```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
import nltk
>>> br_tw = nltk.corpus.brown.tagged_words(categories='mystery')
>>> len(br_tw)
57169
>>> br_tw[:5]
[('There', 'EX'), ('were', 'BED'), ('thirty-eight', 'CD'), ('patients', 'NNS'), ('on', 'IN')]

# ---------------------------- Lowercasing tokens

>>> br_tw_lower = [(w.lower(),t) for (w,t) in br_tw]
>>> br_tw_lower[:5]
[('there', 'EX'), ('were', 'BED'), ('thirty-eight', 'CD'), ('patients', 'NNS'), ('on', 'IN')]

# ---------------------------- POS frequencies of word types

>>> br_cfd = nltk.ConditionalFreqDist(br_tw_lower)
>>> br_cfd['so']
FreqDist({'RB': 48, 'QL': 44, 'CS': 34})
>>> br_cfd['question']
FreqDist({'NN': 5, 'VB': 3})
>>> br_cfd['question'].freq('NN')
0.625
>>> br_cfd['question'].freq('VB')
0.375
>>> br_cfd['like']
FreqDist({'CS': 103, 'VB': 19, 'IN': 14, 'JJ': 3})
>>> nltk.help.brown_tagset('CS')
CS: conjunction, subordinating
    that as after whether before while like because if since for than altho
    until so unless though providing once lest s'posin' till whereas
    whereupon supposing tho' albeit then so's 'fore
>>> nltk.help.brown_tagset('IN')
IN: preposition
    of in for by considering to on among at through with under into
    regarding than since despite according per before toward against as
    after during including between without except upon out over ...
>>> nltk.help.brown_tagset('VB')
VB: verb, base: uninflected present, imperative or infinitive
    investigate find act follow inure achieve reduce take remedy re-set
    distribute realize disable feel receive continue place protect
    eliminate elaborate work permit run enter force ...
>>> nltk.help.brown_tagset('V.*')
VB: verb, base: uninflected present, imperative or infinitive
    investigate find act follow inure achieve reduce take remedy re-set
    distribute realize disable feel receive continue place protect
    eliminate elaborate work permit run enter force ...
VB+AT: verb, base: uninflected present or infinitive + article
    wanna
VB+IN: verb, base: uninflected present, imperative or infinitive + preposition
    lookit
VB+JJ: verb, base: uninflected present, imperative or infinitive + adjective
    die-dead
VB+PPO: verb, uninflected present tense + pronoun, personal, accusative
    let's lemme gimme
VB+RP: verb, imperative + adverbial particle
    g'ahn c'mon
VB+TO: verb, base: uninflected present, imperative or infinitive + infinitival to
    wanta wanna
VB+VB: verb, base: uninflected present, imperative or infinitive; hypenated pair
    say-speak
VBD: verb, past tense
    said produced took recommended commented urged found added praised
```

```
    charged listed became announced brought attended wanted voted defeated
    received got stood shot scheduled feared promised made ...
VBG: verb, present participle or gerund
    modernizing improving purchasing Purchasing lacking enabling pricing
    keeping getting picking entering voting warning making strengthening
    setting neighboring attending participating moving ...
VBG+TO: verb, present participle + infinitival to
    gonna
VBN: verb, past participle
    conducted charged won received studied revised operated accepted
    combined experienced recommended effected granted seen protected
    adopted retarded notarized selected composed gotten printed ...
VBN+TO: verb, past participle + infinitival to
    gotta
VBZ: verb, present tense, 3rd person singular
    deserves believes receives takes goes expires says opposes starts
    permits expects thinks faces votes teaches holds calls fears spends
    collects backs eliminates sets flies gives seeks reads ...

# ----------------------------- building trigrams

>>> br_tw_3grams = list(nltk.ngrams(br_tw_lower, 3))
>>> br_tw_3grams[0]
(('there', 'EX'), ('were', 'BED'), ('thirty-eight', 'CD'))
>>> br_tw_3grams[1]
(('were', 'BED'), ('thirty-eight', 'CD'), ('patients', 'NNS'))
>>> br_tw_3grams[2]
(('thirty-eight', 'CD'), ('patients', 'NNS'), ('on', 'IN'))
>>> br_tw_3grams[-1]
(('me', 'PPO'), ("''", "''"), ('.', '.'))

# -------------------------- conditional frequency of preceding POS

>>> br_pre = [(w2+"/"+t2, t1) for ((w1,t1), (w2,t2), (w3,t3)) in br_tw_3grams]
>>> br_pre[0]
('were/BED', 'EX')
>>> br_pre[:5]
[('were/BED', 'EX'), ('thirty-eight/CD', 'BED'), ('patients/NNS', 'CD'), ('on/IN', 'NNS'), ('the/AT',
'IN')]
>>> br_pre_cfd = nltk.ConditionalFreqDist(br_pre)
>>> br_pre_cfd['so/CS']
FreqDist({'.': 7, ',': 6, 'NN': 5, 'NNS': 3, 'QLP': 2, 'PPO': 2, 'VBD': 2, '--': 2, 'RB': 1, 'CC': 1,
...})
>>> br_pre_cfd['so/QL']
FreqDist({'IN': 5, 'BEDZ': 5, ',': 4, 'RB': 3, 'NN': 3, 'QL': 3, 'VBN': 2, 'VB': 2, 'PPO': 2, 'BEN':
1, ...})
>>> br_pre_cfd['so/QL'].most_common(5)
[('IN', 5), ('BEDZ', 5), (',', 4), ('RB', 3), ('NN', 3)]
>>> nltk.help.brown_tagset('BEDZ')
BEDZ: verb 'to be', past tense, 1st and 3rd person singular
    was
>>> br_pre_cfd['question/VB']
FreqDist({'TO': 2, 'DO*': 1})
>>> nltk.help.brown_tagset('DO*')
DO*: verb 'to do', uninflected present tense or imperative, negated
    don't
>>> br_pre_cfd['question/NN']
FreqDist({'JJ': 2, 'CD': 1, 'AT': 1, 'OD': 1})
>>> br_pre_cfd['like/VB']
FreqDist({'PPSS': 8, 'PPSS+MD': 5, 'NNS': 2, 'DOD*': 1, 'DO*': 1, 'TO': 1, 'MD*': 1})
>>> nltk.help.brown_tagset('PPSS')
PPSS: pronoun, personal, nominative, not 3rd person singular
    they we I you ye thou you'uns
>>> br_pre_cfd['like/CS']
FreqDist({'NN': 18, ',': 14, 'RB': 11, 'VB': 8, 'VBD': 7, 'BEDZ': 5, 'NNS': 5, 'VBN': 4, 'RBR': 4,
```

```
'VBG': 4, ...})
>>> br_pre_cfd['like/IN']
FreqDist({'BEDZ*': 2, 'PN': 2, 'BED': 2, 'NN': 2, 'VB': 1, 'NNS': 1, 'VBG': 1, 'RB': 1, 'NP': 1,
'RBR': 1})
>>> nltk.help.brown_tagset('BEDZ*')
BEDZ*: verb 'to be', past tense, 1st and 3rd person singular, negated
    wasn't



# --------------------------- Now onto Penn Treebank

>>> from nltk.corpus import treebank
>>> treebank.words()
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', ...]
>>> treebank.tagged_words()
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ...]
>>> treebank.tagged_words()[:10]
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'),
(',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT')]
>>> treebank.tagged_sents()
[[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'),
(',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT'), ('board', 'NN'), ('as', 'IN'), ('a',
'DT'), ('nonexecutive', 'JJ'), ('director', 'NN'), ('Nov.', 'NNP'), ('29', 'CD'), ('.', '.')],
[('Mr.', 'NNP'), ('Vinken', 'NNP'), ('is', 'VBZ'), ('chairman', 'NN'), ('of', 'IN'), ('Elsevier',
'NNP'), ('N.V.', 'NNP'), (',', ','), ('the', 'DT'), ('Dutch', 'NNP'), ('publishing', 'VBG'),
('group', 'NN'), ('.', '.')], ...]
>>> treebank.tagged_sents()[0]
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'),
(',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT'), ('board', 'NN'), ('as', 'IN'), ('a',
'DT'), ('nonexecutive', 'JJ'), ('director', 'NN'), ('Nov.', 'NNP'), ('29', 'CD'), ('.', '.')]
>>> treebank.tagged_sents()[1]
[('Mr.', 'NNP'), ('Vinken', 'NNP'), ('is', 'VBZ'), ('chairman', 'NN'), ('of', 'IN'), ('Elsevier',
'NNP'), ('N.V.', 'NNP'), (',', ','), ('the', 'DT'), ('Dutch', 'NNP'), ('publishing', 'VBG'),
('group', 'NN'), ('.', '.')]
>>> treebank.tagged_sents()[-1]
[('Trinity', 'NNP'), ('said', 'VBD'), ('0', '-NONE-'), ('it', 'PRP'), ('plans', 'VBZ'), ('*-1', '-
NONE-'), ('to', 'TO'), ('begin', 'VB'), ('delivery', 'NN'), ('in', 'IN'), ('the', 'DT'), ('first',
'JJ'), ('quarter', 'NN'), ('of', 'IN'), ('next', 'JJ'), ('year', 'NN'), ('.', '.')]
>>> treebank.parsed_sents()[0]
Tree('S', [Tree('NP-SBJ', [Tree('NP', [Tree('NNP', ['Pierre']), Tree('NNP', ['Vinken'])]), Tree(',',
[',']), Tree('ADJP', [Tree('NP', [Tree('CD', ['61']), Tree('NNS', ['years'])]), Tree('JJ',
['old'])]), Tree(',', [','])]), Tree('VP', [Tree('MD', ['will']), Tree('VP', [Tree('VB', ['join']),
Tree('NP', [Tree('DT', ['the']), Tree('NN', ['board'])]), Tree('PP-CLR', [Tree('IN', ['as']),
Tree('NP', [Tree('DT', ['a']), Tree('JJ', ['nonexecutive']), Tree('NN', ['director'])])]), Tree('NP-
TMP', [Tree('NNP', ['Nov.']), Tree('CD', ['29'])])])]), Tree('.', ['.'])])

# --------------------------- Sytactic trees are RECURSIVE!

>>> treebank.parsed_sents()[0].pprint()
(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
    (ADJP (NP (CD 61) (NNS years)) (JJ old))
    (, ,))
  (VP
    (MD will)
    (VP
      (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director)))
      (NP-TMP (NNP Nov.) (CD 29))))
  (. .))
>>> treebank.parsed_sents()[0].draw()
```

```
# --------------------------- Penn Treebank POS distribution

>>> treebank.tagged_words()[:10]
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'),
(',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT')]
>>> treebank_tw_lower = [(w.lower(),t) for (w,t) in treebank.tagged_words()]
>>> treebank.tagged_words()[:10]
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'),
(',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT')]
>>> treebank_tw_lower[:10]
[('pierre', 'NNP'), ('vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old', 'JJ'),
(',', ','), ('will', 'MD'), ('join', 'VB'), ('the', 'DT')]

>>> treebank_cfd = nltk.ConditionalFreqDist(treebank_tw_lower)
>>> treebank_cfd['question']
FreqDist({'NN': 12, 'VBP': 1, 'VB': 1})
>>> treebank_cfd['so']
FreqDist({'RB': 66, 'IN': 27})               # only RB and IN?
>>> nltk.help.upenn_tagset('RB')
RB: adverb
    occasionally unabatingly maddeningly adventurously professedly
    stirringly prominently technologically magisterially predominately
    swiftly fiscally pitilessly ...
>>> nltk.help.upenn_tagset('IN')
IN: preposition or conjunction, subordinating
    astride among uppon whether out inside pro despite on by throughout
    below within for towards near behind atop around if like until below
    next into if beside ...

# --------------------------- Treebank treats CS usage of 'so' as part of its preposition (IN) use
                               and its QL (qualifier) use as RB (adverb) broadly

>>> treebank_cfd['share']
FreqDist({'NN': 117, 'VB': 3})

# --------------------------- 'share' is overwhelmingly a noun?? Digging deeper...

>>> dir(treebank)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__',
'__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_citation',
'_comment_char', '_detect_blocks', '_encoding', '_fileids', '_get_root', '_license', '_normalize',
'_parse', '_read_block', '_read_parsed_sent_block', '_read_sent_block', '_read_tagged_sent_block',
'_read_tagged_word_block', '_read_word_block', '_readme', '_root', '_tag', '_tagset', '_unload',
'_word', 'abspath', 'abspaths', 'citation', 'encoding', 'ensure_loaded', 'fileids', 'license',
'open', 'parsed_sents', 'raw', 'readme', 'root', 'sents', 'tagged_sents', 'tagged_words', 'words']

>>> treebank.readme()[:500]
'[ PENN TREEBANK SAMPLE ]\r\nhttp://www.cis.upenn.edu/~treebank/home.html\r\n\r\nThis is a ~5%
fragment of Penn Treebank, (C) LDC 1995.  It is made\r\navailable under fair use for the purposes of
illustrating NLTK tools\r\nfor tokenizing, tagging, chunking and parsing.  This data is for\r\nnon-
commercial use only.\r\n\r\nContents: raw, tagged, parsed and combined data from Wall
Street\r\nJournal for 1650 sentences (99 treebank files wsj_0001 .. wsj_0099).\r\nFor details about
each of the four types, please see the o'
>>> print(treebank.readme()[:500])
[ PENN TREEBANK SAMPLE ]
http://www.cis.upenn.edu/~treebank/home.html

This is a ~5% fragment of Penn Treebank, (C) LDC 1995.  It is made
available under fair use for the purposes of illustrating NLTK tools
for tokenizing, tagging, chunking and parsing.  This data is for
non-commercial use only.
```

```
Contents: raw, tagged, parsed and combined data from Wall Street
Journal for 1650 sentences (99 treebank files wsj_0001 .. wsj_0099).
For details about each of the four types, please see the o

# --------------------------- More tag ambiguity

>>> treebank_cfd['will']
FreqDist({'MD': 280, 'NN': 1})
>>> treebank_cfd['fly']
FreqDist({'VB': 1, 'VBP': 1})
>>> treebank_cfd['like']
FreqDist({'IN': 49, 'VB': 8, 'VBP': 4, 'JJ': 1})

# --------------------------- Penn Treebank tagset definitions

>>> nltk.help.upenn_tagset('VB')
VB: verb, base form
    ask assemble assess assign assume atone attention avoid bake balkanize
    bank begin behold believe bend benefit bevel beware bless boil bomb
    boost brace break bring broil brush build ...
>>> nltk.help.upenn_tagset('V.*')
VB: verb, base form
    ask assemble assess assign assume atone attention avoid bake balkanize
    bank begin behold believe bend benefit bevel beware bless boil bomb
    boost brace break bring broil brush build ...
VBD: verb, past tense
    dipped pleaded swiped regummed soaked tidied convened halted registered
    cushioned exacted snubbed strode aimed adopted belied figgered
    speculated wore appreciated contemplated ...
VBG: verb, present participle or gerund
    telegraphing stirring focusing angering judging stalling lactating
    hankerin' alleging veering capping approaching traveling besieging
    encrypting interrupting erasing wincing ...
VBN: verb, past participle
    multihulled dilapidated aerosolized chaired languished panelized used
    experimented flourished imitated reunifed factored condensed sheared
    unsettled primed dubbed desired ...
VBP: verb, present tense, not 3rd person singular
    predominate wrap resort sue twist spill cure lengthen brush terminate
    appear tend stray glisten obtain comprise detest tease attract
    emphasize mold postpone sever return wag ...
VBZ: verb, present tense, 3rd person singular
    bases reconstructs marks mixes displeases seals carps weaves snatches
    slumps stretches authorizes smolders pictures emerges stockpiles
    seduces fizzes uses bolsters slaps speaks pleads ...
```