:[1] In

```
from nltk.stem import PorterStemmer
```

:[2] In

```
import nltk
from nltk.stem import PorterStemmer
nltk.download('punkt')
porter = PorterStemmer()
l_words = ['dogs', 'programming', 'programs', 'programmed', 'cakes', 'indices', 'matrices']
for word in l_words:
    print(f'{word} \t -> {porter.stem(word)}'.expandtabs(15))
```

```
dogs            -> dog
programming     -> program
programs        -> program
programmed      -> program
cakes           -> cake
indices         -> indic
matrices        -> matric

nltk_data] Downloading package punkt to]
...nltk_data]     C:\Users\D7me_\AppData\Roaming\nltk_data]
!nltk_data]   Package punkt is already up-to-date]
```

:[3] In

```
from nltk.tokenize import TreebankWordTokenizer

sentence = '''A stemmer for English operating on the stem cat sh
ould identify such strings as cats, catlike, and catty. A stem
ming algorithm might also reduce the words fishing, fished, an
d fisher to the stem fish. The stem need not be a word, for ex
ample the Porter algorithm reduces, argue, argued, argues, arg
uing, and argus to the stem argu.'''

## tokenize the sentence
list = nltk.word_tokenize(sentence)

## print each word in the sentence before and after Stemming
for word in list:
    print(f'{word} \t -> {porter.stem(word)}'.expandtabs(15))
```

```
A               -> A
stemmer         -> stemmer
for             -> for
English         -> english
operating       -> oper
on              -> on
the             -> the
stem            -> stem
cat             -> cat
sh              -> sh
ould            -> ould
identify        -> identifi
such            -> such
strings         -> string
as              -> as
cats            -> cat
, <-              ,
catlike         -> catlik
, <-              ,
and             -> and
catty           -> catti
. <-              .
A               -> A
stem            -> stem
ming            -> ming
algorithm       -> algorithm
might           -> might
also            -> also
reduce          -> reduc
the             -> the
words           -> word
fishing         -> fish
, <-              ,
fished          -> fish
, <-              ,
an              -> an
d               -> d
fisher          -> fisher
to              -> to
the             -> the
stem            -> stem
```

```
fish            -> fish
. <-              .
The             -> the
stem            -> stem
need            -> need
not             -> not
be              -> be
a               -> a
word            -> word
, <-              ,
for             -> for
ex              -> ex
ample           -> ampl
the             -> the
Porter          -> porter
algorithm       -> algorithm
reduces         -> reduc
, <-              ,
argue           -> argu
, <-              ,
argued          -> argu
, <-              ,
argues          -> argu
, <-              ,
arg             -> arg
uing            -> u
, <-              ,
and             -> and
argus           -> argu
to              -> to
the             -> the
stem            -> stem
argu            -> argu
. <-              .
```

:[ ] In

:[4] In

```
from nltk.tokenize import sent_tokenize, word_tokenize
sentence = "A stemmer for English operating on the stem cat should identify such strings as
tokenized_words = word_tokenize(sentence)
tokenized_sentence = []
for word in tokenized_words:
    tokenized_sentence.append(porter.stem(word))
tokenized_sentence = " ".join(tokenized_sentence)
tokenized_sentence
```

Out[4]:

```
'A stemmer for english oper on the stem cat should identifi such string as c'
at , catlik , and catty.a stem algorithm might also reduc the word fish , fi
'sh , and fisher to the stem fish
```

In [5]:

```
from nltk.stem.isri import ISRIStemmer
st = ISRIStemmer()
w='حركات'
print(st.stem(w))
```

```
حرك
```

In [6]:

```
file=open("D:\\Users\\D7me_\\Anaconda3\\Abdulrhman.txt")
Sentences= file.read()
def stemSentence(sentence):
 token_words=word_tokenize(sentence)
 token_words
 stem_sentence=[]
 for word in token_words: stem_sentence.append(porter.stem(word))
 stem_sentence.append(" ")
 return "".join(stem_sentence)
print(Sentences)
print("Stemmed sentence")
x=stemSentence(Sentences)
print(x)
```

```
In computer science, artificial intelligence (AI), sometimes called machi
 ne intelligence, is intelligence demonstrated by machines, incontrast to
the natural intelligence displayed by humans and animals. Computer scienc
 e defines AI research as the study of intelligent agents: any device that
perceives its environment and takes actions that maximize its chance of s
uccessfullyachieving its goals
Stemmed sentence
Incomputscienc,artificiintellig(AI),sometimcallmachinintellig,isintelligd
emonstrbymachin,incontrasttothenaturintelligdisplaybyhumanandanim.computs
ciencdefinAIresearchasthestudiofintelligagent:anidevicthatperceivitenviro
 nandtakeactionthatmaximitchancofsuccessfullyachievitgoal
```

In [7]:

```
nltk.download('wordnet')
```

```
nltk_data] Downloading package wordnet to]
...nltk_data]     C:\Users\D7me_\AppData\Roaming\nltk_data]
!nltk_data]   Package wordnet is already up-to-date]
```

Out[7]:

```
True
```

In [ ]:

:[8] In

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
sentence1 = "He was running and eating at same time. He has bad habit of swimming after pla

punctuations="?:!.,;"
sentence_words = nltk.word_tokenize(sentence)
for word in sentence_words:
    if word in punctuations:
        sentence_words.remove(word)
sentence_words
print("{0:20}{1:20}".format("Word","Lemma"))
for word in sentence_words:
    print ("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word)))
```

```
Word                 Lemma
A                    A
stemmer              stemmer
for                  for
English              English
operating            operating
on                   on
the                  the
stem                 stem
cat                  cat
should               should
identify             identify
such                 such
strings              string
as                   a
cats                 cat
catlike              catlike
and                  and
catty.A              catty.A
stemming             stemming
algorithm            algorithm
might                might
also                 also
reduce               reduce
the                  the
words                word
fishing              fishing
fished               fished
and                  and
fisher               fisher
to                   to
the                  the
stem                 stem
fish                 fish
```

:[9] In

```
for word in sentence_words:
    print ("{0:20}{1:20}".format(word,wordnet_lemmatizer.lemmatize(word, pos="v")))
```

```
                   A                   A
             stemmer             stemmer
                 for                 for
             English             English
           operating             operate
                  on                  on
                 the                 the
                stem                stem
                 cat                 cat
              should              should
            identify            identify
                such                such
             strings              string
                  as                  as
                cats                 cat
            catlike             catlike
                and                 and
           catty.A             catty.A
            stemming                stem
           algorithm           algorithm
               might               might
                also                also
              reduce              reduce
                 the                 the
               words               word
             fishing                fish
              fished                fish
                 and                 and
              fisher              fisher
                  to                  to
                 the                 the
                stem                stem
                fish                fish
```

:[11] In

```
file=open(r'D:\Users\D7me_\Anaconda3\Abdulrhman.txt',encoding = 'UTF-8')
from nltk.stem.isri import ISRIStemmer
st = ISRIStemmer()
Sentences= file.read()
def stemSentence(sentence):
    token_words=word_tokenize(sentence)
    token_words
    stem_sentence=[]
    for word in token_words:
        stem_sentence.append(st.stem(word))
        stem_sentence.append(" ")
    return "".join(stem_sentence)

print(Sentences)
print("Stemmed sentence")
x = stemSentence(Sentences)
print(x)
```

```
B777-300ER
Stemmed sentence
 B777-300ER
```

:[ ] In

```

```

:[ ] In

```

```