In [1]:
```
from nltk.tokenize import sent_tokenize
```

In [2]:
```
file=open('nlp.txt','r')
text=file.read()
print(text)
```

```
Natural language processing refers to the branch of computer science and more specifically, the branch of artificial intelligence or AI, c
oncerned with giving computers the ability to understand text and spoken words in much the same way human beings can. NLP combines computa
tional linguistics 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 sent
iment.
```

In [3]:
```
sentences=sent_tokenize(text)
```

In [4]:
```
print("number of sentences:",len(sentences))
for i in  range(len(sentences)):
    print("\nSentence",i+1,":\n",sentences[i])
```

```
number of sentences: 3

Sentence 1 :
 Natural language processing 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.

Sentence 2 :
 NLP combines computational linguistics with statistical, machine learning, and deep learning models.

Sentence 3 :
 Together, these technologies enable computers to process human language in the form of text or voice data and to understand its full mean
ing, complete with the speaker or writer's intent and sentiment.
```

In [5]:
```
from nltk.tokenize import word_tokenize
```

In [6]:
```
words=word_tokenize(text)
```

In [7]:
```
print("total number of words:",len(words))
print(words)
```

```
total number of words: 94
['Natural', 'language', 'processing', 'refers', 'to', 'the', 'branch', 'of', 'computer', 'science', 'and', 'more', 'specifically', ',', 't
he', 'branch', 'of', 'artificial', 'intelligence', 'or', 'AI', ',', 'concerned', 'with', 'giving', 'computers', 'the', 'ability', 'to', 'u
nderstand', 'text', 'and', 'spoken', 'words', 'in', 'much', 'the', 'same', 'way', 'human', 'beings', 'can', '.', 'NLP', 'combines', 'compu
tational', 'linguistics', 'with', 'statistical', ',', 'machine', 'learning', ',', 'and', 'deep', 'learning', 'models', '.', 'Together',
',', 'these', 'technologies', 'enable', 'computers', 'to', 'process', 'human', 'language', 'in', 'the', 'form', 'of', 'text', 'or', 'voic
e', 'data', 'and', 'to', 'understand', 'its', 'full', 'meaning', ',', 'complete', 'with', 'the', 'speaker', 'or', 'writer', "'s", 'inten
t', 'and', 'sentiment', '.']
```

In [8]:
```
words=word_tokenize(text,preserve_line=True)
len(words)
```

Out[8]:
```
92
```

In [9]:
```
from nltk.tokenize import word_tokenize
```

In [10]:
```
file=open('opinion.txt','r')
text=file.read()
```

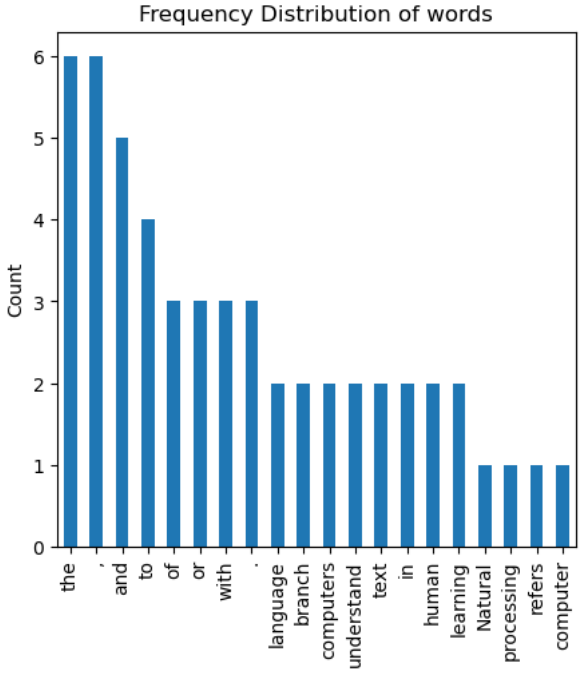
In [11]:
```
words=word_tokenize(text)
len(words)
```

Out[11]:
```
93
```

In [12]:
```
from nltk.probability import FreqDist
all_fdist=FreqDist(words).most_common(20)
print(all_fdist)
```

```
[('the', 6), (',', 6), ('and', 5), ('to', 4), ('of', 3), ('or', 3), ('with', 3), ('.', 3), ('language', 2), ('branch', 2), ('computers',
2), ('understand', 2), ('text', 2), ('in', 2), ('human', 2), ('learning', 2), ('Natural', 1), ('processing', 1), ('refers', 1), ('compute
r', 1)]
```

In [13]:
```
import matplotlib.pyplot as plt
import pandas as pd
all_fdist=pd.Series(dict(all_fdist))
fig,ax=plt.subplots(figsize=(5,5))
all_fdist.plot(kind='bar')
plt.title('Frequency Distribution of words')
plt.ylabel('Count')
plt.savefig('a.jpg')
```



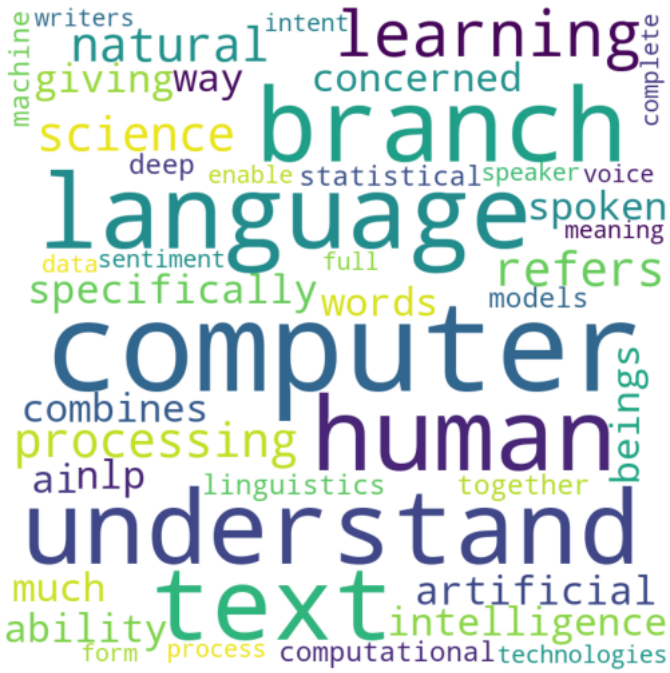
In [14]:
```
text=text.lower()
```

In [15]:
```
import re
text=re.sub('[^A-Za-z0-9]+',' ',text)
```

In [16]:
```
text=re.sub('\S*\d\S*','',text).strip()
```

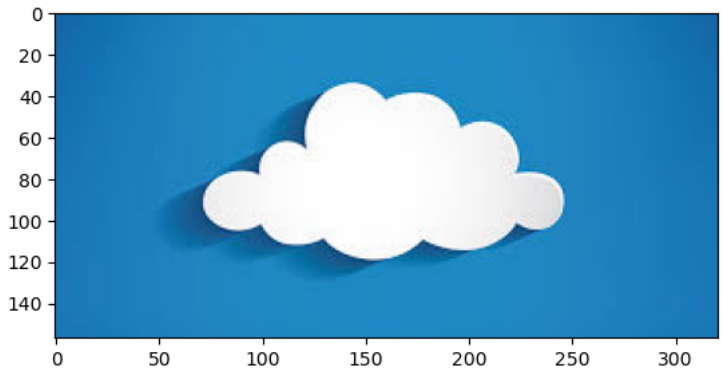
In [19]:
```
from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
stopwords=set(STOPWORDS)
wordcloud=WordCloud(width=800,height=800,
            background_color='white',
            stopwords=stopwords,min_font_size=10).generate(text)

plt.figure(figsize=(5,5),facecolor=None)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



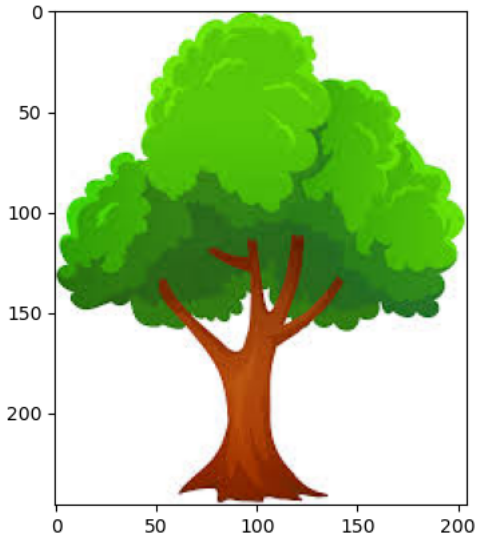
In [20]:
```
from skimage.io import imread
cloud=imread('cloud.png')
plt.imshow(cloud)
```

Out[20]: <matplotlib.image.AxesImage at 0x206d1bff4c0>



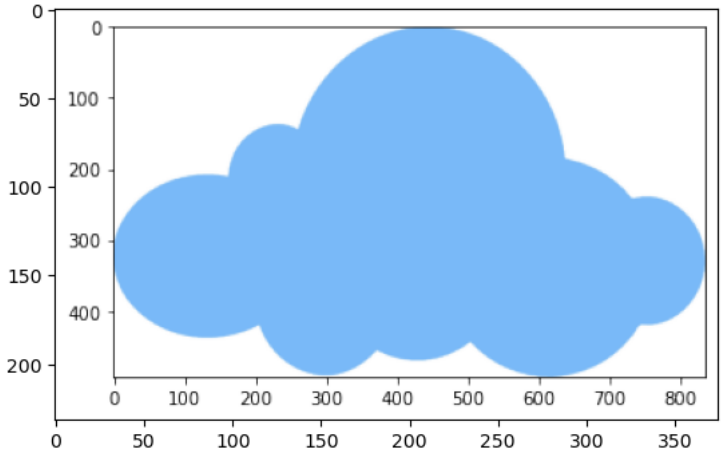
In [21]:
```
from skimage.io import imread
tree=imread('tree.jpeg')
plt.imshow(tree)
```

Out[21]: <matplotlib.image.AxesImage at 0x206d1d9b7c0>



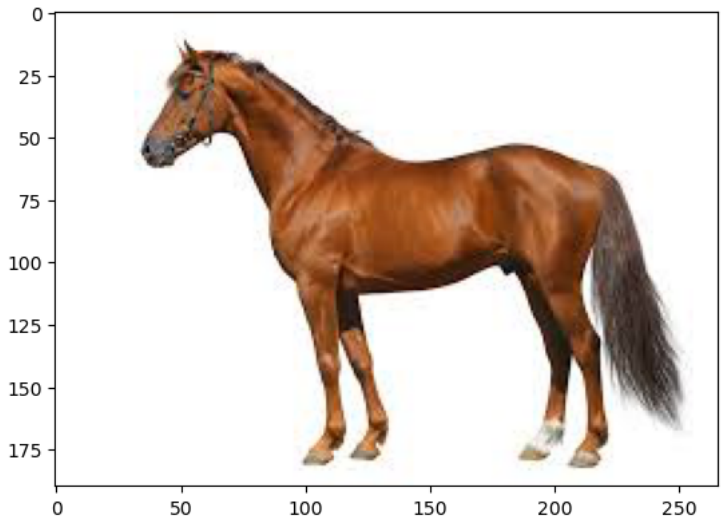
In [22]:
```
from skimage.io import imread
cloud1=imread('cloud1.png')
plt.imshow(cloud1)
```

Out[22]: <matplotlib.image.AxesImage at 0x206ce3792b0>



In [23]:
```
from skimage.io import imread
horse=imread('horse.png')
plt.imshow(horse)
```

Out[23]:
```
<matplotlib.image.AxesImage at 0x206ce3c8b20>
```



In [24]:
```
from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
stopwords=set(STOPWORDS)
wordcloud=WordCloud(width=800,height=800,
            background_color='white',
            stopwords=stopwords,min_font_size=10,mask=cloud).generate(text)

plt.figure(figsize=(5,5),facecolor=None)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
plt.savefig('abc.png')
```

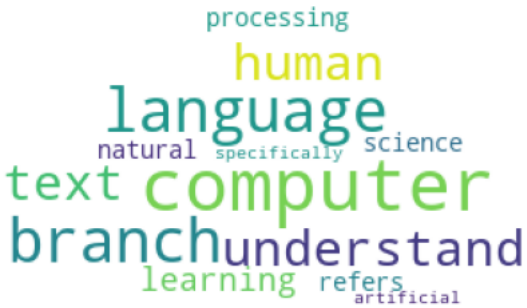

```
<Figure size 640x480 with 0 Axes>
```

In [25]:
```
from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
stopwords=set(STOPWORDS)
wordcloud=WordCloud(width=800,height=800,
            background_color='white',
            stopwords=stopwords,min_font_size=10,mask=horse).generate(text)

plt.figure(figsize=(5,5),facecolor=None)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

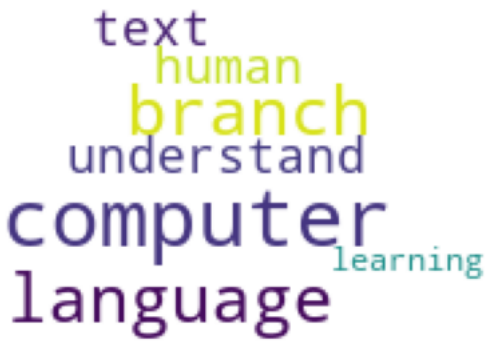
In [26]:
```
from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
stopwords=set(STOPWORDS)
wordcloud=WordCloud(width=800,height=800,
            background_color='white',
            stopwords=stopwords,min_font_size=10,mask=cloud1).generate(text)

plt.figure(figsize=(5,5),facecolor=None)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



In [27]:
```
from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
stopwords=set(STOPWORDS)
wordcloud=WordCloud(width=800,height=800,
            background_color='white',
            stopwords=stopwords,min_font_size=10,mask=tree).generate(text)

plt.figure(figsize=(5,5),facecolor=None)
plt.imshow(wordcloud)
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



SpellCorrection

In [28]:
```
import nltk
from nltk.metrics.distance import edit_distance
```

In [29]:
```
nltk.download('words')
from nltk.corpus import words
cw=words.words()
```

```
[nltk_data] Error loading words: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
```

In [30]:
```
iw=['happpy','amzzzzzing','intelliegent']
for word in iw:
    temp=[(edit_distance(word,w),w)for w in cw if w[0]==word[0]]
    print(sorted(temp,key=lambda val:val[0])[0][1])
```

```
happy
amazing
intelligent
```

Stemming

In [31]:
```
from nltk.tokenize import word_tokenize
file=open('nlp.txt','r')
text=file.read()
text=text.lower()
import re
text=re.sub('[^A-Za-z0-9]',' ',text)
text=re.sub('\s*\d\s*','',text).strip()
print(text)
```

```
natural language processing refers to the branch of computer science and more specifically  the branch of artificial intelligence or ai  c
oncerned with giving computers the ability to understand text and spoken words in much the same way human beings can  nlp combines computa
tional linguistics 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 sent
iment
```

In [33]:
```
words=word_tokenize(text,preserve_line=True)
print(words)
```

```
['natural', 'language', 'processing', 'refers', 'to', 'the', 'branch', 'of', 'computer', 'science', 'and', 'more', 'specifically', 'the',
'branch', 'of', 'artificial', 'intelligence', 'or', 'ai', 'concerned', 'with', 'giving', 'computers', 'the', 'ability', 'to', 'understan
d', 'text', 'and', 'spoken', 'words', 'in', 'much', 'the', 'same', 'way', 'human', 'beings', 'can', 'nlp', 'combines', 'computational', 'l
inguistics', 'with', 'statistical', 'machine', 'learning', 'and', 'deep', 'learning', 'models', 'together', 'these', 'technologies', 'enab
le', 'computers', 'to', 'process', 'human', 'language', 'in', 'the', 'form', 'of', 'text', 'or', 'voice', 'data', 'and', 'to', 'understan
d', 'its', 'full', 'meaning', 'complete', 'with', 'the', 'speaker', 'or', 'writer', 's', 'intent', 'and', 'sentiment']
```

In [35]:
```
from nltk.stem import PorterStemmer
ps=PorterStemmer()
ps_stem_sent=[ps.stem(words_sent)for words_sent in words]
print(ps_stem_sent)
```

```
['natur', 'languag', 'process', 'refer', 'to', 'the', 'branch', 'of', 'comput', 'scienc', 'and', 'more', 'specif', 'the', 'branch', 'of',
'artifici', 'intellig', 'or', 'ai', 'concern', 'with', 'give', 'comput', 'the', 'abil', 'to', 'understand', 'text', 'and', 'spoken', 'wor
d', 'in', 'much', 'the', 'same', 'way', 'human', 'be', 'can', 'nlp', 'combin', 'comput', 'linguist', 'with', 'statist', 'machin', 'learn',
'and', 'deep', 'learn', 'model', 'togeth', 'these', 'technolog', 'enabl', 'comput', 'to', 'process', 'human', 'languag', 'in', 'the', 'for
m', 'of', 'text', 'or', 'voic', 'data', 'and', 'to', 'understand', 'it', 'full', 'mean', 'complet', 'with', 'the', 'speaker', 'or', 'write
r', 's', 'intent', 'and', 'sentiment']
```

Lemmatization

In [36]:
```
print(words)
```

```
['natural', 'language', 'processing', 'refers', 'to', 'the', 'branch', 'of', 'computer', 'science', 'and', 'more', 'specifically', 'the',
'branch', 'of', 'artificial', 'intelligence', 'or', 'ai', 'concerned', 'with', 'giving', 'computers', 'the', 'ability', 'to', 'understan
d', 'text', 'and', 'spoken', 'words', 'in', 'much', 'the', 'same', 'way', 'human', 'beings', 'can', 'nlp', 'combines', 'computational', 'l
inguistics', 'with', 'statistical', 'machine', 'learning', 'and', 'deep', 'learning', 'models', 'together', 'these', 'technologies', 'enab
le', 'computers', 'to', 'process', 'human', 'language', 'in', 'the', 'form', 'of', 'text', 'or', 'voice', 'data', 'and', 'to', 'understan
d', 'its', 'full', 'meaning', 'complete', 'with', 'the', 'speaker', 'or', 'writer', 's', 'intent', 'and', 'sentiment']
```

In [38]:
```
from nltk.stem.wordnet import WordNetLemmatizer
l=WordNetLemmatizer()
ls=[l.lemmatize(words_sent)for words_sent in words]
print(ls)
```

```
['natural', 'language', 'processing', 'refers', 'to', 'the', 'branch', 'of', 'computer', 'science', 'and', 'more', 'specifically', 'the',
'branch', 'of', 'artificial', 'intelligence', 'or', 'ai', 'concerned', 'with', 'giving', 'computer', 'the', 'ability', 'to', 'understand',
'text', 'and', 'spoken', 'word', 'in', 'much', 'the', 'same', 'way', 'human', 'being', 'can', 'nlp', 'combine', 'computational', 'linguist
ics', 'with', 'statistical', 'machine', 'learning', 'and', 'deep', 'learning', 'model', 'together', 'these', 'technology', 'enable', 'comp
uter', 'to', 'process', 'human', 'language', 'in', 'the', 'form', 'of', 'text', 'or', 'voice', 'data', 'and', 'to', 'understand', 'it', 'f
ull', 'meaning', 'complete', 'with', 'the', 'speaker', 'or', 'writer', 's', 'intent', 'and', 'sentiment']
```

In [42]:
```
from nltk.stem import WordNetLemmatizer
l=WordNetLemmatizer()
print('rocks:',l.lemmatize('rocks'))
print('corpora:',l.lemmatize('corpora'))
print('better:',l.lemmatize('better',pos='a'))
```

```
rocks: rock
corpora: corpus
better: good
```

Parts of speech tagging

In [43]:
```
import nltk
from nltk import word_tokenize
```

In [45]:
```
text='I am very hungry but stomak is empty'
words=word_tokenize(text)
print('parts of speech:',nltk.pos_tag(words))
```

```
parts of speech: [('I', 'PRP'), ('am', 'VBP'), ('very', 'RB'), ('hungry', 'JJ'), ('but', 'CC'), ('stomak', 'JJ'), ('is', 'VBZ'), ('empty',
'JJ')]
```

Vectorization

In [51]:
```
from sklearn.feature_extraction.text import CountVectorizer
s=['He is smart boy.she is also smart',
  'chirag and man is a smart persons']
```

In [52]:
```
cv=CountVectorizer()
x=cv.fit_transform(s)
x=x.toarray()
v=sorted(cv.vocabulary_.keys())
print(v)
print(x)
```

```
['also', 'and', 'boy', 'chirag', 'he', 'is', 'man', 'persons', 'she', 'smart']
[[1 0 1 0 1 2 0 0 1 2]
 [0 1 0 1 0 1 1 1 0 1]]
```

In [54]:
```
cv=CountVectorizer(ngram_range=(2,2))
x=cv.fit_transform(s)
x=x.toarray()
v=sorted(cv.vocabulary_.keys())
print(v)
print(x)
```

```
['also smart', 'and man', 'boy she', 'chirag and', 'he is', 'is also', 'is smart', 'man is', 'she is', 'smart boy', 'smart persons']
[[1 0 1 0 1 1 1 0 1 1 0]
 [0 1 0 1 0 0 1 1 0 0 1]]
```

In [56]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
s=['corona virus is a highly infectious disease',
  'corona virus affects older people the most',
  'older people are at high risk due to this disease']
```

In [60]:
```
tfidf=TfidfVectorizer()
t=tfidf.fit_transform(s)
import pandas as pd
df=pd.DataFrame(t[0].T.todense(),
                index=tfidf.get_feature_names_out(),
                columns=['TF-IDF'])
df=df.sort_values('TF-IDF',ascending=False)
df
```

Out[60]:

	TF-IDF
is	0.459548
infectious	0.459548
highly	0.459548
disease	0.349498
virus	0.349498
corona	0.349498
due	0.000000
high	0.000000
at	0.000000
are	0.000000
most	0.000000
older	0.000000
people	0.000000
risk	0.000000
the	0.000000
this	0.000000
to	0.000000
affects	0.000000

In [77]:
```
import re
from nltk.util import ngrams
from nltk.tokenize import word_tokenize

text="""Natural language processing refers to the branch of
        computer science concerned with giving computers
        the ability to understand text"""

words=word_tokenize(text)
output=list(ngrams(words,2))
output
```

Out[77]:
```
[('Natural', 'language'),
 ('language', 'processing'),
 ('processing', 'refers'),
 ('refers', 'to'),
 ('to', 'the'),
 ('the', 'branch'),
 ('branch', 'of'),
 ('of', 'computer'),
 ('computer', 'science'),
 ('science', 'concerned'),
 ('concerned', 'with'),
 ('with', 'giving'),
 ('giving', 'computers'),
 ('computers', 'the'),
 ('the', 'ability'),
 ('ability', 'to'),
 ('to', 'understand'),
 ('understand', 'text')]
```

In [78]:
```
import matplotlib.pyplot as plt
x=[1,1,2,3,3,4,5,6,7,7,8,8,9,10,10,9,9,11,11,11,12,12]
y=[1,3,2,3,1,1,3,1,1,3,1,3,3,3,1,1,2,1,3,1,1,3]
plt.plot(x,y,'*--b')
plt.show()
```