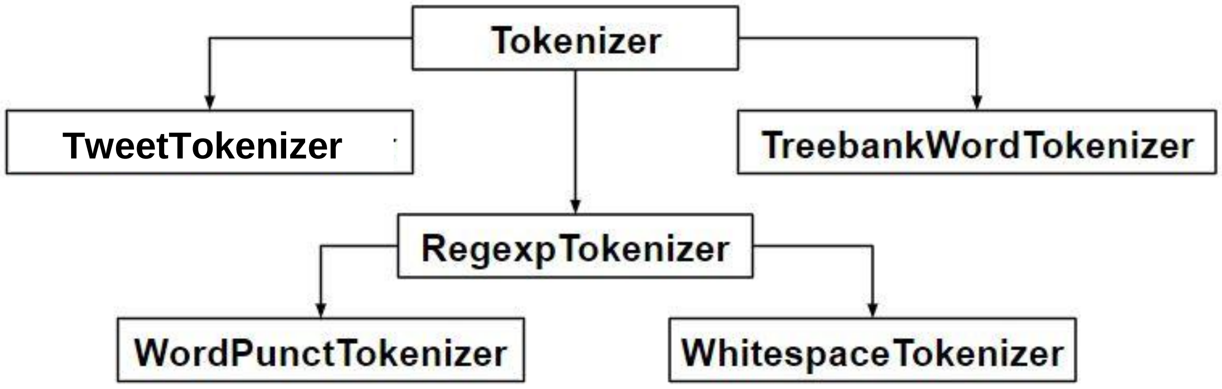
Yash Verma
CSE-AI & ML
3RD YEAR
500069950
ROLL NO.110
B2 BATCH
LAB-3 ASSIGNMENT
COMPUTATIONAL LINGUISTICS AND NLP LAB

```
# -*- coding: utf-8 -*-
"""Untitled16.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1w7qOuuGMSa-
n9TVJkPg7PPUjpL2XBTl1
"""
```

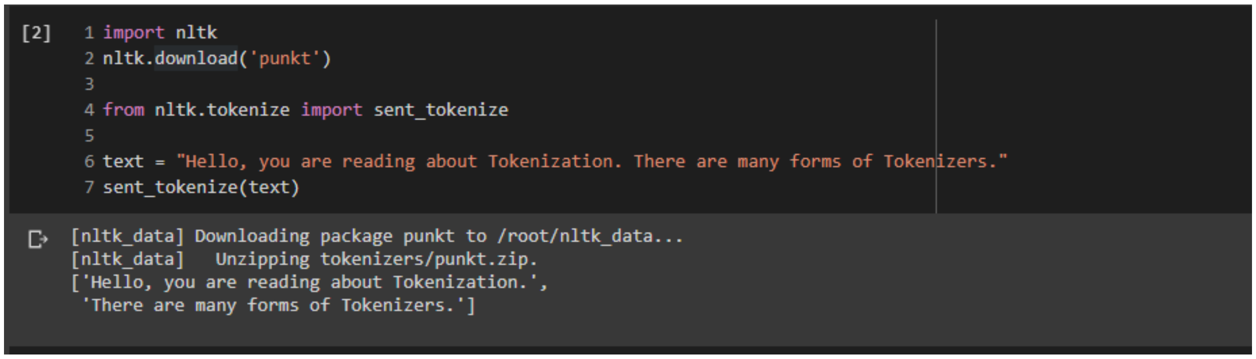
Natural Language Processing (NLP) is a subfield of computer science, artificial intelligence, information engineering, and human-computer interaction. This field focuses on how to program computers to process and analyze large amounts of natural language data. It is difficult to perform as the process of reading and understanding languages is far more complex than it seems at first glance.
Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph.
Key points that I'll be discussing are shown in the graph:



1.Sentence Tokenization – Splitting sentences in the paragraph

How sent_tokenize works ?
The sent_tokenize function uses an instance of PunktSentenceTokenizer from the nltk.tokenize.punkt module, which is already been trained and thus very well knows to mark the end and beginning of sentence at what characters and punctuation

```
import nltk
nltk.download('punkt')

from nltk.tokenize import sent_tokenize
text = "Hello, you are reading about Tokenization. There are many forms of Tokenizers."
sent_tokenize(text)
```



```
[2]  import nltk
     nltk.download('punkt')

     from nltk.tokenize import sent_tokenize

     text = "Hello, you are reading about Tokenization. There are many forms of Tokenizers."
     sent_tokenize(text)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['Hello, you are reading about Tokenization.',
 'There are many forms of Tokenizers.']
```

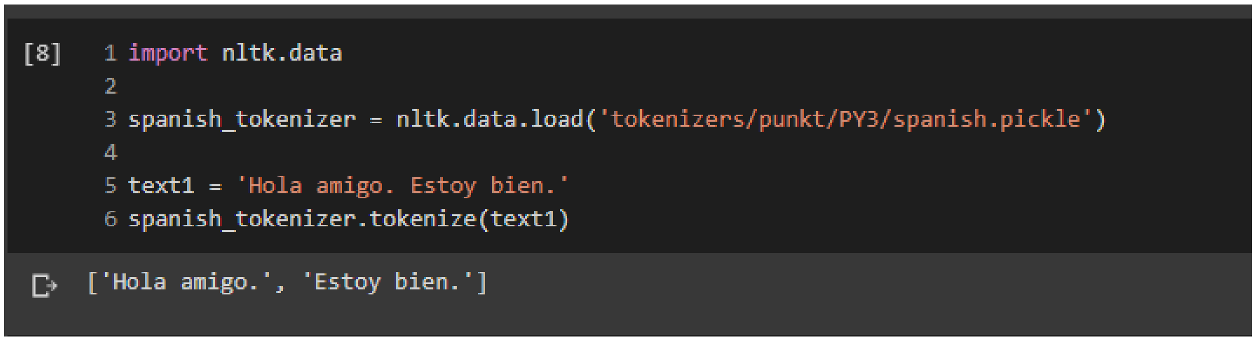
2.PunktSentenceTokenizer – When we have huge chunks of data then it is efficient to use it.

```
import nltk.data
newsent= "Tokenization is a way of separating a piece of text into smaller units called tokens.Here, tokens can be either words, characters, or subwords. Hence, tokenization can be broadly classified into 3 types – word, character, and subword (n-gram characters) tokenization.For example, consider the sentence:Never give up.The most common way of forming tokens is based on space. Assuming space as a delimiter, the tokenization of the sentence results in 3 tokens – Never-give-up. As each token is a word, it becomes an example of Word tokenization.Similarly, tokens can be either characters or subwords. For example, let us consider “smarter”:Character tokens: s-m-a-r-t-e-rSubword tokens: smart-er.But then is this necessary? Do we really need tokenization to do all of this?"
# Loading PunktSentenceTokenizer using English pickle file
tokenizer = nltk.data.load('tokenizers/punkt/PY3/english.pickle')

tokenizer.tokenize(newsent)
```

```
['Tokenization is a way of separating a piece of text into smaller units called tokens.Here, tokens can be either words, characters, or subwords.',
 'Hence, tokenization can be broadly classified into 3 types – word, character, and subword (n-gram characters) tokenization.For example, consider the sentence:Never give up.The most common way of forming tokens is based on space.',
 'Assuming space as a delimiter, the tokenization of
the sentence results in 3 tokens – Never-give-up.',
 'As each token is a word, it becomes an example of
Word tokenization.Similarly, tokens can be either
characters or subwords.',
 'For example, let us consider “smarter”:Character
tokens: s-m-a-r-t-e-rSubword tokens: smart-er.But then
is this necessary?',
 'Do we really need tokenization to do all of this?']
```

3.Tokenize sentence of different language – One can also tokenize sentence from different languages using different pickle file other than English.

```
import nltk.data

spanish_tokenizer =
nltk.data.load('tokenizers/punkt/PY3/spanish.pickle')

text1 = 'Hola amigo. Estoy bien.'
spanish_tokenizer.tokenize(text1)
```

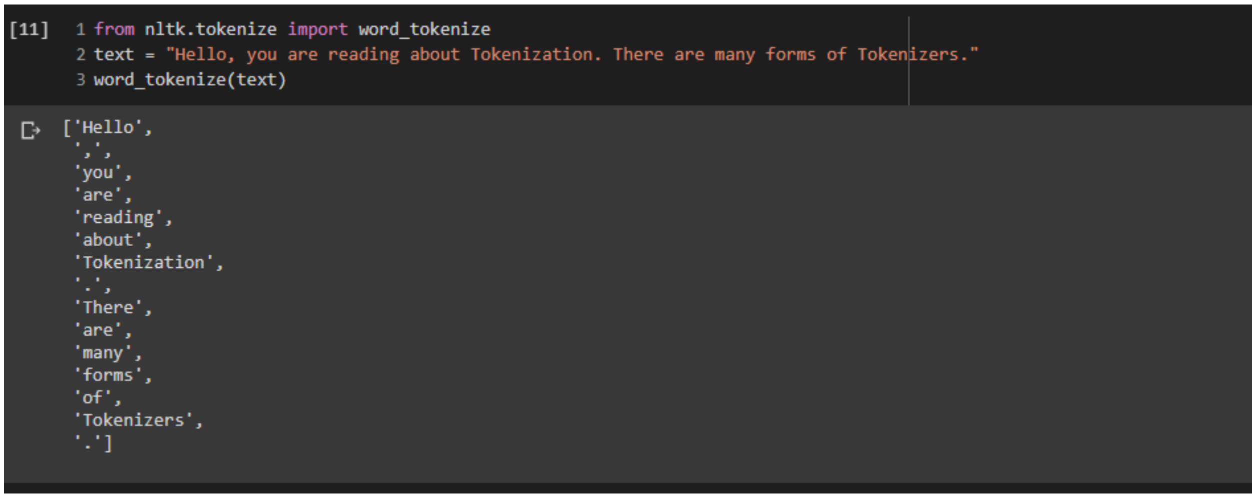


```
[8] import nltk.data

    spanish_tokenizer = nltk.data.load('tokenizers/punkt/PY3/spanish.pickle')

    text1 = 'Hola amigo. Estoy bien.'
    spanish_tokenizer.tokenize(text1)

['Hola amigo.', 'Estoy bien.']
```

4.Word Tokenization – Splitting words in a sentence.

How word_tokenize works?
word_tokenize() function is a wrapper function that calls tokenize() on an instance of the TreebankWordTokenizer class.

```
from nltk.tokenize import word_tokenize
text = "Hello, you are reading about Tokenization. There are many forms of Tokenizers."
word_tokenize(text)
```

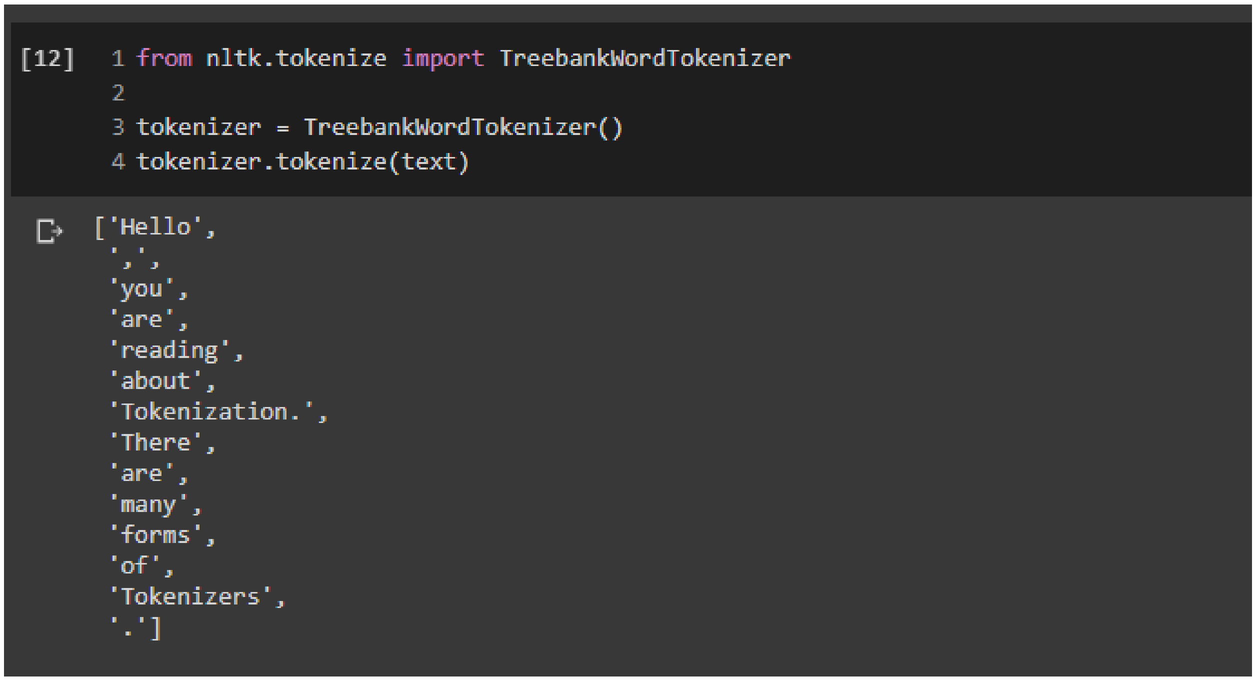


```
[11]
from nltk.tokenize import word_tokenize
text = "Hello, you are reading about Tokenization. There are many forms of Tokenizers."
word_tokenize(text)
```

```
['Hello',
 ',',
 'you',
 'are',
 'reading',
 'about',
 'Tokenization',
 '.',
 'There',
 'are',
 'many',
 'forms',
 'of',
 'Tokenizers',
 '.']
```

5.Using TreebankWordTokenizer
These tokenizers work by separating the words using punctuation and spaces. And as mentioned in the code outputs above, it does not discard the punctuation, allowing a user to decide what to do with the punctuations at the time of pre-processing.

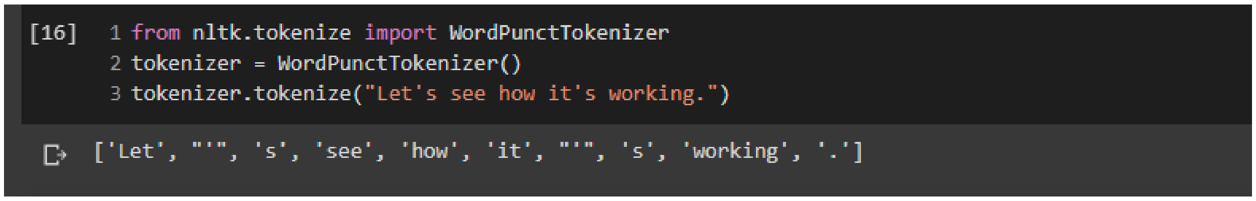
```
from nltk.tokenize import TreebankWordTokenizer
tokenizer = TreebankWordTokenizer()
tokenizer.tokenize(text)
```


```
[12]
from nltk.tokenize import TreebankWordTokenizer

tokenizer = TreebankWordTokenizer()
tokenizer.tokenize(text)
```

```
['Hello',
 ',',
 'you',
 'are',
 'reading',
 'about',
 'Tokenization.',
 'There',
 'are',
 'many',
 'forms',
 'of',
 'Tokenizers',
 '.']
```

6.WordPunctTokenizer – It seperates the punctuation from the words.

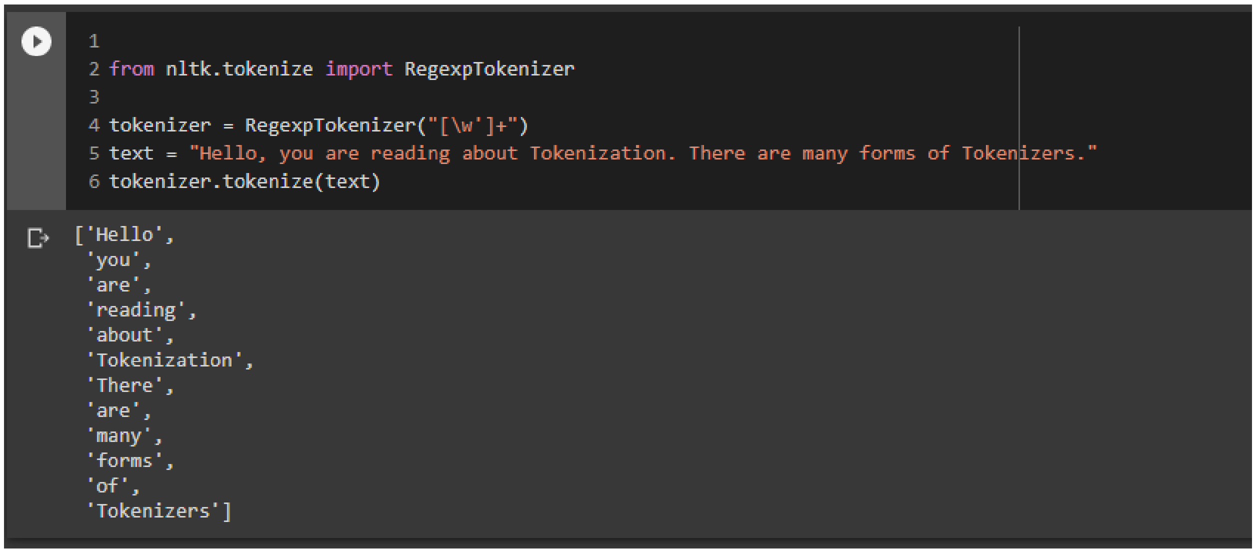
```
from nltk.tokenize import WordPunctTokenizer
tokenizer = WordPunctTokenizer()
tokenizer.tokenize("Let's see how it's working.")
```


```
[16]
from nltk.tokenize import WordPunctTokenizer
tokenizer = WordPunctTokenizer()
tokenizer.tokenize("Let's see how it's working.")
```

```
['Let', "'", 's', 'see', 'how', 'it', "'", 's', 'working', '.']
```

7.Using Regular Expression

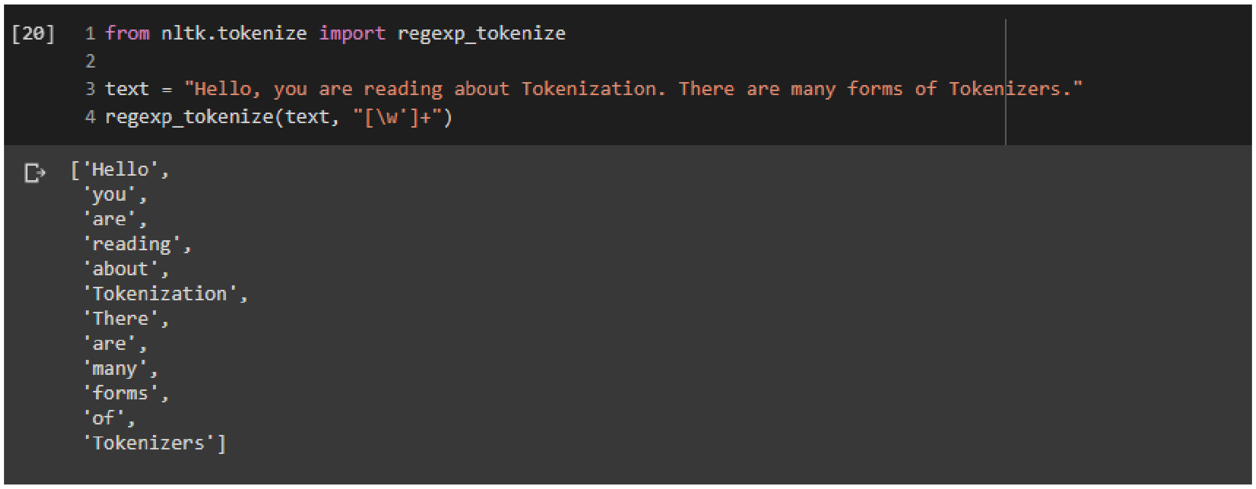
```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer("[\w']+")
text = "Hello, you are reading about Tokenization.
There are many forms of Tokenizers."
tokenizer.tokenize(text)
```



```

from nltk.tokenize import RegexpTokenizer

tokenizer = RegexpTokenizer("[\w']+")
text = "Hello, you are reading about Tokenization. There are many forms of Tokenizers."
tokenizer.tokenize(text)
```

```
['Hello',
 'you',
 'are',
 'reading',
 'about',
 'Tokenization',
 'There',
 'are',
 'many',
 'forms',
 'of',
 'Tokenizers']
```

```
from nltk.tokenize import regexp_tokenize
text = "Hello, you are reading about Tokenization.
There are many forms of Tokenizers."
regexp_tokenize(text, "[\w']+")
```

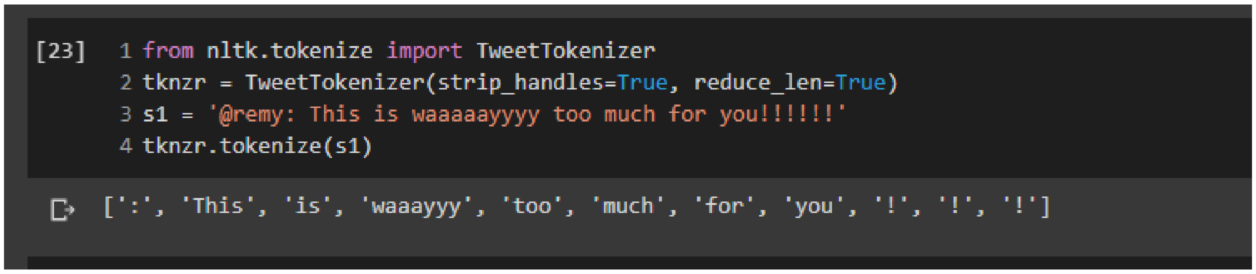


```
[20] from nltk.tokenize import regexp_tokenize

     text = "Hello, you are reading about Tokenization. There are many forms of Tokenizers."
     regexp_tokenize(text, "[\w']+")
```

```
['Hello',
 'you',
 'are',
 'reading',
 'about',
 'Tokenization',
 'There',
 'are',
 'many',
 'forms',
 'of',
 'Tokenizers']
```

8.Twitter-aware tokenizer,
Is designed to be flexible and easy to adapt to new domains and tasks.Because there are many popular Twitter language and very colloquial things,I think it is vary useful for analyzing data from Twitter. And one of the most interesting features of TweetTokenizer is parameter of *reduce_len parameters which is to replace repeated character sequences of length 3 or greater with sequences of length 3 and parameter of remove_handles which is to remove Twitter username handles from text.Here is an example:*

```
from nltk.tokenize import TweetTokenizer
tknzr = TweetTokenizer(strip_handles=True,
reduce_len=True)
s1 = '@remy: This is waaaaayyyy too much for you!!!!!!'
tknzr.tokenize(s1)
```



```
[23]
from nltk.tokenize import TweetTokenizer
tknzr = TweetTokenizer(strip_handles=True, reduce_len=True)
s1 = '@remy: This is waaaaayyyy too much for you!!!!!!'
tknzr.tokenize(s1)
```

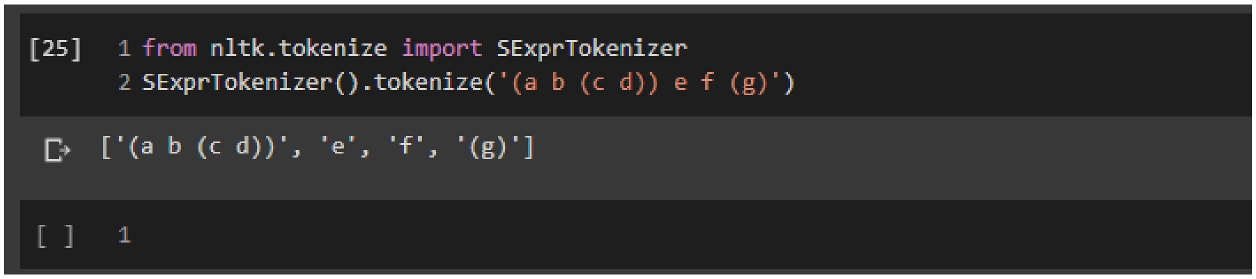
```
[':', 'This', 'is', 'waaayyy', 'too', 'much', 'for', 'you', '!', '!', '!']
```

9.SExprTokenizer

It divides the string into tokens based on parenthesized expressions and whitespace.

When there are non-matching parentheses there will be a ‘valueerror’ exception. The “strict" argument can be set to False to allow for non-matching parentheses.

```
from nltk.tokenize import SExprTokenizer
SExprTokenizer().tokenize('(a b (c d)) e f (g)')
```



```
[25] from nltk.tokenize import SExprTokenizer
     SExprTokenizer().tokenize('(a b (c d)) e f (g)')

['(a b (c d))', 'e', 'f', '(g)']

[ ]
```