Text processing

Open a .txt file

In [95]:
```
f = open("txt/snippet-2.txt", "r")
```

In [88]:
```
f
```

Out[88]:
```
<_io.TextIOWrapper name='txt/snippet-2.txt'
mode='r' encoding='UTF-8'>
```

In [89]:
```
s = f.read()
```

In [90]:
```
s
```

Out[90]:
```
'What this library is about?\nWhat do you
want to create?\nWhy does gardening matter to
you?\nWhat elements are part of your dream
garden?\nWhat elements are not part of your
dream garden?\nWhat is the first books,
magazines, or publications that immediately
comes to mind when you think of library
produce?\nAre you going to have help from
library members and other people?\nThink
about your specific growing and garden zone.
Do you have a long growing season or a short
one?\nDo you want "crops" you can store for
some months?\nDo you like caring for people
(readers, volunteers, team members)?\nDo you
have a container garden with seeds?\nBeginner
set up recommendations for seeds?\nHow do you
know what to put together? Do you organize by
author, genre, cover color?\nConditions -
sunlight, soil, moisture?\nWhat do we like to
read?\nIs it accessible?'
```

In [91]:
```
type(s)
```

Out[91]:
```
str
```

In [96]:
```
l = f.readlines()
```

In [97]:
```
l
```

Out[97]:
```
['What this library is about?\n',
 'What do you want to create?\n',
 'Why does gardening matter to you?\n',
 'What elements are part of your dream garden
?\n',
 'What elements are not part of your dream
garden?\n',
 'What is the first books, magazines, or
publications that immediately comes to mind
when you think of library produce?\n',
 'Are you going to have help from library
members and other people?\n',
 'Think about your specific growing and
garden zone. Do you have a long growing
season or a short one?\n',
 'Do you want "crops" you can store for some
months?\n',
 'Do you like caring for people (readers,
volunteers, team members)?\n',
 'Do you have a container garden with seeds?
\n',
 'Beginner set up recommendations for seeds?
\n',
 'How do you know what to put together? Do
you organize by author, genre, cover color?
\n',
 'Conditions - sunlight, soil, moisture?\n',
 'What do we like to read?\n',
 'Is it accessible?']
```

In [98]:
```
type(l)
```

Out[98]:
```
list
```

In [ ]:

Opening a folder with .txt files

In [100]:
```
import os

texts = []

folder = "./txt/"

for filename in os.listdir(folder):
    if ".txt" in filename:
        current_file_path = folder + filename
        print(current_file_path)
        print("---")

        txt = open(current_file_path).read()
        print(txt)
        print("===")

        texts.append(txt)
```

```
./txt/snippet-2.txt
---
What this library is about?
What do you want to create?
Why does gardening matter to you?
What elements are part of your dream garden?
What elements are not part of your dream garden?
What is the first books, magazines, or publications
that immediately comes to mind when you think of
library produce?
Are you going to have help from library members and
other people?
Think about your specific growing and garden zone. Do
you have a long growing season or a short one?
Do you want "crops" you can store for some months?
Do you like caring for people (readers, volunteers,
team members)?
Do you have a container garden with seeds?
Beginner set up recommendations for seeds?
How do you know what to put together? Do you organize
by author, genre, cover color?
Conditions - sunlight, soil, moisture?
What do we like to read?
Is it accessible?
===
./txt/snippet-1.txt
---
When we ask "HOW" we want to know more about methods,
systems, ways to do something. This question is about
the process (or steps) that can lead us to achieving
a certain output. So the answer of this SI's question
might be hidden in the process of gardening. Seeing
"library" and "garden" as intertwined actions, and
gardening as the way we can library something, to
answer the HOW question.
===
```

In [101]:
```
print(texts)
```

```
['What this library is about?\nWhat do you want to
create?\nWhy does gardening matter to you?\nWhat
elements are part of your dream garden?\nWhat
elements are not part of your dream garden?\nWhat is
the first books, magazines, or publications that
immediately comes to mind when you think of library
produce?\nAre you going to have help from library
members and other people?\nThink about your specific
growing and garden zone. Do you have a long growing
season or a short one?\nDo you want "crops" you can
store for some months?\nDo you like caring for people
(readers, volunteers, team members)?\nDo you have a
container garden with seeds?\nBeginner set up
recommendations for seeds?\nHow do you know what to
put together? Do you organize by author, genre, cover
color?\nConditions - sunlight, soil, moisture?\nWhat
do we like to read?\nIs it accessible?', 'When we ask
"HOW" we want to know more about methods, systems,
ways to do something. This question is about the
process (or steps) that can lead us to achieving a
certain output. So the answer of this SI\'s question
might be hidden in the process of gardening. Seeing
"library" and "garden" as intertwined actions, and
gardening as the way we can library something, to
answer the HOW question.']
```

In [ ]:

From text to words

In [104]:
```
text = texts[1]
```

In [105]:
```
print(text)
```

When we ask "HOW" we want to know more about methods, systems, ways to do something. This question is about the process (or steps) that can lead us to achieving a certain output. So the answer of this SI's question might be hidden in the process of gardening. Seeing "library" and "garden" as intertwined actions, and gardening as the way we can library something, to answer the HOW question.

In [106]:
```
words = text.split()
```

In [107]:
```
print(words)
```

```
['When', 'we', 'ask', '"HOW"', 'we', 'want', 'to',
'know', 'more', 'about', 'methods,', 'systems,',
'ways', 'to', 'do', 'something.', 'This', 'question',
'is', 'about', 'the', 'process', '(or', 'steps)',
'that', 'can', 'lead', 'us', 'to', 'achieving', 'a',
'certain', 'output.', 'So', 'the', 'answer', 'of',
'this', "SI's", 'question', 'might', 'be', 'hidden',
'in', 'the', 'process', 'of', 'gardening.', 'Seeing',
'"library"', 'and', '"garden"', 'as', 'intertwined',
'actions,', 'and', 'gardening', 'as', 'the', 'way',
'we', 'can', 'library', 'something,', 'to', 'answer',
'the', 'HOW', 'question.']
```

In [108]:
```
for word in words:
    print(word)
```

When
we
ask
"HOW"
we
want
to
know
more
about
methods,
systems,
ways
to
do
something.
This
question
is
about
the
process
(or
steps)
that
can
lead
us
to
achieving
a
certain
output.
So
the
answer
of
this
SI's
question
might
be
....

Using NLTK

word_tokenize

https://www.nltk.org/api/nltk.tokenize.html

In [26]:
```
from nltk.tokenize import word_tokenize
```

In [27]:
```
words = word_tokenize(texts[0])
```

```
---------------------------------------------------------------
LookupError                                 Traceback (most
Cell In [27], line 1
----> 1 word_tokenize(texts[0])

File ~/.local/lib/python3.9/site-packages/nltk/tokenize/_
in word_tokenize(text, language, preserve_line)
    114 def word_tokenize(text, language="english", prese
    115     """
    116     Return a tokenized copy of *text*,
    117     using NLTK's recommended word tokenizer
   (...)
    127     :type preserve_line: bool
    128     """
--> 129     sentences = [text] if preserve_line else sent
nguage)
    130     return [
    131         token for sent in sentences for token in
_treebank_word_tokenizer.tokenize(sent)
    132     ]

File ~/.local/lib/python3.9/site-packages/nltk/tokenize/_
in sent_tokenize(text, language)
     96 def sent_tokenize(text, language="english"):
     97     """
     98     Return a sentence-tokenized copy of *text*,
     99     using NLTK's recommended sentence tokenizer
   (...)
    104     :param language: the model name in the Punkt
    105     """
--> 106     tokenizer = load(f"tokenizers/punkt/{language
    107     return tokenizer.tokenize(text)

File ~/.local/lib/python3.9/site-packages/nltk/data.py:75
ce_url, format, cache, verbose, logic_parser, fstruct_rea
    747     print(f"<<Loading {resource_url}>>")
    749 # Load the resource.
--> 750 opened_resource = _open(resource_url)
    752 if format == "raw":
    753     resource_val = opened_resource.read()

File ~/.local/lib/python3.9/site-packages/nltk/data.py:87
```

In [28]:
```
nltk.download("punkt")
```

```
[nltk_data] Downloading package punkt to /home/
manetta/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

Out[28]: True

In [31]:
```
words = word_tokenize(texts[0])
```

In [32]:
```
for word in words:
    print(word)
```

When
we
ask
``
HOW
''
we
want
to
know
more
about
methods
,
systems
,
ways
to
do
something
.
This
question
is
about
the
process
(
or
steps
)
that
can
lead
us
to
achieving
a
certain
output
.
So

The downloaded NLTK data is saved in your home folder.

If you want to look into it, you can just open the folder:

In [44]:
```
! ls ~/nltk_data/
```

```
corpora  taggers  tokenizers
```

POS (part-of-speech) tagger

https://www.nltk.org/api/nltk.tag.html

In [33]:
```
from nltk import pos_tag, word_tokenize
```

In [34]:
```
text = texts[0]
```

In [35]:
```
words = word_tokenize(text)
```

In [36]:
```
tags = pos_tag(words)
```

```
---------------------------------------------------------------
LookupError                                 Traceback (most
Cell In [36], line 1
----> 1 pos_tag(words)

File ~/.local/lib/python3.9/site-packages/nltk/tag/__init
_tag(tokens, tagset, lang)
    140 def pos_tag(tokens, tagset=None, lang="eng"):
    141     """
    142     Use NLTK's currently recommended part of spee
    143     tag the given list of tokens.
   (...)
    163     :rtype: list(tuple(str, str))
    164     """
--> 165     tagger = _get_tagger(lang)
    166     return _pos_tag(tokens, tagset, tagger, lang)

File ~/.local/lib/python3.9/site-packages/nltk/tag/__init
t_tagger(lang)
    105     tagger.load(ap_russian_model_loc)
    106 else:
--> 107     tagger = PerceptronTagger()
    108 return tagger

File ~/.local/lib/python3.9/site-packages/nltk/tag/percep
rceptronTagger.__init__(self, load)
    164 self.classes = set()
    165 if load:
    166     AP_MODEL_LOC = "file:" + str(
--> 167         find("taggers/averaged_perceptron_tagger/
    168     )
    169     self.load(AP_MODEL_LOC)

File ~/.local/lib/python3.9/site-packages/nltk/data.py:58
ce_name, paths)
    581 sep = "*" * 70
    582 resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 583 raise LookupError(resource_not_found)

LookupError:
**********************************************************
  Resource averaged_perceptron_tagger not found.
```

In [37]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package
averaged_perceptron_tagger to
[nltk_data]     /home/manetta/nltk_data...
[nltk_data]   Unzipping taggers/
averaged_perceptron_tagger.zip.
```

Out[37]:
```
True
```

In [38]:
```
tags = pos_tag(words)
```

In [39]:
```
print(tags)
```

```
[('When', 'WRB'), ('we', 'PRP'), ('ask', 'VBP'),
(``, ``), ('HOW', 'NNP'), ("''", "''"), ('we',
'PRP'), ('want', 'VBP'), ('to', 'TO'), ('know',
'VB'), ('more', 'JJR'), ('about', 'IN'), ('methods',
'NNS'), (',', ','), ('systems', 'NNS'), (',', ','),
('ways', 'NNS'), ('to', 'TO'), ('do', 'VB'),
('something', 'NN'), ('.', '.'), ('This', 'DT'),
('question', 'NN'), ('is', 'VBZ'), ('about', 'IN'),
('the', 'DT'), ('process', 'NN'), ('(', '('), ('or',
'CC'), ('steps', 'NNS'), (')', ')'), ('that', 'WDT'),
('can', 'MD'), ('lead', 'VB'), ('us', 'PRP'), ('to',
'TO'), ('achieving', 'VBG'), ('a', 'DT'), ('certain',
'JJ'), ('output', 'NN'), ('.', '.'), ('So', 'IN'),
('the', 'DT'), ('answer', 'NN'), ('of', 'IN'),
('this', 'DT'), ('SI', 'NNP'), ("'s", 'POS'),
('question', 'NN'), ('might', 'MD'), ('be', 'VB'),
('hidden', 'VBN'), ('in', 'IN'), ('the', 'DT'),
('process', 'NN'), ('of', 'IN'), ('gardening', 'NN'),
('.', '.'), ('Seeing', 'VBG'), (``, ``),
('library', 'JJ'), ("''", "''"), ('and', 'CC'),
(``, ``), ('garden', 'NN'), ("''", "''"), ('as',
'IN'), ('intertwined', 'JJ'), ('actions', 'NNS'),
(',', ','), ('and', 'CC'), ('gardening', 'NN'),
('as', 'IN'), ('the', 'DT'), ('way', 'NN'), ('we',
'PRP'), ('can', 'MD'), ('library', 'VB'),
('something', 'NN'), (',', ','), ('to', 'TO'),
('answer', 'VB'), ('the', 'DT'), ('HOW', 'NNP'),
('question', 'NN'), ('.', '.')]
```

> An off-the-shelf tagger is available for English. It uses the Penn Treebank tagset.

https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

The output of the POS tagger is a list of *tuples*.

A tuple is one of the Python data objects (like the *list* and *string* we saw last time).

A tuple is always a 2 value object, separated with a comma and wrapped in parantheses: `(value, value)`

You can loop through a list of tuples in this way:

In [111]:
```
for word, tag in tags:
    print(word)
    print(tag)
    print("---")
```

When
WRB
---
we
PRP
---
ask
VBP

``
``

HOW
NNP

''
''
---
we
PRP
---
want
VBP
---
to
TO
---
know
VB
---
more
JJR
---
about
IN
---
methods
NNS

,
,

Now you have access to some of the grammar information of sentences.

We can, for example, store all the verbs in a list.

In [112]:
```
verbs = []

        for word, tag in tags:
            if "VB" in tag:
                print(word)
                verbs.append(word)
```

```
ask
want
know
do
is
lead
achieving
be
hidden
Seeing
library
answer
```

In [113]:
```
print(verbs)
```

```
['ask', 'want', 'know', 'do', 'is', 'lead',
'achieving', 'be', 'hidden', 'Seeing', 'library',
'answer']
```

stopwords

In [24]:
```
import nltk
```

In [40]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/manetta/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
Out[40]: True
```

```
In [48]: ! ls ~/nltk_data/
```

```
 corpora  taggers  tokenizers
```

```
In [49]: ! ls ~/nltk_data/corpora/
```

```
 stopwords  stopwords.zip
```

```
In [52]: ! ls ~/nltk_data/corpora/stopwords/
```

```
 arabic       chinese  french    hungarian
 norwegian       slovene
 azerbaijani  danish   german    indonesian
 portuguese      spanish
 basque       dutch    greek     italian
 README  swedish
 bengali      english  hebrew    kazakh
 romanian        tajik
 catalan      finnish  hinglish  nepali
 russian turkish
```

```
In [53]: ! cat ~/nltk_data/corpora/stopwords/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

In …
```
stopwords = open("/home/manetta/nltk_data/corpora/
stopwords/english", "r").readlines()
```

In [57]:
```
print(stopwords)
```

```
['i\n', 'me\n', 'my\n', 'myself\n', 'we\n', 'our\n',
'ours\n', 'ourselves\n', 'you\n', "you're\n",
"you've\n", "you'll\n", "you'd\n", 'your\n',
'yours\n', 'yourself\n', 'yourselves\n', 'he\n',
'him\n', 'his\n', 'himself\n', 'she\n', "she's\n",
'her\n', 'hers\n', 'herself\n', 'it\n', "it's\n",
'its\n', 'itself\n', 'they\n', 'them\n', 'their\n',
'theirs\n', 'themselves\n', 'what\n', 'which\n',
'who\n', 'whom\n', 'this\n', 'that\n', "that'll\n",
'these\n', 'those\n', 'am\n', 'is\n', 'are\n',
'was\n', 'were\n', 'be\n', 'been\n', 'being\n',
'have\n', 'has\n', 'had\n', 'having\n', 'do\n',
'does\n', 'did\n', 'doing\n', 'a\n', 'an\n', 'the\n',
'and\n', 'but\n', 'if\n', 'or\n', 'because\n',
'as\n', 'until\n', 'while\n', 'of\n', 'at\n', 'by\n',
'for\n', 'with\n', 'about\n', 'against\n',
'between\n', 'into\n', 'through\n', 'during\n',
'before\n', 'after\n', 'above\n', 'below\n', 'to\n',
'from\n', 'up\n', 'down\n', 'in\n', 'out\n', 'on\n',
'off\n', 'over\n', 'under\n', 'again\n', 'further\n',
'then\n', 'once\n', 'here\n', 'there\n', 'when\n',
'where\n', 'why\n', 'how\n', 'all\n', 'any\n',
'both\n', 'each\n', 'few\n', 'more\n', 'most\n',
'other\n', 'some\n', 'such\n', 'no\n', 'nor\n',
'not\n', 'only\n', 'own\n', 'same\n', 'so\n',
'than\n', 'too\n', 'very\n', 's\n', 't\n', 'can\n',
'will\n', 'just\n', 'don\n', "don't\n", 'should\n',
"should've\n", 'now\n', 'd\n', 'll\n', 'm\n', 'o\n',
're\n', 've\n', 'y\n', 'ain\n', 'aren\n', "aren't\n",
'couldn\n', "couldn't\n", 'didn\n', "didn't\n",
'doesn\n', "doesn't\n", 'hadn\n', "hadn't\n",
'hasn\n', "hasn't\n", 'haven\n', "haven't\n",
'isn\n', "isn't\n", 'ma\n', 'mightn\n', "mightn't\n",
'mustn\n', "mustn't\n", 'needn\n', "needn't\n",
'shan\n', "shan't\n", 'shouldn\n', "shouldn't\n",
'wasn\n', "wasn't\n", 'weren\n', "weren't\n",
'won\n', "won't\n", 'wouldn\n', "wouldn't\n"]
```

In […
```
stopwords = open("/home/manetta/nltk_data/corpora/
stopwords/english", "r").read()
stopwords = stopwords.split("\n")
```

In [64]:
```
print(stopwords)
```

```
['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 [ ]: