In [107]:
```
import os
import nltk
nltk.download()
```

```
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
(https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)
```

Out[107]: True

In [108]:
```
# you can create your own words
nlp='''Natural language processing (NLP) refers to the branch of
computer science—and more specifically, the branch of artificial
intelligence or AI—concerned with giving computers the ability to
understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics—rule-based modeling of human
language—with statistical, machine learning, and deep learning models.
Together, these technologies enable computers to process human language
in the form of text or voice data and to ‘understand’ its full meaning,
complete with the speaker or writer’s intent and sentiment.'''
```

In [109]:
```
nlp
```

Out[109]:
```
'Natural language processing (NLP) refers to the branch of \ncomputer science—and
more specifically, the branch of artificial \nintelligence or AI—concerned with gi
ving computers the ability to \nunderstand text and spoken words in much the same
way human beings can.\n\nNLP combines computational linguistics—rule-based modelin
g of human \nlanguage—with statistical, machine learning, and deep learning model
s. \nTogether, these technologies enable computers to process human language \nin
the form of text or voice data and to ‘understand’ its full meaning, \ncomplete wi
th the speaker or writer’s intent and sentiment.'
```

In [110]:
```
type(nlp)
```

Out[110]: str

word_tokenize

In [111]:
```
from nltk.tokenize import word_tokenize
```

In [112]:
```
nlp_tokens=word_tokenize(nlp)
```

In [113]:
```
nlp_tokens
```

Out[113]:
```
['Natural',
 'language',
 'processing',
 '(',
 'NLP',
 ')',
 'refers',
 'to',
 'the',
 'branch',
 'of',
 'computer',
 'science—and',
 'more',
 'specifically',
 ',',
 'the',
 'branch',
 'of',
 'artificial',
 'intelligence',
 'or',
 'AI—concerned',
 'with',
 'giving',
 'computers',
 'the',
 'ability',
 'to',
 'understand',
 'text',
 'and',
 'spoken',
 'words',
 'in',
 'much',
 'the',
 'same',
 'way',
 'human',
 'beings',
 'can',
 '.',
 'NLP',
 'combines',
 'computational',
 'linguistics—rule-based',
 'modeling',
 'of',
 'human',
 'language—with',
 'statistical',
 ',',
 'machine',
 'learning',
 ',',
 'and',
 'deep',
```

```
 'learning',
 'models',
 '.',
 'Together',
 ',',
 'these',
 'technologies',
 'enable',
 'computers',
 'to',
 'process',
 'human',
 'language',
 'in',
 'the',
 'form',
 'of',
 'text',
 'or',
 'voice',
 'data',
 'and',
 'to',
 '‘',
 'understand',
 '’',
 'its',
 'full',
 'meaning',
 ',',
 'complete',
 'with',
 'the',
 'speaker',
 'or',
 'writer',
 '’',
 's',
 'intent',
 'and',
 'sentiment',
 '.']
```

In [114]:
```
len(nlp_tokens)
```

Out[114]: 100

sent_tokenize

In [115]:
```
from nltk.tokenize import sent_tokenize
```

In [116]:
```
nlp_sen=sent_tokenize(nlp)
```

In [117]:
```
nlp_sen
```

Out[117]:
```
['Natural language processing (NLP) refers to the branch of \ncomputer science—and
more specifically, the branch of artificial \nintelligence or AI—concerned with gi
ving computers the ability to \nunderstand text and spoken words in much the same
way human beings can.',
 'NLP combines computational linguistics—rule-based modeling of human \nlanguage—w
ith statistical, machine learning, and deep learning models.',
 'Together, these technologies enable computers to process human language \nin the
form of text or voice data and to ‘understand’ its full meaning, \ncomplete with t
he speaker or writer’s intent and sentiment.']
```

In [118]:
```
len(nlp_sen)
```

Out[118]:
```
3
```

blankline_tokenize

In [119]:
```
from nltk.tokenize import blankline_tokenize
```

In [120]:
```
nlp_blank=blankline_tokenize(nlp)
```

In [121]:
```
nlp_blank
```

Out[121]:
```
['Natural language processing (NLP) refers to the branch of \ncomputer science—and
more specifically, the branch of artificial \nintelligence or AI—concerned with gi
ving computers the ability to \nunderstand text and spoken words in much the same
way human beings can.',
 'NLP combines computational linguistics—rule-based modeling of human \nlanguage—w
ith statistical, machine learning, and deep learning models. \nTogether, these tec
hnologies enable computers to process human language \nin the form of text or voic
e data and to ‘understand’ its full meaning, \ncomplete with the speaker or write
r’s intent and sentiment.']
```

In [122]:
```
len(nlp_blank)
```

Out[122]:
```
2
```

WhitespaceTokenizer

In [123]:
```
from nltk.tokenize import WhitespaceTokenizer
```

In [124]:
```
nlp_ws2=WhitespaceTokenizer().tokenize(nlp)
nlp_ws2
```

Out[124]:
```
['Natural',
 'language',
 'processing',
 '(NLP)',
 'refers',
 'to',
 'the',
 'branch',
 'of',
 'computer',
 'science—and',
 'more',
 'specifically,',
 'the',
 'branch',
 'of',
 'artificial',
 'intelligence',
 'or',
 'AI—concerned',
 'with',
 'giving',
 'computers',
 'the',
 'ability',
 'to',
 'understand',
 'text',
 'and',
 'spoken',
 'words',
 'in',
 'much',
 'the',
 'same',
 'way',
 'human',
 'beings',
 'can.',
 'NLP',
 'combines',
 'computational',
 'linguistics—rule-based',
 'modeling',
 'of',
 'human',
 'language—with',
 'statistical,',
 'machine',
 'learning,',
 'and',
 'deep',
 'learning',
 'models.',
 'Together,',
 'these',
 'technologies',
 'enable',
```

```
 'computers',
 'to',
 'process',
 'human',
 'language',
 'in',
 'the',
 'form',
 'of',
 'text',
 'or',
 'voice',
 'data',
 'and',
 'to',
 '‘understand’',
 'its',
 'full',
 'meaning,',
 'complete',
 'with',
 'the',
 'speaker',
 'or',
 'writer’s',
 'intent',
 'and',
 'sentiment.']
```

In [125]:
```
len(ws_1)
```

Out[125]: 86

wordpunct_tokenize

In [169]:
```
from nltk.tokenize import wordpunct_tokenize
```

In [178]:
```
sen2='the best and most beautifull thing in the world  $30.49rs'
```

In [179]:
```
wpt=wordpunct_tokenize(sen2)
```

In [180]:
```
wpt
```

Out[180]:
```
['the',
 'best',
 'and',
 'most',
 'beautifull',
 'thing',
 'in',
 'the',
 'world',
 '$',
 '30',
 '.',
 '49rs']
```

NEXT WE WILL SEE HOW WE WILL USE UNI-GRAM,BI-GRAM,TRI-GRAM USING NLTK

In [126]:
```
from nltk.util import bigrams,trigrams,ngrams
```

In [127]:
```
sentence = 'NO MATTER HOW HARD OR IMPOSSIBLE IT IS, NEVER LOSE SIGHT OF YOUR GOAL.'
```

In [128]:
```
sentence
```

Out[128]:
```
'NO MATTER HOW HARD OR IMPOSSIBLE IT IS, NEVER LOSE SIGHT OF YOUR GOAL.'
```

In [129]:
```
quotes_tokens=word_tokenize(sentence)
```

In [130]:
```
quotes_tokens
```

Out[130]:
```
['NO',
 'MATTER',
 'HOW',
 'HARD',
 'OR',
 'IMPOSSIBLE',
 'IT',
 'IS',
 ',',
 'NEVER',
 'LOSE',
 'SIGHT',
 'OF',
 'YOUR',
 'GOAL',
 '.']
```

In [131]:
```
len(quotes_tokens)
```

Out[131]: 16

In [132]:
```
quotes_big=list(bigrams(quotes_tokens))
```

In [133]:
```
quotes_big
```

Out[133]:
```
[('NO', 'MATTER'),
 ('MATTER', 'HOW'),
 ('HOW', 'HARD'),
 ('HARD', 'OR'),
 ('OR', 'IMPOSSIBLE'),
 ('IMPOSSIBLE', 'IT'),
 ('IT', 'IS'),
 ('IS', ','),
 (',', 'NEVER'),
 ('NEVER', 'LOSE'),
 ('LOSE', 'SIGHT'),
 ('SIGHT', 'OF'),
 ('OF', 'YOUR'),
 ('YOUR', 'GOAL'),
 ('GOAL', '.')]
```

In [134]:
```
1 len(quotes_big)
```

Out[134]: 15

In [135]:
```
quotes_tri=list(trigrams(quotes_tokens))
```

In [136]:
```
quotes_tri
```

Out[136]:
```
[('NO', 'MATTER', 'HOW'),
 ('MATTER', 'HOW', 'HARD'),
 ('HOW', 'HARD', 'OR'),
 ('HARD', 'OR', 'IMPOSSIBLE'),
 ('OR', 'IMPOSSIBLE', 'IT'),
 ('IMPOSSIBLE', 'IT', 'IS'),
 ('IT', 'IS', ','),
 ('IS', ',', 'NEVER'),
 (',', 'NEVER', 'LOSE'),
 ('NEVER', 'LOSE', 'SIGHT'),
 ('LOSE', 'SIGHT', 'OF'),
 ('SIGHT', 'OF', 'YOUR'),
 ('OF', 'YOUR', 'GOAL'),
 ('YOUR', 'GOAL', '.')]
```

In [137]:
```
len(quotes_tri)
```

Out[137]: 14

In [138]:
```
quotes_ngr=list(ngrams(quotes_tokens))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[138], line 1
----> 1 quotes_ngr=list(ngrams(quotes_tokens))

TypeError: ngrams() missing 1 required positional argument: 'n'
```

In [139]:
```
quotes_ngr=list(ngrams(quotes_tokens,4))
```

In [140]:
```
quotes_ngr
```

Out[140]:
```
[('NO', 'MATTER', 'HOW', 'HARD'),
 ('MATTER', 'HOW', 'HARD', 'OR'),
 ('HOW', 'HARD', 'OR', 'IMPOSSIBLE'),
 ('HARD', 'OR', 'IMPOSSIBLE', 'IT'),
 ('OR', 'IMPOSSIBLE', 'IT', 'IS'),
 ('IMPOSSIBLE', 'IT', 'IS', ','),
 ('IT', 'IS', ',', 'NEVER'),
 ('IS', ',', 'NEVER', 'LOSE'),
 (',', 'NEVER', 'LOSE', 'SIGHT'),
 ('NEVER', 'LOSE', 'SIGHT', 'OF'),
 ('LOSE', 'SIGHT', 'OF', 'YOUR'),
 ('SIGHT', 'OF', 'YOUR', 'GOAL'),
 ('OF', 'YOUR', 'GOAL', '.')]
```

In [141]:
```
quotes_ngr1=list(ngrams(quotes_tokens,6))
```

In [106]:
```
quotes_ngr1
```

Out[106]:
```
[('NO', 'MATTER', 'HOW', 'HARD', 'OR', 'IMPOSSIBLE'),
 ('MATTER', 'HOW', 'HARD', 'OR', 'IMPOSSIBLE', 'IT'),
 ('HOW', 'HARD', 'OR', 'IMPOSSIBLE', 'IT', 'IS'),
 ('HARD', 'OR', 'IMPOSSIBLE', 'IT', 'IS', ','),
 ('OR', 'IMPOSSIBLE', 'IT', 'IS', ',', 'NEVER'),
 ('IMPOSSIBLE', 'IT', 'IS', ',', 'NEVER', 'LOSE'),
 ('IT', 'IS', ',', 'NEVER', 'LOSE', 'SIGHT'),
 ('IS', ',', 'NEVER', 'LOSE', 'SIGHT', 'OF'),
 (',', 'NEVER', 'LOSE', 'SIGHT', 'OF', 'YOUR'),
 ('NEVER', 'LOSE', 'SIGHT', 'OF', 'YOUR', 'GOAL'),
 ('LOSE', 'SIGHT', 'OF', 'YOUR', 'GOAL', '.')]
```

porter-stemmer

In [142]:
```
from nltk.stem import PorterStemmer
```

In [145]:
```
PorterStemmer().stem("affection")
```

Out[145]:
```
'affect'
```

In [146]:
```
por=PorterStemmer()
```

In [147]:
```
por.stem("working")
```

Out[147]:
```
'work'
```

In [148]:
```
por.stem("playing")
```

Out[148]:
```
'play'
```

In [149]:
```
por.stem('give')
```

Out[149]:
```
'give'
```

In [150]:
```
words_arr=['give','giving','given','gave']
```

In [154]:
```
for words in words_arr:
    print(words,":",por.stem(words))
```

```
give : give
giving : give
given : given
gave : gave
```

In [155]:
```
words_arr2=['give','giving','given','gave','thinking', 'loving', 'final', 'finalize
```

In [156]:
```
for words in words_arr2:
    print(words,":",por.stem(words))
```

```
give : give
giving : give
given : given
gave : gave
thinking : think
loving : love
final : final
finalized : final
finally : final
```

LancasterStemmer

In [157]:
```
from nltk.stem import LancasterStemmer
# lancasterstemmer is more aggresive then the porterstemmer
```

In [158]:
```
las=LancasterStemmer()
```

In [160]:
```
for words in words_arr2:
    print(words,":",las.stem(words))
```

```
give : giv
giving : giv
given : giv
gave : gav
thinking : think
loving : lov
final : fin
finalized : fin
finally : fin
```

SnowballStemmer

In [162]:
```
from nltk.stem import SnowballStemmer
#snowball stemmer is same as portstemmer
```

In [164]:
```
sns=SnowballStemmer('english')
```

In [165]:
```
for words in words_arr2:
    print(words,":",sns.stem(words))
```

```
give : give
giving : give
given : given
gave : gave
thinking : think
loving : love
final : final
finalized : final
finally : final
```

WordNetLemmatizer

In [181]:
```
from nltk.stem import WordNetLemmatizer
```

In [182]:
```
wnt=WordNetLemmatizer()
```

In [183]:
```
for words in words_arr2:
    print(words,":",wnt.lemmatize(words))
```

```
give : give
giving : giving
given : given
gave : gave
thinking : thinking
loving : loving
final : final
finalized : finalized
finally : finally
```

In [185]:
```
por.stem('final')
```

Out[185]: 'final'

In [186]:
```
las.stem('finally')
```

Out[186]: 'fin'

In [187]:
```
sns.stem('finalized')
```

Out[187]: 'final'

In [189]:
```
las.stem('final')
```

Out[189]: 'fin'

In [190]:
```
las.stem('finalized')
```

Out[190]: 'fin'

stopwords

#there is other concept called POS (part of speech) which deals with subject, noun, pronoun but before of this lets go with other concept called STOPWORDS #STOPWORDS = i, is, as,at, on, about & nltk has their own list of stopewords

In [192]:
```
from nltk.corpus import stopwords
```

In [195]:
```
stopwords.words("english")
```

Out[195]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
```

In [196]:
```
len(stopwords.words("english"))
```

Out[196]:
```
179
```

In [197]:
```
stopwords.words("spanish")
```

Out[197]:
```
['de',
 'la',
 'que',
 'el',
 'en',
 'y',
 'a',
 'los',
 'del',
 'se',
 'las',
 'por',
 'un',
 'para',
 'con',
 'no',
 'una',
 'su',
 'al',
```

In [198]:
```
len(stopwords.words("spanish"))
```

Out[198]:
```
313
```

In [200]:
```
stopwords.words("chinese")
```

```
'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',
```

In [201]:
```
len(stopwords.words("chinese"))
```

Out[201]: 841

In [ ]:
```
stopwords.words('hindi') # research phase
```

pos_tag([part of sppech])

In [214]:
```
from nltk import pos_tag
```

In [215]:
```
sent = 'janu is a natural when it comes to drawing'
```

In [216]:
```
s_token=word_tokenize(sent)
```

In [217]:
```
s_token
```

Out[217]: ['janu', 'is', 'a', 'natural', 'when', 'it', 'comes', 'to', 'drawing']

In [222]:
```
for token in s_token:
    print(pos_tag([token]))
```

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

In [225]:
```
sent2 = 'jadu is eating a delicious cake'
s_token2=word_tokenize(sent2)
for token in s_token2:
    print(pos_tag([token]))
```

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

NER (NAMED ENTITIY RECOGNITION)

In [226]:
```
from nltk import ne_chunk
```

In [227]:
```
NE_sent = 'The US president stays in the WHITEHOUSE '
```

In [228]:
```
NE_tokens = word_tokenize(NE_sent)
```

In [229]:
```
NE_tokens
```

Out[229]:
```
['The', 'US', 'president', 'stays', 'in', 'the', 'WHITEHOUSE']
```

In [230]:
```
NE_tags =pos_tag(NE_tokens)
NE_tags
```

Out[230]:
```
[('The', 'DT'),
 ('US', 'NNP'),
 ('president', 'NN'),
 ('stays', 'NNS'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('WHITEHOUSE', 'NNP')]
```

In [233]:
```
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 [245]:
```
new = 'the big cat ate the little mouse who was after fresh cheese'
NE_tokens = word_tokenize(new)
print(NE_tokens)
NE_NER1=pos_tag(NE_tokens)
print(NE_NER1)
```

```
['the', 'big', 'cat', 'ate', 'the', 'little', 'mouse', 'who', 'was', 'after', 'fre
sh', 'cheese']
[('the', 'DT'), ('big', 'JJ'), ('cat', 'NN'), ('ate', 'VBD'), ('the', 'DT'), ('lit
tle', 'JJ'), ('mouse', 'NN'), ('who', 'WP'), ('was', 'VBD'), ('after', 'IN'), ('fr
esh', 'JJ'), ('cheese', 'NN')]
```

In [246]:
```
NE_NER2 = ne_chunk(NE_NER1)
print(NE_NER2)
```

```
(S
  the/DT
  big/JJ
  cat/NN
  ate/VBD
  the/DT
  little/JJ
  mouse/NN
  who/WP
  was/VBD
  after/IN
  fresh/JJ
  cheese/NN)
```

wordcloud

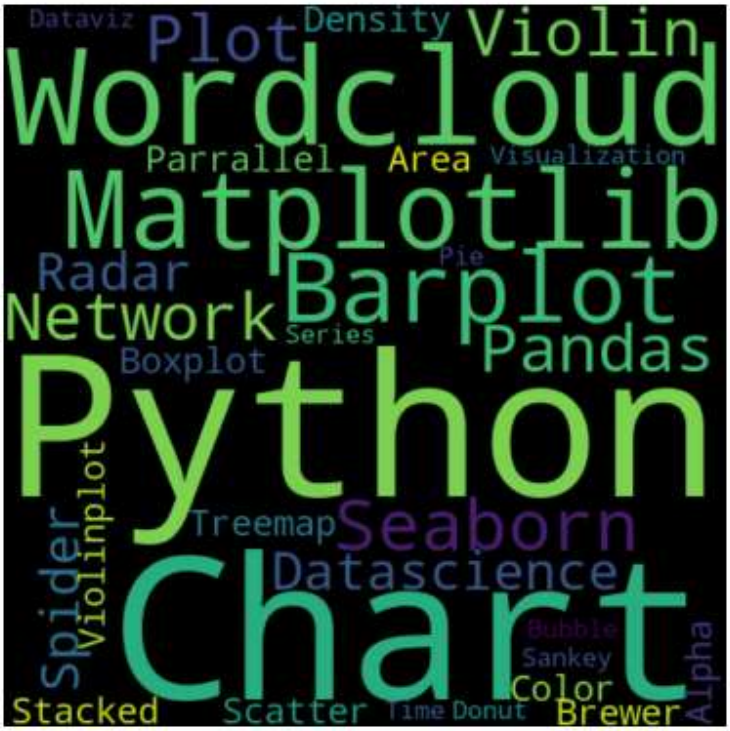
In [ ]:
```
pip install wordcloud
```

In [251]:
```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

In [252]:
```
art Visualization Dataviz Donut Pie Time-Series Wordcloud Wordcloud Sankey Bubble")
```

In [273]:
```
wordcloud = WordCloud(width=480, height=480, margin=0).generate(text)
```

In [274]:
```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
plt.show()
```



In [269]:
```
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(tex
```

In [270]:
```
wordcloud
```

Out[270]:
```
<wordcloud.wordcloud.WordCloud at 0x1e36abd9a50>
```

In [272]:
```
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.margins(x=0, y=0)
```

In [ ]:
```
```