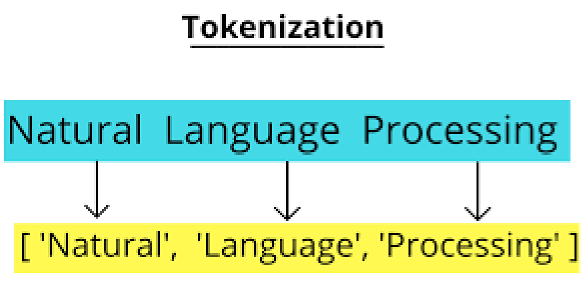
- Tokenization: Splitting the text into individual words or tokens.
- Lowercasing: Converting all text to lowercase to ensure consistent word representation.
- Stop word removal: Removing common words that do not carry much meaning (e.g., "a," "the," "is") to reduce noise.
- Punctuation removal: Removing punctuation marks to focus on the essential words.
- Lemmatization or stemming: Reducing words to their base or root form to normalize variations (e.g., "running" to "run").
- Removing numbers: Eliminating numerical values that may not be relevant for the analysis.
- Removing special characters: Eliminating symbols or special characters that do not contribute to the meaning.
- Handling contractions: Expanding contractions (e.g., "can't" to "cannot") for consistent word representation.
- Removing HTML tags (if applicable): Removing HTML tags if dealing with web data.
- Handling encoding issues: Addressing encoding problems to ensure proper text handling.
- Handling missing data: Dealing with missing values in the text, if any, through imputation or removal.
- Removing irrelevant information: Eliminating non-textual content, such as URLs or email addresses.
- Spell checking/correction: Correcting common spelling errors to improve the quality of the text.
- Removing excess white spaces: Eliminating extra spaces or tabs between words.
- Normalizing whitespace: Ensuring consistent spacing between words.
- Sentence segmentation: Splitting the text into individual sentences, if required.
- Feature engineering: Extracting additional features from the text, such as n-grams or part-of-speech tags, for more advanced analyses.

Tokenization


Tokenization
Natural Language Processing
['Natural', 'Language', 'Processing']

Word Tokenization

In [5]:
```
text = """There are multiple ways we can perform tokenization on given text data.
          We can choose any method based on langauge, library and purpose of mode
tokens = text.split()
print(tokens)
```

```
['There', 'are', 'multiple', 'ways', 'we', 'can', 'perform', 'tokenization', 'on',
'given', 'text', 'data.', 'We', 'can', 'choose', 'any', 'method', 'based', 'on',
'langauge,', 'library', 'and', 'purpose', 'of', 'modeling.']
```

Sentence Tokenization

In [12]:
```
text = """Characters like periods, exclamation point and newline char are used to s

line = text.split(". ")
line
```

Out[12]:
```
['Characters like periods, exclamation point and newline char are used to separate
the sentences',
 'But one drawback with split() method, that we can only use one separator at a ti
me! So sentence tonenization wont be foolproof with split() method.']
```

Tokenization Using RegEx

In [14]:
```
import re
text = """There are multiple ways we can perform tokenization on given text data.
We can choose any method based on langauge, library and purpose of modeling."""
tokens = re.findall("[\w]+", text)
print(tokens)
```

```
['There', 'are', 'multiple', 'ways', 'we', 'can', 'perform', 'tokenization', 'on',
'given', 'text', 'data', 'We', 'can', 'choose', 'any', 'method', 'based', 'on', 'l
angauge', 'library', 'and', 'purpose', 'of', 'modeling']
```

Sentence Tokenization

In [17]:
```
text = """Characters like periods, exclamation point and newline char are used to s
tokens_sent = re.compile('[.!?] ').split(text)
tokens_sent
```

Out[17]:
```
['Characters like periods, exclamation point and newline char are used to separate
the sentences.But one drawback with split() method, that we can only use one separ
ator at a time',
 'So sentence tonenization wont be foolproof with split() method.']
```

Tokenization Using NLTK

word Tokenization

In [18]:
```
from nltk.tokenize import word_tokenize
text = """There are multiple ways we can perform tokenization on given text data. W
tokens = word_tokenize(text)
print(tokens)
```

```
['There', 'are', 'multiple', 'ways', 'we', 'can', 'perform', 'tokenization', 'on',
'given', 'text', 'data', '.', 'We', 'can', 'choose', 'any', 'method', 'based', 'o
n', 'langauge', ',', 'library', 'and', 'purpose', 'of', 'modeling', '.']
```

sentence Tokenization

In [20]:
```
from nltk.tokenize import sent_tokenize

text = """There are multiple ways we can perform tokenization on given text data. W
tokens = sent_tokenize(text)
print(tokens)
```

```
['There are multiple ways we can perform tokenization on given text data.', 'We ca
n choose any method based on langauge, library and purpose of modeling.']
```

Tokenization Using spaCy

word Tokenization

In [23]:
```
from spacy.lang.en import English
nlp = English()
text = """There are multiple ways we can perform tokenization on given text data. W
doc = nlp(text)
token = []
for tok in doc:
    token.append(tok)
print(token)
```

```
[There, are, multiple, ways, we, can, perform, tokenization, on, given, text, dat
a, ., We, can, choose, any, method, based, on, langauge, ,, library, and, purpose,
of, modeling, .]
```

sentence Tokenization

In [32]:
```
nlp = English()
nlp.add_pipe('sentencizer')
text = """Characters like periods, exclamation point and newline char are used to s
doc = nlp(text)
sentence_list =[]
for sentence in doc.sents:
    sentence_list.append(sentence.text)
print(sentence_list)
```

```
['Characters like periods, exclamation point and newline char are used to separate
the sentences.', 'But one drawback with split() method, that we can only use one s
eparator at a time!', 'So sentence tonenization wont be foolproof with split() met
hod.']
```

Tokenization using Keras

word Tokenization

In [33]:
```
from keras.preprocessing.text import text_to_word_sequence

text = """There are multiple ways we can perform tokenization on given text data. W

tokens = text_to_word_sequence(text)
print(tokens)
```

```
['there', 'are', 'multiple', 'ways', 'we', 'can', 'perform', 'tokenization', 'on',
'given', 'text', 'data', 'we', 'can', 'choose', 'any', 'method', 'based', 'on', 'l
angauge', 'library', 'and', 'purpose', 'of', 'modeling']
```

sentence Tokenization

In [34]:
```
from keras.preprocessing.text import text_to_word_sequence

text = """Characters like periods, exclamation point and newline char are used to s

text_to_word_sequence(text, split= ".", filters="!.\n")
```

```
Out[34]: ['characters like periods, exclamation point and newline char are used to separate
         the sentences',
          ' but one drawback with split() method, that we can only use one separator at a t
         ime',
          ' so sentence tonenization wont be foolproof with split() method']
```