Experiment No.2
Apply Tokenization on given English and Indian Language Text
Date of Performance:
Date of Submission:

Aim: Apply Tokenization on given English and Indian Language Text

Objective: Able to perform sentence and word tokenization for the given input text for English and Indian Langauge.

Theory:

Tokenization is one of the first step in any NLP pipeline. Tokenization is nothing but splitting the raw text into small chunks of words or sentences, called tokens. If the text is split into words, then its called as 'Word Tokenization' and if it's split into sentences then its called as 'Sentence Tokenization'. Generally 'space' is used to perform the word tokenization and characters like 'periods, exclamation point and newline char are used for Sentence Tokenization. We have to choose the appropriate method as per the task in hand. While performing the tokenization few characters like spaces, punctuations are ignored and will not be the part of final list of tokens.

Why Tokenization is Required?

Every sentence gets its meaning by the words present in it. So by analyzing the words present in the text we can easily interpret the meaning of the text. Once we have a list of words we can also use statistical tools and methods to get more insights into the text. For example, we can use word count and word frequency to find out important of word in that sentence or document.

Input Text

Tokenization is one of the first step in any NLP pipeline. Tokenization is nothing but splitting the raw text into small chunks of words or sentences, called tokens.

Word Tokenization

Tokenization	is	one	of
the	first	step	in
any	NLP	pipeline	Tokenization
is	nothing	but	splitting
the	raw	text	into
small	chunks	of	words
or	sentences	called	tokens

Sentence Tokenization

Tokenization is one of the first step in any NLP pipeline
Tokenization is nothing but splitting the raw text into small chunks of words or sentences, called tokens

Output:

Experiment 02

In [ ]:

Library required for Preprocessing

In [ ]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.6)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

In [ ]:
```
import nltk
```

In [ ]:
```
nltk.download()
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-27-6e230a00a763> in <cell line: 1>()
----> 1 nltk.download()

/usr/local/lib/python3.10/dist-packages/nltk/downloader.py in download(self, info_or_id, download_dir, quiet, force, prefix, halt_on_error, raise_on_error, print_error_to)
    761             if download_dir is not None:
    762                 self._download_dir = download_dir
--> 763             self._interactive_download()
    764             return True
    765

/usr/local/lib/python3.10/dist-packages/nltk/downloader.py in _interactive_download(self)
   1113                 DownloaderGUI(self).mainloop()
   1114             except TclError:
-> 1115                 DownloaderShell(self).run()
   1116         else:
   1117             DownloaderShell(self).run()

/usr/local/lib/python3.10/dist-packages/nltk/downloader.py in run(self)
   1139                 "q) Quit",
   1140             )
-> 1141             user_input = input("Downloader> ").strip()
   1142             if not user_input:
   1143                 print()

/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py in raw_input(self, prompt)
    849                 "raw_input was called, but this frontend does not support input requests."
    850             )
--> 851         return self._input_request(str(prompt),
    852             self._parent_ident,
    853             self._parent_header,

/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident, parent, password)
    893             except KeyboardInterrupt:
    894                 # re-raise KeyboardInterrupt, to truncate traceback
--> 895                 raise KeyboardInterrupt("Interrupted by user") from None
    896             except Exception as e:
    897                 self.log.warning("Invalid Message:", exc_info=True)
```

In [ ]:
```
from nltk.tokenize import sent_tokenize
```

In [ ]:
```
text = '''Stephenson 2-18 is now known as being one of the largest, if not the current largest star ever discovered, surpas
        Stephenson 2-18 has a radius of 2,150 solar radii, being larger than almost the entire orbit of Saturn (1,940 - 2,1
```

In [ ]:
```
text
```

Out[ ]:
```
'Stephenson 2-18 is now known as being one of the largest, if not the current largest star ever discovered, surpassing other
stars like VY Canis Majoris and UY Scuti.\n        Stephenson 2-18 has a radius of 2,150 solar radii, being larger than almo
st the entire orbit of Saturn (1,940 - 2,160 solar radii).'
```

In [ ]:
```
sentences = sent_tokenize (text)
```

In [ ]:
```
sentences
```

Out[ ]:
```
['Stephenson 2-18 is now known as being one of the largest, if not the current largest star ever discovered, surpassing othe
r stars like VY Canis Majoris and UY Scuti.',
 'Stephenson 2-18 has a radius of 2,150 solar radii, being larger than almost the entire orbit of Saturn (1,940 - 2,160 sola
r radii).']
```

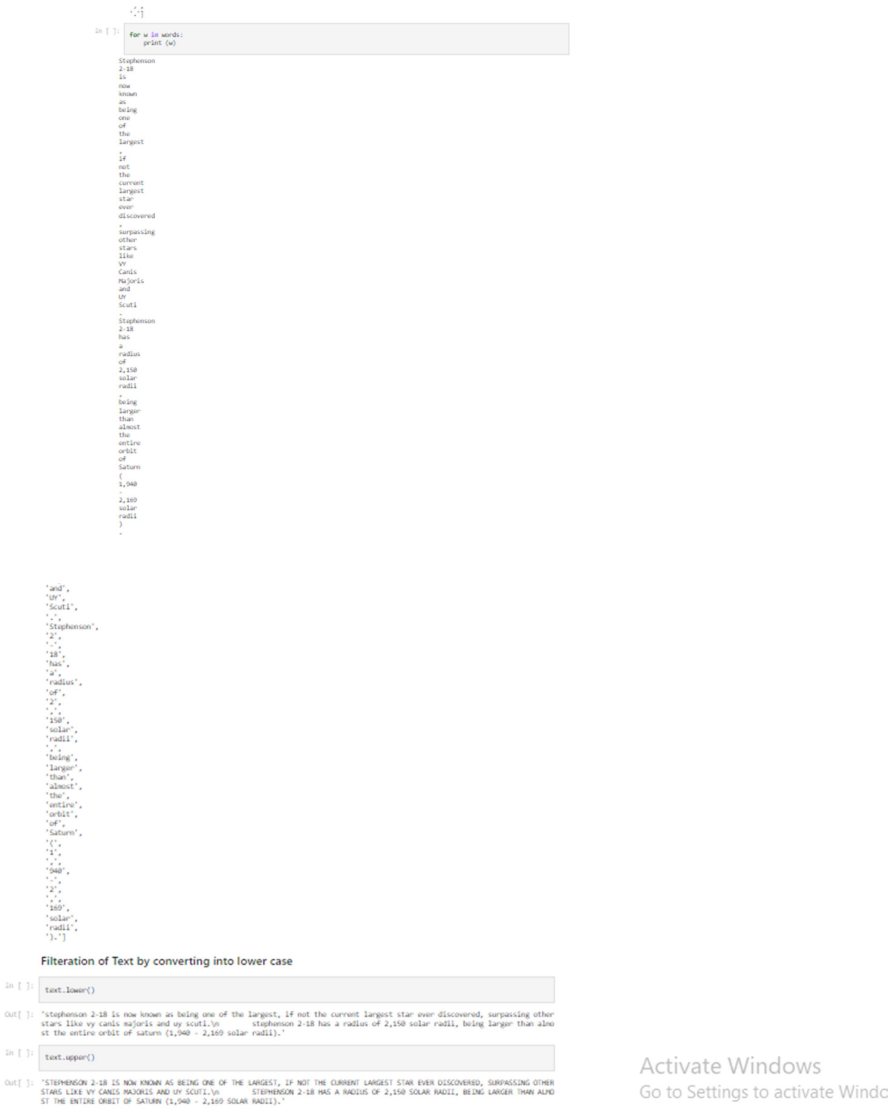
Word Tokenization

In [ ]:
```
from nltk.tokenize import word_tokenize
```

In [ ]:
```
words = word_tokenize (text)
```

In [ ]:
```
words
```

Out[ ]:
```
['Stephenson',
 '2-18',
 'is',
 'now',
 'known',
 'as',
 'being',
 'one',
 'of',
 'the',
 'largest',
 ',',
 'if',
 'not',
 'the',
 'current',
 'largest',
 'star',
 'ever',
 'discovered',
 ',',
 'surpassing',
 'other',
 'stars',
 'like',
 'VY',
 'Canis',
 'Majoris',
 'and',
 'UY',
 'Scuti',
 '.',
 'Stephenson',
 '2-18',
 'has',
 'a',
 'radius',
 'of',
 '2,150',
 'solar',
 'radii',
 ',',
 'being',
```



Filteration of Text by converting into lower case

Conclusion:

There are a number of tools available for tokenization of Indian language input. Some of the most popular tools include: iNLTK: iNLTK is a Python library for natural language processing (NLP) in Indian languages. It includes a variety of NLP tools, including a tokenizer for Indian languages. Mila NMT: Mila NMT is a machine translation toolkit that includes a tokenizer for Indian languages. Indic NLP Library: The Indic NLP Library is a Python library for NLP in Indian languages. It includes a variety of NLP tools, including a tokenizer for Indian languages. spaCy: spaCy is a Python library for NLP. It includes a tokenizer for Indian languages, but it is not as comprehensive as the other tools listed above.