```python
In [5]:  import os
         import nltk
         #nltk.download()
```

```python
In [67]:  AI =('''Artificial intelligence (AI) is a wide-ranging branch of computer science concer

          Artificial intelligence allows machines to model, and even improve upon, the capabilitie

           ''')
```

```python
In [68]:  AI
```

```
Out[68]:  'Artificial intelligence (AI) is a wide-ranging branch of computer science concerned wit
          h building smart machines capable of performing tasks that typically require human intel
          ligence. AI is an interdisciplinary science with multiple approaches, but advancements i
          n machine learning and deep learning are creating a paradigm shift in virtually every se
          ctor of the tech industry. \n\nArtificial intelligence allows machines to model, and eve
          n improve upon, the capabilities of the human mind. From the development of self-driving
          cars to the proliferation of smart assistants like Siri and Alexa, AI is a growing part
          of everyday life. As a result, many tech companies across various industries are investi
          ng in artificially intelligent technologies.\n\n '
```

```python
In [3]:  from nltk.tokenize import word_tokenize
```

```python
In [5]:  AI_word = word_tokenize(AI)
         AI_word
```

```
Out[5]:  ['Artificial',
          'intelligence',
          '(',
          'AI',
          ')',
          'is',
          'a',
          'wide-ranging',
          'branch',
          'of',
          'computer',
          'science',
          'concerned',
          'with',
          'building',
          'smart',
          'machines',
          'capable',
          'of',
          'performing',
          'tasks',
          'that',
          'typically',
          'require',
          'human',
          'intelligence',
          '.',
          'AI',
          'is',
          'an',
          'interdisciplinary',
          'science',
          'with',
          'multiple',
          'approaches',
          ',',
```

'but',
'advancements',
'in',
'machine',
'learning',
'and',
'deep',
'learning',
'are',
'creating',
'a',
'paradigm',
'shift',
'in',
'virtually',
'every',
'sector',
'of',
'the',
'tech',
'industry',
'.',
'Artificial',
'intelligence',
'allows',
'machines',
'to',
'model',
',',
'and',
'even',
'improve',
'upon',
',',
'the',
'capabilities',
'of',
'the',
'human',
'mind',
'.',
'From',
'the',
'development',
'of',
'self-driving',
'cars',
'to',
'the',
'proliferation',
'of',
'smart',
'assistants',
'like',
'Siri',
'and',
'Alexa',
',',
'AI',
'is',
'a',
'growing',
'part',
'of',
'everyday',
'life',

```
    '.',
    'As',
    'a',
    'result',
    ',',
    'many',
    'tech',
    'companies',
    'across',
    'various',
    'industries',
    'are',
    'investing',
    'in',
    'artificially',
    'intelligent',
    'technologies',
    '.']
```

In [6]: `len(AI_word)`

Out[6]: 120

Tokenizer

In [7]:
```python
from nltk.tokenize import sent_tokenize
```

In [8]:
```python
AI_sent = sent_tokenize(AI)
AI_sent
```

Out[8]: ['Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence.',
 'AI is an interdisciplinary science with multiple approaches, but advancements in machine learning and deep learning are creating a paradigm shift in virtually every sector of the tech industry.',
 'Artificial intelligence allows machines to model, and even improve upon, the capabilities of the human mind.',
 'From the development of self-driving cars to the proliferation of smart assistants like Siri and Alexa, AI is a growing part of everyday life.',
 'As a result, many tech companies across various industries are investing in artificially intelligent technologies.']

In [9]: `len(AI_sent)`

Out[9]: 5

In [11]:
```python
from nltk.tokenize import blankline_tokenize
AI_blank = blankline_tokenize(AI)
AI_blank
```

Out[11]: ['Artificial intelligence (AI) is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence. AI is an interdisciplinary science with multiple approaches, but advancements in machine learning and deep learning are creating a paradigm shift in virtually every sector of the tech industry.',
 'Artificial intelligence allows machines to model, and even improve upon, the capabilities of the human mind. From the development of self-driving cars to the proliferation of smart assistants like Siri and Alexa, AI is a growing part of everyday life. As a result, many tech companies across various industries are investing in artificially intelligent technologies.']

```
In [12]: len(AI_blank)
```

```
Out[12]: 2
```

bigrams,trigrams,ngrams

```
In [4]: from nltk.util import bigrams,trigrams,ngrams
```

```
In [11]: String = 'Artificial intelligence allows machines to model, and even improve upon, the c
         quotes_token = nltk.word_tokenize(String)
```

```
In [12]: String
```

```
Out[12]: 'Artificial intelligence allows machines to model, and even improve upon, the capabiliti
         es of the human mind.'
```

```
In [13]: quotes_token
```

```
Out[13]: ['Artificial',
          'intelligence',
          'allows',
          'machines',
          'to',
          'model',
          ',',
          'and',
          'even',
          'improve',
          'upon',
          ',',
          'the',
          'capabilities',
          'of',
          'the',
          'human',
          'mind',
          '.']
```

```
In [14]: len(quotes_token)
```

```
Out[14]: 19
```

```
In [16]: quotes_bigrams = list(nltk.bigrams(quotes_token))
         quotes_bigrams
```

```
Out[16]: [('Artificial', 'intelligence'),
          ('intelligence', 'allows'),
          ('allows', 'machines'),
          ('machines', 'to'),
          ('to', 'model'),
          ('model', ','),
          (',', 'and'),
          ('and', 'even'),
          ('even', 'improve'),
          ('improve', 'upon'),
          ('upon', ','),
          (',', 'the'),
          ('the', 'capabilities'),
          ('capabilities', 'of'),
          ('of', 'the'),
          ('the', 'human'),
```

```
          ('human', 'mind'),
          ('mind', '.')]

In [17]:   quotes_trigrams = list(nltk.trigrams(quotes_token))
           quotes_trigrams

Out[17]:   [('Artificial', 'intelligence', 'allows'),
            ('intelligence', 'allows', 'machines'),
            ('allows', 'machines', 'to'),
            ('machines', 'to', 'model'),
            ('to', 'model', ','),
            ('model', ',', 'and'),
            (',', 'and', 'even'),
            ('and', 'even', 'improve'),
            ('even', 'improve', 'upon'),
            ('improve', 'upon', ','),
            ('upon', ',', 'the'),
            (',', 'the', 'capabilities'),
            ('the', 'capabilities', 'of'),
            ('capabilities', 'of', 'the'),
            ('of', 'the', 'human'),
            ('the', 'human', 'mind'),
            ('human', 'mind', '.')]

In [25]:   quotes_ngrams = list(nltk.ngrams(quotes_token,5))
           quotes_ngrams

Out[25]:   [('Artificial', 'intelligence', 'allows', 'machines', 'to'),
            ('intelligence', 'allows', 'machines', 'to', 'model'),
            ('allows', 'machines', 'to', 'model', ','),
            ('machines', 'to', 'model', ',', 'and'),
            ('to', 'model', ',', 'and', 'even'),
            ('model', ',', 'and', 'even', 'improve'),
            (',', 'and', 'even', 'improve', 'upon'),
            ('and', 'even', 'improve', 'upon', ','),
            ('even', 'improve', 'upon', ',', 'the'),
            ('improve', 'upon', ',', 'the', 'capabilities'),
            ('upon', ',', 'the', 'capabilities', 'of'),
            (',', 'the', 'capabilities', 'of', 'the'),
            ('the', 'capabilities', 'of', 'the', 'human'),
            ('capabilities', 'of', 'the', 'human', 'mind'),
            ('of', 'the', 'human', 'mind', '.')]
```

Stemming

```
In [27]:   from nltk.stem import PorterStemmer
           pst=PorterStemmer()

In [29]:   pst.stem('giving')

Out[29]:   'give'

In [30]:   pst.stem('loving')

Out[30]:   'love'

In [36]:   words_to_stem=['give','giving','given','gave']
           for words in words_to_stem:
             print(words+':'+pst.stem(words))

           give:give
           giving:give
```

```
         given:given
         gave:gave
```

In [37]:
```python
words_to_stem=['give','giving','given','gave','writing','thinking']
for words in words_to_stem:
  print(words+':'+pst.stem(words))
```

```
give:give
giving:give
given:given
gave:gave
writing:write
thinking:think
```

In [38]:
```python
from nltk.stem import LancasterStemmer
lst=LancasterStemmer()
```

In [39]:
```python
words_to_stem=['give','giving','given','gave','writing','thinking']
for words in words_to_stem:
  print(words+':'+lst.stem(words))
```

```
give:giv
giving:giv
given:giv
gave:gav
writing:writ
thinking:think
```

In [46]:
```python
from nltk.stem import SnowballStemmer
sbst=SnowballStemmer('english')
```

In [47]:
```python
words_to_stem=['give','giving','given','gave','writing','thinking']
for words in words_to_stem:
  print(words+':'+sbst.stem(words))
```

```
give:give
giving:give
given:given
gave:gave
writing:write
thinking:think
```

lemmatizations

In [51]:
```python
from nltk.stem import wordnet
from nltk.stem import WordNetLemmatizer
word_lem = WordNetLemmatizer()
```

In [52]:
```python
words_to_stem
```

Out[52]:
```
['give', 'giving', 'given', 'gave', 'writing', 'thinking']
```

In [53]:
```python
for words in words_to_stem:
  print(words+':'+word_lem.lemmatize(words))
```

```
give:give
giving:giving
given:given
gave:gave
writing:writing
thinking:thinking
```

Stopwords

```python
from nltk.corpus import stopwords
```

```python
stopwords.words('english')
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
 'do',
 'does',
```

```
'did',
'doing',
'a',
'an',
'the',
'and',
'but',
'if',
'or',
'because',
'as',
'until',
'while',
'of',
'at',
'by',
'for',
'with',
'about',
'against',
'between',
'into',
'through',
'during',
'before',
'after',
'above',
'below',
'to',
'from',
'up',
'down',
'in',
'out',
'on',
'off',
'over',
'under',
'again',
'further',
'then',
'once',
'here',
'there',
'when',
'where',
'why',
'how',
'all',
'any',
'both',
'each',
'few',
'more',
'most',
'other',
'some',
'such',
'no',
'nor',
'not',
'only',
'own',
'same',
'so',
'than',
```

```
    'too',
    'very',
    's',
    't',
    'can',
    'will',
    'just',
    'don',
    "don't",
    'should',
    "should've",
    'now',
    'd',
    'll',
    'm',
    'o',
    're',
    've',
    'y',
    'ain',
    'aren',
    "aren't",
    'couldn',
    "couldn't",
    'didn',
    "didn't",
    'doesn',
    "doesn't",
    'hadn',
    "hadn't",
    'hasn',
    "hasn't",
    'haven',
    "haven't",
    'isn',
    "isn't",
    'ma',
    'mightn',
    "mightn't",
    'mustn',
    "mustn't",
    'needn',
    "needn't",
    'shan',
    "shan't",
    'shouldn',
    "shouldn't",
    'wasn',
    "wasn't",
    'weren',
    "weren't",
    'won',
    "won't",
    'wouldn',
    "wouldn't"]
```

In [58]: `stopwords.words('chinese')`

Out[58]:
```
['一',
 '一下',
 '一些',
 '一切',
 '一则',
 '一天',
 '一定',
 '一方面',
```

'一旦',
'一时',
'一来',
'一样',
'一次',
'一片',
'一直',
'一致',
'一般',
'一起',
'一边',
'一面',
'万一',
'上下',
'上升',
'上去',
'上来',
'上述',
'上面',
'下列',
'下去',
'下来',
'下面',
'不一',
'不久',
'不仅',
'不会',
'不但',
'不光',
'不单',
'不变',
'不只',
'不可',
'不同',
'不够',
'不如',
'不得',
'不怕',
'不惟',
'不成',
'不拘',
'不敢',
'不断',
'不是',
'不比',
'不然',
'不特',
'不独',
'不管',
'不能',
'不要',
'不论',
'不足',
'不过',
'不问',
'与',
'与其',
'与否',
'与此同时',
'专门',
'且',
'两者',
'严格',
'严重',
'个',
'个人',

'个别',
'中小',
'中间',
'丰富',
'临',
'为',
'为主',
'为了',
'为什么',
'为什麽',
'为何',
'为着',
'主张',
'主要',
'举行',
'乃',
'乃至',
'么',
'之',
'之一',
'之前',
'之后',
'之後',
'之所以',
'之类',
'乌乎',
'乎',
'乘',
'也',
'也好',
'也是',
'也罢',
'了',
'了解',
'争取',
'于',
'于是',
'于是乎',
'云云',
'互相',
'产生',
'人们',
'人家',
'什么',
'什么样',
'什麽',
'今后',
'今天',
'今年',
'今後',
'仍然',
'从',
'从事',
'从而',
'他',
'他人',
'他们',
'他的',
'代替',
'以',
'以上',
'以下',
'以为',
'以便',
'以免',
'以前',

'以及',
'以后',
'以外',
'以後',
'以来',
'以至',
'以至于',
'以致',
'们',
'任',
'任何',
'任凭',
'任务',
'企图',
'伟大',
'似乎',
'似的',
'但',
'但是',
'何',
'何况',
'何处',
'何时',
'作为',
'你',
'你们',
'你的',
'使得',
'使用',
'例如',
'依',
'依照',
'依靠',
'促进',
'保持',
'俺',
'俺们',
'倘',
'倘使',
'倘或',
'倘然',
'倘若',
'假使',
'假如',
'假若',
'做到',
'像',
'允许',
'充分',
'先后',
'先後',
'先生',
'全部',
'全面',
'兮',
'共同',
'关于',
'其',
'其一',
'其中',
'其二',
'其他',
'其余',
'其它',
'其实',
'其次',

'具体',
'具体地说',
'具体说来',
'具有',
'再者',
'再说',
'冒',
'冲',
'决定',
'况且',
'准备',
'几',
'几乎',
'几时',
'凭',
'凭借',
'出去',
'出来',
'出现',
'分别',
'则',
'别',
'别的',
'别说',
'到',
'前后',
'前者',
'前进',
'前面',
'加之',
'加以',
'加入',
'加强',
'十分',
'即',
'即令',
'即使',
'即便',
'即或',
'即若',
'却不',
'原来',
'又',
'及',
'及其',
'及时',
'及至',
'双方',
'反之',
'反应',
'反映',
'反过来',
'反过来说',
'取得',
'受到',
'变成',
'另',
'另一方面',
'另外',
'只是',
'只有',
'只要',
'只限',
'叫',
'叫做',
'召开',

'叮咚',
'可',
'可以',
'可是',
'可能',
'可见',
'各',
'各个',
'各人',
'各位',
'各地',
'各种',
'各级',
'各自',
'合理',
'同',
'同一',
'同时',
'同样',
'后来',
'后面',
'向',
'向着',
'吓',
'吗',
'否则',
'吧',
'吧哒',
'吱',
'呀',
'呃',
'呕',
'呗',
'呜',
'呜呼',
'呢',
'周围',
'呵',
'呸',
'呼哧',
'咋',
'和',
'咚',
'咦',
'咱',
'咱们',
'咳',
'哇',
'哈',
'哈哈',
'哉',
'哎',
'哎呀',
'哎哟',
'哗',
'哟',
'哦',
'哩',
'哪',
'哪个',
'哪些',
'哪儿',
'哪天',
'哪年',
'哪怕',
'哪样',

'哪边',
'哪里',
'哼',
'哼唷',
'唉',
'啊',
'唔',
'啥',
'啦',
'啪达',
'喂',
'喏',
'喔唷',
'嗡嗡',
'嗬',
'嗯',
'嗳',
'嘎',
'嘎登',
'嘘',
'嘛',
'嘻',
'嘿',
'因',
'因为',
'因此',
'因而',
'固然',
'在',
'在下',
'地',
'坚决',
'坚持',
'基本',
'处理',
'复杂',
'多',
'多少',
'多数',
'多次',
'大力',
'大多数',
'大大',
'大家',
'大批',
'大约',
'大量',
'失去',
'她',
'她们',
'她的',
'好的',
'好象',
'如',
'如上所述',
'如下',
'如何',
'如其',
'如果',
'如此',
'如若',
'存在',
'宁',
'宁可',
'宁愿',
'宁肯',

'它',
'它们',
'它们的',
'它的',
'安全',
'完全',
'完成',
'实现',
'实际',
'宣布',
'容易',
'密切',
'对',
'对于',
'对应',
'将',
'少数',
'尔后',
'尚且',
'尤其',
'就',
'就是',
'就是说',
'尽',
'尽管',
'属于',
'岂但',
'左右',
'巨大',
'巩固',
'己',
'已经',
'帮助',
'常常',
'并',
'并不',
'并不是',
'并且',
'并没有',
'广大',
'广泛',
'应当',
'应用',
'应该',
'开外',
'开始',
'开展',
'引起',
'强烈',
'强调',
'归',
'当',
'当前',
'当时',
'当然',
'当着',
'形成',
'彻底',
'彼',
'彼此',
'往',
'往往',
'待',
'後来',
'後面',
'得',

'得出',
'得到',
'心里',
'必然',
'必要',
'必须',
'怎',
'怎么',
'怎么办',
'怎么样',
'怎样',
'怎麼',
'总之',
'总是',
'总的来看',
'总的来说',
'总的说来',
'总结',
'总而言之',
'恰恰相反',
'您',
'意思',
'愿意',
'慢说',
'成为',
'我',
'我们',
'我的',
'或',
'或是',
'或者',
'战斗',
'所',
'所以',
'所有',
'所谓',
'打',
'扩大',
'把',
'抑或',
'拿',
'按',
'按照',
'换句话说',
'换言之',
'据',
'掌握',
'接着',
'接著',
'故',
'故此',
'整个',
'方便',
'方面',
'旁人',
'无宁',
'无法',
'无论',
'既',
'既是',
'既然',
'时候',
'明显',
'明确',
'是',
'是否',

'是的',
'显然',
'显著',
'普通',
'普遍',
'更加',
'曾经',
'替',
'最后',
'最大',
'最好',
'最後',
'最近',
'最高',
'有',
'有些',
'有关',
'有利',
'有力',
'有所',
'有效',
'有时',
'有点',
'有的',
'有着',
'有著',
'望',
'朝',
'朝着',
'本',
'本着',
'来',
'来着',
'极了',
'构成',
'果然',
'果真',
'某',
'某个',
'某些',
'根据',
'根本',
'欢迎',
'正在',
'正如',
'正常',
'此',
'此外',
'此时',
'此间',
'毋宁',
'每',
'每个',
'每天',
'每年',
'每当',
'比',
'比如',
'比方',
'比较',
'毫不',
'没有',
'沿',
'沿着',
'注意',
'深入',

'清楚',
'满足',
'漫说',
'焉',
'然则',
'然后',
'然後',
'然而',
'照',
'照着',
'特别是',
'特殊',
'特点',
'现代',
'现在',
'甚么',
'甚而',
'甚至',
'用',
'由',
'由于',
'由此可见',
'的',
'的话',
'目前',
'直到',
'直接',
'相似',
'相信',
'相反',
'相同',
'相对',
'相对而言',
'相应',
'相当',
'相等',
'省得',
'看出',
'看到',
'看来',
'看看',
'看见',
'真是',
'真正',
'着',
'着呢',
'矣',
'知道',
'确定',
'离',
'积极',
'移动',
'突出',
'突然',
'立即',
'第',
'等',
'等等',
'管',
'紧接着',
'纵',
'纵令',
'纵使',
'纵然',
'练习',
'组成',

'经',
'经常',
'经过',
'结合',
'结果',
'给',
'绝对',
'继续',
'继而',
'维持',
'综上所述',
'罢了',
'考虑',
'者',
'而',
'而且',
'而况',
'而外',
'而已',
'而是',
'而言',
'联系',
'能',
'能否',
'能够',
'腾',
'自',
'自个儿',
'自从',
'自各儿',
'自家',
'自己',
'自身',
'至',
'至于',
'良好',
'若',
'若是',
'若非',
'范围',
'莫若',
'获得',
'虽',
'虽则',
'虽然',
'虽说',
'行为',
'行动',
'表明',
'表示',
'被',
'要',
'要不',
'要不是',
'要不然',
'要么',
'要是',
'要求',
'规定',
'觉得',
'认为',
'认真',
'认识',
'让',
'许多',
'论',

'设使',
'设若',
'该',
'说明',
'诸位',
'谁',
'谁知',
'赶',
'起',
'起来',
'起见',
'趁',
'趁着',
'越是',
'跟',
'转动',
'转变',
'转贴',
'较',
'较之',
'边',
'达到',
'迅速',
'过',
'过去',
'过来',
'运用',
'还是',
'还有',
'这',
'这个',
'这么',
'这么些',
'这么样',
'这么点儿',
'这些',
'这会儿',
'这儿',
'这就是说',
'这时',
'这样',
'这点',
'这种',
'这边',
'这里',
'这麽',
'进入',
'进步',
'进而',
'进行',
'连',
'连同',
'适应',
'适当',
'适用',
'逐步',
'逐渐',
'通常',
'通过',
'造成',
'遇到',
'遭到',
'避免',
'那',
'那个',
'那么',

```
            '那么些',
            '那么样',
            '那些',
            '那会儿',
            '那儿',
            '那时',
            '那样',
            '那边',
            '那里',
            '那麼',
            '部分',
            '鄙人',
            '采取',
            '里面',
            '重大',
            '重新',
            '重要',
            '鉴于',
            '问题',
            '防止',
            '阿',
            '附近',
            '限制',
            '除',
            '除了',
            '除此之外',
            '除非',
            '随',
            '随着',
            '随著',
            '集中',
            '需要',
            '非但',
            '非常',
            '非徒',
            '靠',
            '顺',
            '顺着',
            '首先',
            '高兴',
            '是不是']
```

In [59]: `len(stopwords.words('chinese'))`

Out[59]: 841

In [60]: `stopwords.words('hindi')`

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_10764\1472798112.py in <module>
----> 1 stopwords.words('hindi')

~\anaconda3\lib\site-packages\nltk\corpus\reader\wordlist.py in words(self, fileids, ign
ore_lines_startswith)
     19         return [
     20             line
---> 21             for line in line_tokenize(self.raw(fileids))
     22             if not line.startswith(ignore_lines_startswith)
     23         ]

~\anaconda3\lib\site-packages\nltk\corpus\reader\api.py in raw(self, fileids)
    216         contents = []
    217         for f in fileids:
--> 218             with self.open(f) as fp:
```

```
   219                          contents.append(fp.read())
   220              return concat(contents)


~\anaconda3\lib\site-packages\nltk\corpus\reader\api.py in open(self, file)
   229          """
   230          encoding = self.encoding(file)
--> 231          stream = self._root.join(file).open(encoding)
   232          return stream
   233


~\anaconda3\lib\site-packages\nltk\data.py in join(self, fileid)
   332      def join(self, fileid):
   333          _path = os.path.join(self._path, fileid)
--> 334          return FileSystemPathPointer(_path)
   335
   336      def __repr__(self):


~\anaconda3\lib\site-packages\nltk\compat.py in _decorator(*args, **kwargs)
    39      def _decorator(*args, **kwargs):
    40          args = (args[0], add_py3_data(args[1])) + args[2:]
---> 41          return init_func(*args, **kwargs)
    42
    43      return wraps(init_func)(_decorator)


~\anaconda3\lib\site-packages\nltk\data.py in __init__(self, _path)
   310          _path = os.path.abspath(_path)
   311          if not os.path.exists(_path):
--> 312              raise OSError("No such file or directory: %r" % _path)
   313          self._path = _path
   314


OSError: No such file or directory: 'C:\\Users\\91996\\AppData\\Roaming\\nltk_data\\corp
ora\\stopwords\\hindi'
```

Regular Expressions

In [61]:
```python
import re
punctuation = re.compile(r'[.-?!,:;()|0-9]')
```

In [62]:
```python
punctuation
```

Out[62]:
```
re.compile(r'[.-?!,:;()|0-9]', re.UNICODE)
```

In [69]:
```
AI
```

Out[69]:
```
'Artificial intelligence (AI) is a wide-ranging branch of computer science concerned wit
h building smart machines capable of performing tasks that typically require human intel
ligence. AI is an interdisciplinary science with multiple approaches, but advancements i
n machine learning and deep learning are creating a paradigm shift in virtually every se
ctor of the tech industry. \n\nArtificial intelligence allows machines to model, and eve
n improve upon, the capabilities of the human mind. From the development of self-driving
cars to the proliferation of smart assistants like Siri and Alexa, AI is a growing part
of everyday life. As a result, many tech companies across various industries are investi
ng in artificially intelligent technologies.\n\n '
```

In [73]:
```python
sent = 'khatty is natural when it comes to drawing'
sent_tokens = word_tokenize(sent)
sent_tokens
```

Out[73]:
```
['khatty', 'is', 'natural', 'when', 'it', 'comes', 'to', 'drawing']
```

In [75]:
```python
for token in sent_tokens:
```

```
        print(nltk.pos_tag([token]))
```

```
[('khatty', 'NN')]
[('is', 'VBZ')]
[('natural', 'JJ')]
[('when', 'WRB')]
[('it', 'PRP')]
[('comes', 'VBZ')]
[('to', 'TO')]
[('drawing', 'VBG')]
```

In [6]:
```python
sent2 = 'john is eating a delicious cake'
sent2_tokens = word_tokenize(sent2)
for token in sent2_tokens:
        print(nltk.pos_tag([token]))
```

```
[('john', 'NN')]
[('is', 'VBZ')]
[('eating', 'VBG')]
[('a', 'DT')]
[('delicious', 'JJ')]
[('cake', 'NN')]
```

In [7]:
```python
from nltk import ne_chunk
```

In [8]:
```python
Ne_sent='The US president stays in the WHITEHOUSE'
```

In [10]:
```python
Ne_tokens = word_tokenize(Ne_sent)
Ne_tokens
```

Out[10]:
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [11]:
```python
Ne_tags = nltk.pos_tag(Ne_tokens)
Ne_tags
```

Out[11]:
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [13]:
```python
NE_NER = ne_chunk(Ne_tags)
print(NE_NER)
```

```
(S
  The/DT
  (GSP US/NNP)
  president/NN
  stays/NNS
  in/IN
  the/DT
  (ORGANIZATION WHITEHOUSE/NNP))
```

In [19]:
```python
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [18]:
```python
!pip install wordcloud
```

```
Collecting wordcloud
  Downloading wordcloud-1.8.2.2-cp39-cp39-win_amd64.whl (153 kB)
     ---------------------------------- 153.1/153.1 kB 652.2 kB/s eta 0:00:00
Requirement already satisfied: pillow in c:\users\91996\anaconda3\lib\site-packages (fro
m wordcloud) (9.2.0)
```

```
Requirement already satisfied: numpy>=1.6.1 in c:\users\91996\anaconda3\lib\site-package
s (from wordcloud) (1.21.5)
Requirement already satisfied: matplotlib in c:\users\91996\anaconda3\lib\site-packages
(from wordcloud) (3.5.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\91996\anaconda3\lib\site
-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\91996\anaconda3\lib\site-pac
kages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\91996\anaconda3\lib\site-pa
ckages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\users\91996\anaconda3\lib\site-pack
ages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: cycler>=0.10 in c:\users\91996\anaconda3\lib\site-package
s (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\91996\anaconda3\lib\site-pa
ckages (from matplotlib->wordcloud) (1.4.2)
Requirement already satisfied: six>=1.5 in c:\users\91996\anaconda3\lib\site-packages (f
rom python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.2.2
```

In [2]:
```python
text = (" Python Python Python   KodiMatplotlib Matplotlib Seaborn Network Plot Violin C
```

In [25]:
```python
text
```

Out[25]:
```
' Python Python Python Matplotlib Matplotlib Seaborn Network Plot Violin Chart Pandas Da
tascience Wordcloud Spider Radar Parrallel Alpha Color Brewer Density Scatter Barplot Ba
rplot Boxplot Violinplot Treemap Stacked Area Chart ChartVisualization Dataviz Donut Pie
Time-Series Wordcloud Wordcloud Sankey Bubble'
```

In [30]:

In [39]:
```python
wordcloud = WordCloud(width = 480, height=480, margin = 0,background_color='red').genera
# Display the generated image:
plt.imshow(wordcloud,interpolation='bilinear')
plt.axis('off')
plt.margins(x=0,y=0)
plt.show()
```


In [ ]: