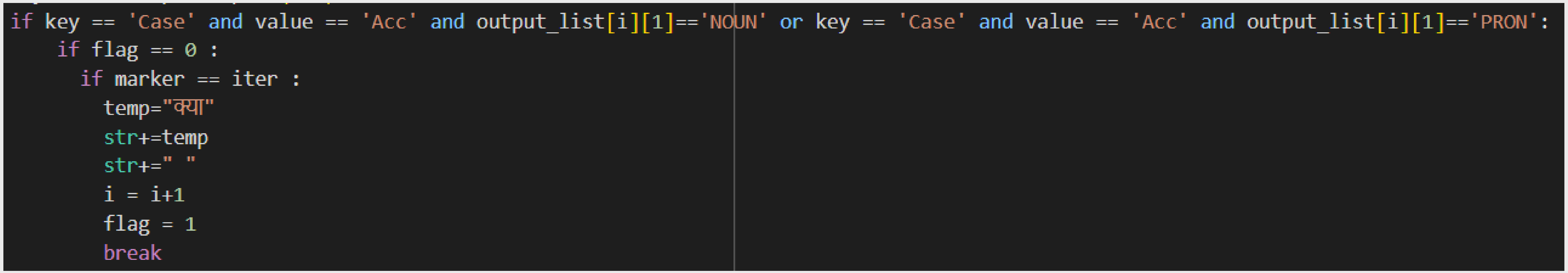
```
{
  "id": 1,
  "text": "प्रस्तुत",
  "upos": "ADJ",
  "xpos": "JJ",
  "feats": "Case=Nom",
  "head": 2,
  "deprel": "amod"
},
```

Replacing the Nominative case where word is either Noun or Pronoun by the question word "kon"

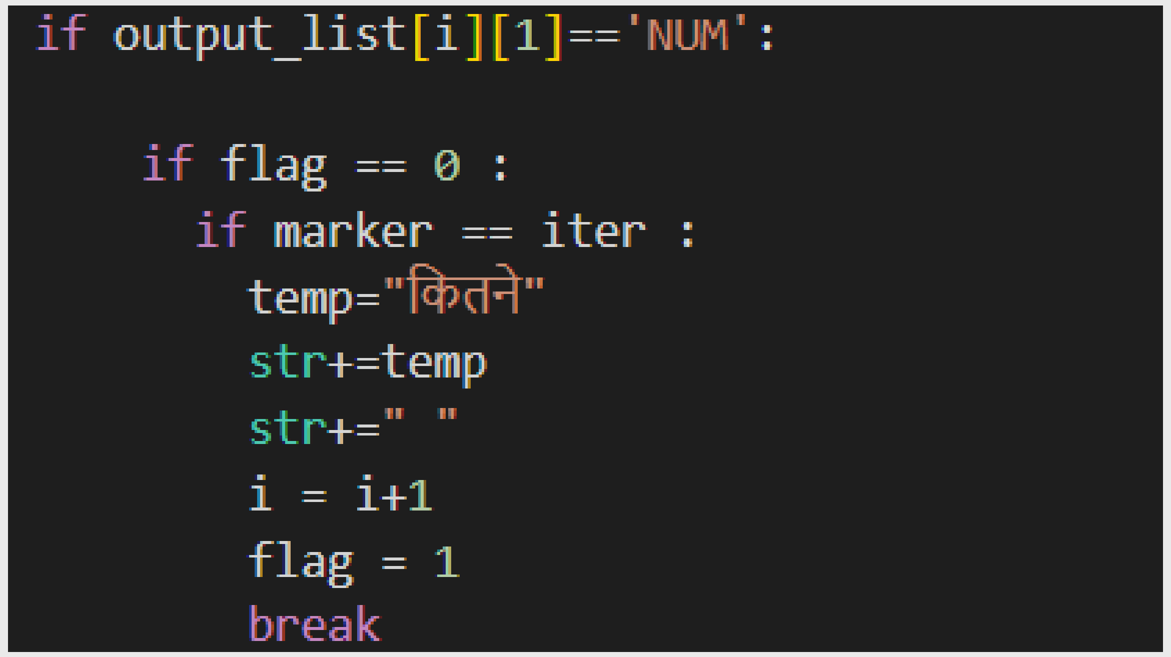


```python
key, value = pair.split('=')
if key == 'Case' and value == 'Nom' and output_list[i][1]=='NOUN'or key == 'Case' and value == 'Nom' and output_list[i][1]=='PRON':
    if flag == 0 :
      if marker == iter :
        temp="कौन"
        str+=temp
        str+=" "
        i = i+1
        flag = 1
        break
```

Replacing Accusative cases where the word is either Noun or Pronoun by the question word "kya"

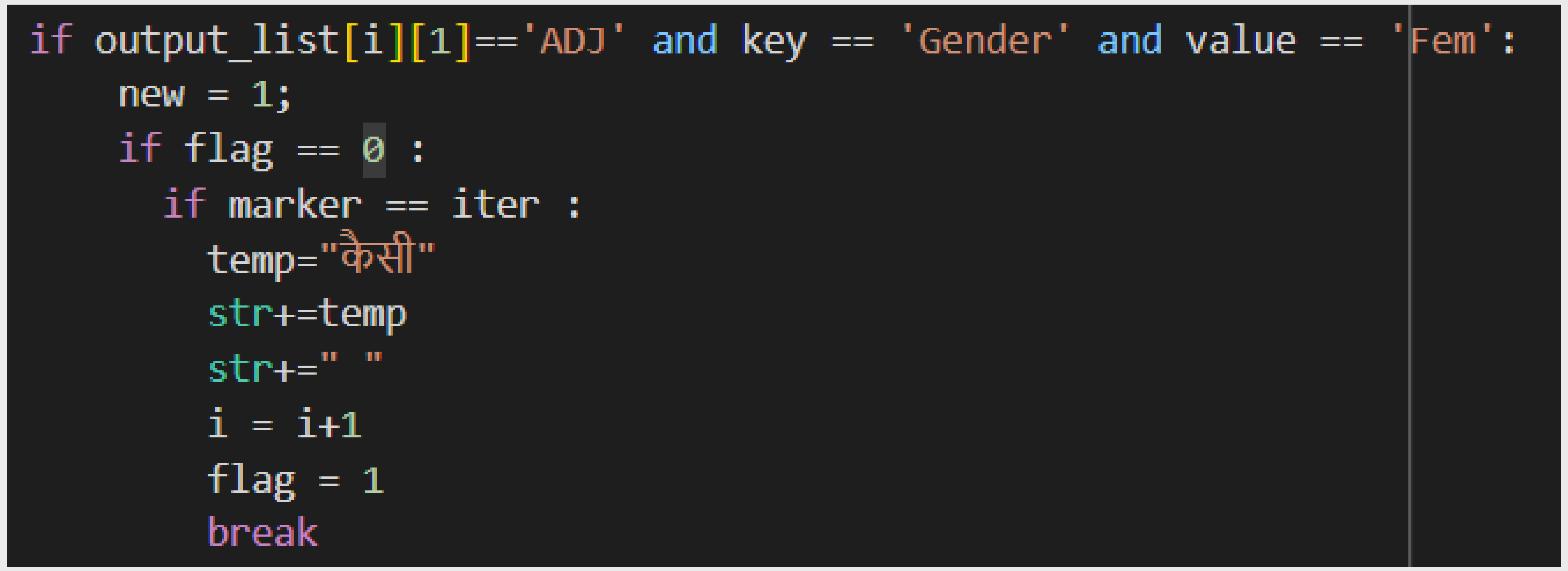


```python
if key == 'Case' and value == 'Acc' and output_list[i][1]=='NOUN' or key == 'Case' and value == 'Acc' and output_list[i][1]=='PRON':
    if flag == 0 :
      if marker == iter :
        temp="क्या"
        str+=temp
        str+=" "
        i = i+1
        flag = 1
        break
```



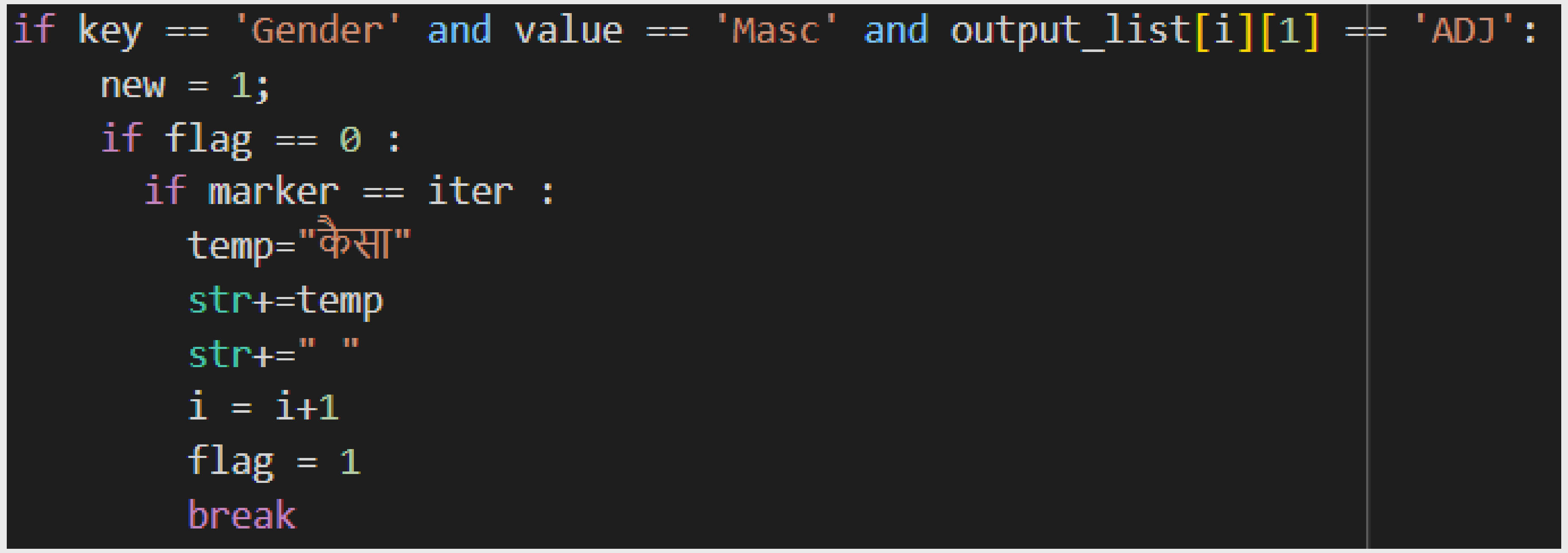
```
if output_list[i][1]=='NUM':

    if flag == 0 :
      if marker == iter :
        temp="कितने"
        str+=temp
        str+=" "
        i = i+1
        flag = 1
        break
```

Replacing the word where POS tag is number by question word "kitne"



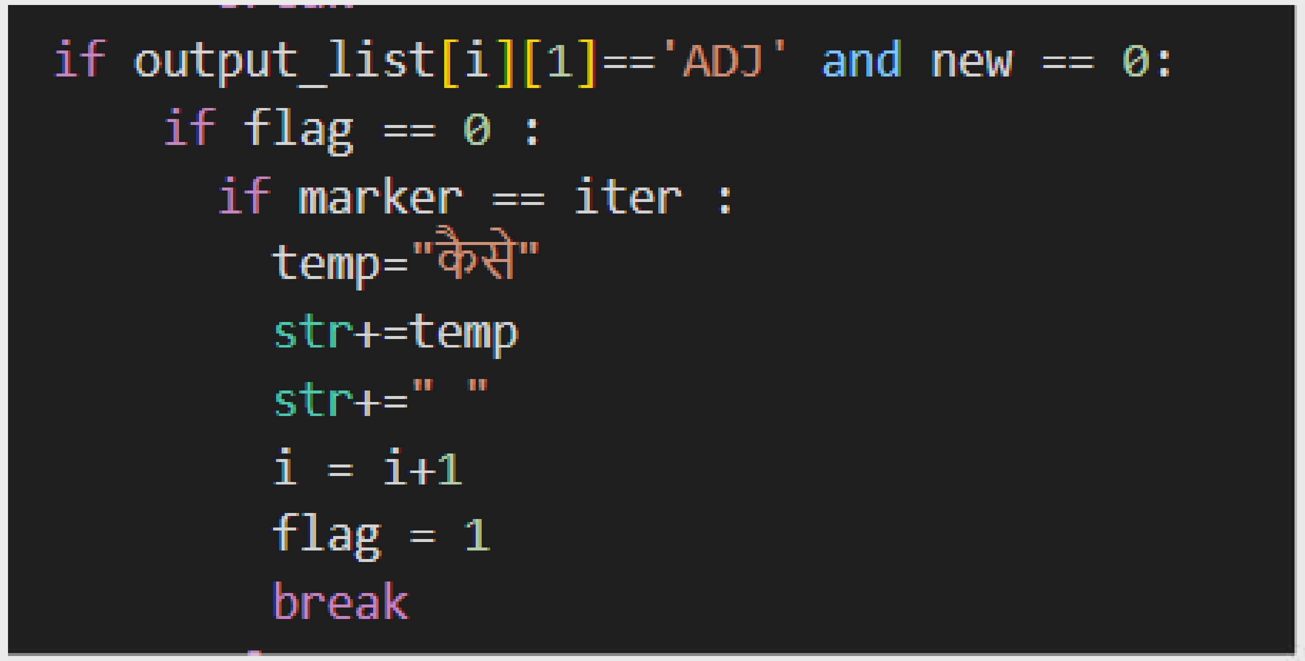
```
if output_list[i][1]=='ADJ' and key == 'Gender' and value == 'Fem':
    new = 1;
    if flag == 0 :
      if marker == iter :
        temp="कैसी"
        str+=temp
        str+=" "
        i = i+1
        flag = 1
        break
```

Replacing the word where POS tag is ADJ and Gender is Fem by 'kaisi'



```
if key == 'Gender' and value == 'Masc' and output_list[i][1] == 'ADJ':
    new = 1;
    if flag == 0 :
      if marker == iter :
        temp="कैसा"
        str+=temp
        str+=" "
        i = i+1
        flag = 1
        break
```

Replacing the word where POS tag is ADJ and Gender is Masc by 'kaisa'



```
if output_list[i][1]=='ADJ' and new == 0:
    if flag == 0 :
      if marker == iter :
        temp="कैसे"
        str+=temp
        str+=" "
        i = i+1
        flag = 1
        break
```

Replacing the word where POS tag is ADJ by 'kaise'

Results

1 . एक जंगल में एक कौन अपना भोजन करके आराम से सो रहा था ?	Correct
2. कौन कई दिनों से नहाया नहीं था ?	Correct
3. कौन गुस्सा हो गया और मक्खी को भाग जाने के लिए कहा ?	Correct
4. कौन मक्खी को प्रणाम करके बोली कि आप धन्य हो ?	Correct
5. शेर को इन आवाज से कौन नहीं आ रही थी ?	Incorrect

- Cannot distinguish animacy and inanimacy
- When adjective is preceding the noun, then the question word is inserted at the position of the noun directly keeping the adjective word too.

Resources

- POS tagger: https://stanfordnlp.github.io/stanza/pos.html
- Dependency Parsing: https://stanfordnlp.github.io/stanza/depparse.html

- Hindi Question Generation using Dependency Structures
  By - Radhika Mamidi, Kaveri Anurajama and Vijjini Rao (LTRC, IIIT Hyderabad)
  https://www.academia.edu/66658585
  /Hindi_Question_Generation_Using_Dependency_Structures -

Link to our Colab Notebook -

HTTPS://COLAB.RESEARCH.GOOGLE.COM/DRIVE/1
GBXYUMCWZXWXDIJ6E_79BGJJIPCEK18U

THANK YOU FOR EXPLORING THE WORLD OF QUESTION GENERATION USING FABLES WITH US!