```
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.

#------------------- Process Emma: commands from Ex4

>>> f = open('ling1330/gutenberg/austen-emma.txt')
>>> etxt = f.read()
>>> f.close()
>>> import nltk
>>> import pickle
>>> etoks = nltk.word_tokenize(etxt)
>>> etoks = nltk.word_tokenize(etxt.lower())
>>> len(etoks)
191781
>>> etoks[-10:]
['answered', 'in', 'the', 'perfect', 'happiness', 'of', 'the', 'union', '.', 'finis']
>>> etypes = sorted(set(etoks))
>>> len(etypes)
7944
>>> efreq = nltk.FreqDist(etoks)
>>> efreq['she']
2336

#------------------- Build bigrams, bigram FD

>>> e2grams = list(nltk.bigrams(etoks))
>>> e2grams[:10]
[('[', 'emma'), ('emma', 'by'), ('by', 'jane'), ('jane', 'austen'), ('austen', '1816'), ('1816',
']'), (']', 'volume'), ('volume', 'i'), ('i', 'chapter'), ('chapter', 'i')]
>>> e2grams[-10:]
[('fully', 'answered'), ('answered', 'in'), ('in', 'the'), ('the', 'perfect'), ('perfect',
'happiness'), ('happiness', 'of'), ('of', 'the'), ('the', 'union'), ('union', '.'), ('.', 'finis')]
>>> e2gramfd = nltk.FreqDist(e2grams)
>>> e2gramfd[('so', 'well')]
31
>>> e2gramfd.freq('so', 'well')
Traceback (most recent call last):
  File "<pyshell#18>", line 1, in <module>
    e2gramfd.freq('so', 'well')
TypeError: FreqDist.freq() takes 2 positional arguments but 3 were given
>>> e2gramfd.freq(('so', 'well'))
0.00016164354990092815

#------------------- Bigram conditional frequency distribution

>>> e2gramcfd = nltk.ConditionalFreqDist(e2grams)
>>> e2gramcfd.most_common(10)
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    e2gramcfd.most_common(10)
AttributeError: 'ConditionalFreqDist' object has no attribute 'most_common'
>>> e2gramcfd['so']
FreqDist({'much': 98, 'very': 86, ',': 34, 'well': 31, 'many': 29, 'long': 27, '.': 21, 'little': 20,
'far': 19, 'i': 18, ...})
>>> e2gramcfd['so'].most_common(10)
[('much', 98), ('very', 86), (',', 34), ('well', 31), ('many', 29), ('long', 27), ('.', 21),
('little', 20), ('far', 19), ('i', 18)]
>>> e2gramcfd['so']['very']
86
>>> e2gramcfd['so'].freq('very')
0.08884297520661157
>>> e2gramcfd['so'].freq('much')
0.10123966942148876
>>> e2gramcfd['never']
```

```
FreqDist({'been': 19, 'had': 19, 'have': 15, 'saw': 12, 'to': 12, 'seen': 11, 'could': 11, 'marry':
9, ',': 9, 'was': 8, ...})
>>> e2gramcfd['never'].most_common(5)
[('been', 19), ('had', 19), ('have', 15), ('saw', 12), ('to', 12)]
>>> e2gramcfd['not'].most_common(5)
[('be', 165), ('have', 102), ('know', 80), ('to', 74), ('think', 58)]
>>> e2gramcfd['do'].most_common(5)
[('not', 235), ('you', 70), (',', 36), ('.', 36), ('?', 20)]
>>> e2gramcfd['does'].most_common(5)
[('not', 64), ('.', 10), (',', 6), ('seem', 4), ('he', 2)]

#------------------- Preceding *two* words as a condition? Must utilize 3grams...

>>> e3grams = list(nltk.ngrams(etoks, 3))
>>> e3grams[:5]
[('[', 'emma', 'by'), ('emma', 'by', 'jane'), ('by', 'jane', 'austen'), ('jane', 'austen', '1816'),
('austen', '1816', ']')]
>>> e3grams[-5:]
[('perfect', 'happiness', 'of'), ('happiness', 'of', 'the'), ('of', 'the', 'union'), ('the', 'union',
'.'), ('union', '.', 'finis')]
>>> e3grams[-10:]
[('were', 'fully', 'answered'), ('fully', 'answered', 'in'), ('answered', 'in', 'the'), ('in', 'the',
'perfect'), ('the', 'perfect', 'happiness'), ('perfect', 'happiness', 'of'), ('happiness', 'of',
'the'), ('of', 'the', 'union'), ('the', 'union', '.'), ('union', '.', 'finis')]
>>> [x for x in e3grams[-10:]]
[('were', 'fully', 'answered'), ('fully', 'answered', 'in'), ('answered', 'in', 'the'), ('in', 'the',
'perfect'), ('the', 'perfect', 'happiness'), ('perfect', 'happiness', 'of'), ('happiness', 'of',
'the'), ('of', 'the', 'union'), ('the', 'union', '.'), ('union', '.', 'finis')]
>>> [(w1,w2,w3) for (w1,w2,w3) in e3grams[-10:]]
[('were', 'fully', 'answered'), ('fully', 'answered', 'in'), ('answered', 'in', 'the'), ('in', 'the',
'perfect'), ('the', 'perfect', 'happiness'), ('perfect', 'happiness', 'of'), ('happiness', 'of',
'the'), ('of', 'the', 'union'), ('the', 'union', '.'), ('union', '.', 'finis')]
>>> [((w1,w2),w3) for (w1,w2,w3) in e3grams[-10:]]
[(('were', 'fully'), 'answered'), (('fully', 'answered'), 'in'), (('answered', 'in'), 'the'), (('in',
'the'), 'perfect'), (('the', 'perfect'), 'happiness'), (('perfect', 'happiness'), 'of'),
(('happiness', 'of'), 'the'), (('of', 'the'), 'union'), (('the', 'union'), '.'), (('union', '.'),
'finis')]
>>> foo = [((w1,w2),w3) for (w1,w2,w3) in e3grams]
>>> foo[:5]
[(('[', 'emma'), 'by'), (('emma', 'by'), 'jane'), (('by', 'jane'), 'austen'), (('jane', 'austen'),
'1816'), (('austen', '1816'), ']')]
>>> foo[1000:1005]
[(('to', 'them'), '.'), (('them', '.'), 'she'), (('.', 'she'), 'had'), (('she', 'had'), 'many'),
(('had', 'many'), 'acquaintance')]
>>> foo[-1]
(('union', '.'), 'finis')


# <-- now foo has correct format: (w1,w2) as condition, w3 as outcome.
# Now to build a trigram conditional frequency distribution...


>>> e3gramcfd = nltk.ConditionalFreqDist(foo)
>>> e3gramcfd[('so', 'very')]
FreqDist({'much': 10, 'obliging': 6, 'kind': 3, 'odd': 3, '.': 3, 'superior': 2, 'great': 2, 'soon':
2, 'well': 2, 'happy': 2, ...})
>>> e3gramcfd[('so', 'very')].most_common(5)
[('much', 10), ('obliging', 6), ('kind', 3), ('odd', 3), ('.', 3)]
>>> e3gramcfd[('have', 'never')].most_common(5)
[('been', 5), ('seen', 4), ('known', 3), ('had', 2), ('any', 1)]
>>> e3gramcfd[('so', 'well')].most_common(5)
[(',', 7), ('as', 5), ('.', 5), ('satisfied', 3), (';', 2)]


# ---------------  How about good-old trigram frequency distribution?
```

```
>>> e3gramfd = nltk.FreqDist(e3grams)
>>> e3gramfd[('have', 'never', 'been')]
5
>>> e3gramfd.freq('have', 'never', 'been')
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    e3gramfd.freq('have', 'never', 'been')
TypeError: freq() takes 2 positional arguments but 4 were given
>>> e3gramfd.freq(('have', 'never', 'been'))       # probability of trigram 'have never been'
2.607167625235297e-05


#------------------ Where am I? Moving into my script folder

>>> import os
>>> os.getcwd()
'C:\\Users\\narae\\Documents'
>>> os.chdir('ling1330')
>>> os.listdir()
['enable1.txt', 'Ex4_emma_enable.py', 'gettysburg_address.txt', 'gift-of-magi.txt',
'gift_shell_explore.pdf', 'gutenberg', 'hello.py', 'nltk_practice.txt', 'process_gift.py',
'process_gift_out.txt', 'words.pkl']


#------------------ Mandatory fun with ENABLE word list

>>> f = open('words.pkl', 'rb')
>>> wlist = pickle.load(f)
>>> f.close()
>>> len(wlist)
172820
>>> wlist[:10]
['aa', 'aah', 'aahed', 'aahing', 'aahs', 'aal', 'aalii', 'aaliis', 'aals', 'aardvark']
>>> wlist[-10:]
['zymology', 'zymosan', 'zymosans', 'zymoses', 'zymosis', 'zymotic', 'zymurgies', 'zymurgy',
'zyzzyva', 'zyzzyvas']
>>> [w for w in wlist if w.startswith('dw')]
['dwarf', 'dwarfed', 'dwarfer', 'dwarfest', 'dwarfing', 'dwarfish', 'dwarfishly', 'dwarfishness',
'dwarfishnesses', 'dwarfism', 'dwarfisms', 'dwarflike', 'dwarfness', 'dwarfnesses', 'dwarfs',
'dwarves', 'dweeb', 'dweebs', 'dwell', 'dwelled', 'dweller', 'dwellers', 'dwelling', 'dwellings',
'dwells', 'dwelt', 'dwindle', 'dwindled', 'dwindles', 'dwindling', 'dwine', 'dwined', 'dwines',
'dwining']
>>> [w for w in wlist if 'a' and 'e' and 'i' and 'o' and 'u' not in w]
KeyboardInterrupt         ############ WRONG syntax!
>>> [w for w in wlist if 'a' not in w and 'e' not in w and 'i' not in w and 'o' not in w and 'u' not
in w]
['brr', 'brrr', 'by', 'byrl', 'byrls', 'bys', 'crwth', 'crwths', 'cry', 'crypt', 'crypts', 'cwm',
'cwms', 'cyst', 'cysts', 'dry', 'dryly', 'drys', 'fly', 'flyby', 'flybys', 'flysch', 'fry', 'ghyll',
'ghylls', 'glycyl', 'glycyls', 'glyph', 'glyphs', 'gym', 'gyms', 'gyp', 'gyps', 'gypsy', 'hm', 'hmm',
'hymn', 'hymns', 'hyp', 'hyps', 'lymph', 'lymphs', 'lynch', 'lynx', 'mm', 'my', 'myrrh', 'myrrhs',
'myth', 'myths', 'mythy', 'nth', 'nymph', 'nymphs', 'pfft', 'phpht', 'pht', 'ply', 'pry', 'psst',
'psych', 'psychs', 'pygmy', 'pyx', 'rhythm', 'rhythms', 'rynd', 'rynds', 'scry', 'sh', 'shh', 'shy',
'shyly', 'sky', 'sly', 'slyly', 'spry', 'spryly', 'spy', 'sty', 'stymy', 'sylph', 'sylphs', 'sylphy',
'syn', 'sync', 'synch', 'synchs', 'syncs', 'synth', 'synths', 'syph', 'syphs', 'syzygy', 'thy',
'thymy', 'try', 'tryst', 'trysts', 'tsk', 'tsks', 'tsktsk', 'tsktsks', 'typp', 'typps', 'typy',
'why', 'whys', 'wry', 'wryly', 'wych', 'wyn', 'wynd', 'wynds', 'wynn', 'wynns', 'wyns', 'xylyl',
'xylyls', 'xyst', 'xysts']
>>> [w for w in wlist if 'a' not in w and 'e' not in w and 'i' not in w and 'o' not in w and 'u' not
in w and 'y' not in w]
['brr', 'brrr', 'crwth', 'crwths', 'cwm', 'cwms', 'hm', 'hmm', 'mm', 'nth', 'pfft', 'phpht', 'pht',
'psst', 'sh', 'shh', 'tsk', 'tsks', 'tsktsk', 'tsktsks']


#------------------------- Palindromes
```

```
>>> def getRev(wd):
...     rev = ''
...     for c in wd:
...         rev = c + rev
...     return rev
...
>>> getRev('penguin')
'niugnep'
>>> getRev('level')
'level'
>>> [w for w in wlist if w == getRev(w)]
['aa', 'aba', 'abba', 'aga', 'aha', 'ala', 'alula', 'ama', 'ana', 'anna', 'ava', 'awa', 'bib', 'bob',
'boob', 'bub', 'civic', 'dad', 'deed', 'deified', 'deked', 'deled', 'denned', 'dewed', 'did', 'dud',
'eke', 'eme', 'ere', 'eve', 'ewe', 'eye', 'gag', 'gig', 'hah', 'halalah', 'hallah', 'heh', 'huh',
'kaiak', 'kayak', 'keek', 'kook', 'level', 'madam', 'marram', 'mem', 'mim', 'minim', 'mm', 'mom',
'mum', 'naan', 'nan', 'noon', 'nun', 'oho', 'otto', 'oxo', 'pap', 'peep', 'pep', 'pip', 'poop',
'pop', 'pullup', 'pup', 'radar', 'redder', 'refer', 'reifier', 'repaper', 'reviver', 'rotator',
'rotor', 'sagas', 'sees', 'selles', 'sememes', 'semes', 'seres', 'sexes', 'shahs', 'sis', 'solos',
'sos', 'stats', 'stets', 'sulus', 'tat', 'tenet', 'terret', 'tet', 'tit', 'toot', 'torot', 'tot',
'tut', 'ulu', 'vav', 'waw', 'wow', 'yay']
>>> [(len(w), w) for w in wlist if w == getRev(w)]
[(2, 'aa'), (3, 'aba'), (4, 'abba'), (3, 'aga'), (3, 'aha'), (3, 'ala'), (5, 'alula'), (3, 'ama'),
(3, 'ana'), (4, 'anna'), (3, 'ava'), (3, 'awa'), (3, 'bib'), (3, 'bob'), (4, 'boob'), (3, 'bub'), (5,
'civic'), (3, 'dad'), (4, 'deed'), (7, 'deified'), (5, 'deked'), (5, 'deled'), (6, 'denned'), (5,
'dewed'), (3, 'did'), (3, 'dud'), (3, 'eke'), (3, 'eme'), (3, 'ere'), (3, 'eve'), (3, 'ewe'), (3,
'eye'), (3, 'gag'), (3, 'gig'), (3, 'hah'), (7, 'halalah'), (6, 'hallah'), (3, 'heh'), (3, 'huh'),
(5, 'kaiak'), (5, 'kayak'), (4, 'keek'), (4, 'kook'), (5, 'level'), (5, 'madam'), (6, 'marram'), (3,
'mem'), (3, 'mim'), (5, 'minim'), (2, 'mm'), (3, 'mom'), (3, 'mum'), (4, 'naan'), (3, 'nan'), (4,
'noon'), (3, 'nun'), (3, 'oho'), (4, 'otto'), (3, 'oxo'), (3, 'pap'), (4, 'peep'), (3, 'pep'), (3,
'pip'), (4, 'poop'), (3, 'pop'), (6, 'pullup'), (3, 'pup'), (5, 'radar'), (6, 'redder'), (5,
'refer'), (7, 'reifier'), (7, 'repaper'), (7, 'reviver'), (7, 'rotator'), (5, 'rotor'), (5, 'sagas'),
(4, 'sees'), (6, 'selles'), (7, 'sememes'), (5, 'semes'), (5, 'seres'), (5, 'sexes'), (5, 'shahs'),
(3, 'sis'), (5, 'solos'), (3, 'sos'), (5, 'stats'), (5, 'stets'), (5, 'sulus'), (3, 'tat'), (5,
'tenet'), (6, 'terret'), (3, 'tet'), (3, 'tit'), (4, 'toot'), (5, 'torot'), (3, 'tot'), (3, 'tut'),
(3, 'ulu'), (3, 'vav'), (3, 'waw'), (3, 'wow'), (3, 'yay')]
>>> sorted([(len(w), w) for w in wlist if w == getRev(w)], reverse=True)
[(7, 'sememes'), (7, 'rotator'), (7, 'reviver'), (7, 'repaper'), (7, 'reifier'), (7, 'halalah'), (7,
'deified'), (6, 'terret'), (6, 'selles'), (6, 'redder'), (6, 'pullup'), (6, 'marram'), (6, 'hallah'),
(6, 'denned'), (5, 'torot'), (5, 'tenet'), (5, 'sulus'), (5, 'stets'), (5, 'stats'), (5, 'solos'),
(5, 'shahs'), (5, 'sexes'), (5, 'seres'), (5, 'semes'), (5, 'sagas'), (5, 'rotor'), (5, 'refer'), (5,
'radar'), (5, 'minim'), (5, 'madam'), (5, 'level'), (5, 'kayak'), (5, 'kaiak'), (5, 'dewed'), (5,
'deled'), (5, 'deked'), (5, 'civic'), (5, 'alula'), (4, 'toot'), (4, 'sees'), (4, 'poop'), (4,
'peep'), (4, 'otto'), (4, 'noon'), (4, 'naan'), (4, 'kook'), (4, 'keek'), (4, 'deed'), (4, 'boob'),
(4, 'anna'), (4, 'abba'), (3, 'yay'), (3, 'wow'), (3, 'waw'), (3, 'vav'), (3, 'ulu'), (3, 'tut'), (3,
'tot'), (3, 'tit'), (3, 'tet'), (3, 'tat'), (3, 'sos'), (3, 'sis'), (3, 'pup'), (3, 'pop'), (3,
'pip'), (3, 'pep'), (3, 'pap'), (3, 'oxo'), (3, 'oho'), (3, 'nun'), (3, 'nan'), (3, 'mum'), (3,
'mom'), (3, 'mim'), (3, 'mem'), (3, 'huh'), (3, 'heh'), (3, 'hah'), (3, 'gig'), (3, 'gag'), (3,
'eye'), (3, 'ewe'), (3, 'eve'), (3, 'ere'), (3, 'eme'), (3, 'eke'), (3, 'dud'), (3, 'did'), (3,
'dad'), (3, 'bub'), (3, 'bob'), (3, 'bib'), (3, 'awa'), (3, 'ava'), (3, 'ana'), (3, 'ama'), (3,
'ala'), (3, 'aha'), (3, 'aga'), (3, 'aba'), (2, 'mm'), (2, 'aa')]


#------------------------- Anagrams

>>> [w for w in wlist if sorted(w) == sorted('stop')]
['opts', 'post', 'pots', 'spot', 'stop', 'tops']
>>> [w for w in wlist if sorted(w) == sorted('street')]
['retest', 'setter', 'street', 'tester']

#------------------------- Wordle words

>>> wordlewords = [w for w in wlist if len(w)==5]
>>> len(wordlewords)
8636
```

```
>>> wordlewords[:20]
['aahed', 'aalii', 'aargh', 'abaca', 'abaci', 'aback', 'abaft', 'abaka', 'abamp', 'abase', 'abash',
'abate', 'abbas', 'abbes', 'abbey', 'abbot', 'abeam', 'abele', 'abets', 'abhor']
>>> wordlewords[-20:]
['zlote', 'zloty', 'zoeae', 'zoeal', 'zoeas', 'zombi', 'zonal', 'zoned', 'zoner', 'zones', 'zonks',
'zooid', 'zooks', 'zooms', 'zoons', 'zooty', 'zoril', 'zoris', 'zowie', 'zymes']
```