Wordnet is an NLTK corpus reader that can look up words using synsets

```python
import nltk
import math
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
from nltk.corpus import sentiwordnet as swn

nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
from nltk.book import *

nltk.download('book')
nltk.download('omw-1.4')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('sentiwordnet')
```

```
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Package timit is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Package toolbox is already up-to-date!
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Package udhr is already up-to-date!
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Package udhr2 is already up-to-date!
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package unicode_samples is already up-to-date!
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Package webtext is already up-to-date!
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /root/nltk_data...
```

```
[nltk_data]    |   Package maxent_treebank_pos_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package universal_tagset is already up-to-date!
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package book_grammars is already up-to-date!
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package city_database is already up-to-date!
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Package tagsets is already up-to-date!
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package panlex_swadesh is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection book
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package sentiwordnet to /root/nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!
```

Output all synsets of 'cat' as well as the definition, examples, and lemmas of one synset. Then traverse the hierarchy as high as you can

```
synsets = wn.synsets('cat')
print("All synsets:", synsets)
print()
synset = synsets[0]

print("Synset:", synset)
print("Definition:", synset.definition())
print("Examples:", synset.examples())
print("Lemmas:", synset.lemmas())
```

```
top = wn.synset('entity.n.01')
while synset:
    print(synset)
    if synset == top:
        break
    if synset.hypernyms():
        synset = synset.hypernyms()[0]
```

```
All synsets: [Synset('cat.n.01'), Synset('guy.n.01'), Synset('cat.n.03'), Synset('kat.n

Synset: Synset('cat.n.01')
Definition: feline mammal usually having thick soft fur and no ability to roar: domestic
Examples: []
Lemmas: [Lemma('cat.n.01.cat'), Lemma('cat.n.01.true_cat')]
Synset('cat.n.01')
Synset('feline.n.01')
Synset('carnivore.n.01')
Synset('placental.n.01')
Synset('mammal.n.01')
Synset('vertebrate.n.01')
Synset('chordate.n.01')
Synset('animal.n.01')
Synset('organism.n.01')
Synset('living_thing.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

Output the hypernyms, hyponyms, meronyms, holonyms, and antonyms of that synset

```
print("Hypernyms:", synset.hypernyms())
print("Hyponyms:", synset.hyponyms())
print("Meronyms:", synset.part_meronyms())
print("Holonyms:", synset.part_holonyms())
print("Antonyms:", synset.lemmas()[0].antonyms())
```

```
Hypernyms: []
Hyponyms: [Synset('abstraction.n.06'), Synset('physical_entity.n.01'), Synset('thing.n.0
Meronyms: []
Holonyms: []
Antonyms: []
```

Select a verb and output synsets, the definition, examples, lemmas, and then traverse the hierarchy as high as you can

```
verbs = wn.synsets('climb')
print("All synsets:", verbs)
print()
synset = verbs[0]

print("Synset:", synset)
print("Definition:", synset.definition())
print("Examples:", synset.examples())
print("Lemmas:", synset.lemmas())

top = wn.synset('entity.n.01')
while synset:
    print(synset)
    if synset == top:
        break
    if synset.hypernyms():
        synset = synset.hypernyms()[0]
```

```
All synsets: [Synset('ascent.n.01'), Synset('climb.n.02'), Synset('climb.n.03'), Synset(

Synset: Synset('ascent.n.01')
Definition: an upward slope or grade (as in a road)
Examples: ["the car couldn't make it up the rise"]
Lemmas: [Lemma('ascent.n.01.ascent'), Lemma('ascent.n.01.acclivity'), Lemma('ascent.n.0
Synset('ascent.n.01')
Synset('slope.n.01')
Synset('geological_formation.n.01')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

Use morphy to find different forms of the word

```
print(wn.morphy('climb'))
print(wn.morphy('climbing'))
print(wn.morphy('climbs'))
print(wn.morphy('climbed'))
```

```
climb
climbing
```

```
climb
climb
```

Select two words that you think might be similar. Find the specific synsets you are interested in. Run the Wu-Palmer similarity metric and the Lesk algorithm.

```
bigsyn = wn.synsets('big')
big = bigsyn[1]
largesyn = wn.synsets('large')
large = largesyn[2]

print(big)
print(big.definition())
print()
print(large)
print(large.definition())
print()

# Wu-Palmer similarity
wn.wup_similarity(big, large)

# Lesk algo
sent = word_tokenize("The empire had grown large")
print(lesk(sent, 'big'))
print(wn.synset('large.a.01').definition())
```

```
Synset('big.s.02')
significant

Synset('large.s.02')
fairly large or important in effect; influential

Synset('large.a.01')
above average in size or number or quantity or magnitude or extent
```

Select an emotionally charged word. Find its senti-synsets and output the polarity scores for each word. Make up a sentence. Output the polarity for each word in the sentence.

```
#sentiwordnet analysis
emote = 'rage'
sentisynsets = list(swn.senti_synsets(emote))
```

```
print('Sentisynsets:')
for synset in sentisynsets:
  print(synset)
  print("Pos:", synset.pos_score(), 'Neg:', synset.neg_score(), 'Obj:', synset.obj_score())
  print()
```

```
Sentisynsets:
<fury.n.01: PosScore=0.25 NegScore=0.5>
Pos: 0.25 Neg: 0.5 Obj: 0.25

<rage.n.02: PosScore=0.0 NegScore=0.125>
Pos: 0.0 Neg: 0.125 Obj: 0.875

<rage.n.03: PosScore=0.625 NegScore=0.0>
Pos: 0.625 Neg: 0.0 Obj: 0.375

<rage.n.04: PosScore=0.0 NegScore=0.125>
Pos: 0.0 Neg: 0.125 Obj: 0.875

<fad.n.01: PosScore=0.25 NegScore=0.0>
Pos: 0.25 Neg: 0.0 Obj: 0.75

<ramp.v.01: PosScore=0.0 NegScore=0.0>
Pos: 0.0 Neg: 0.0 Obj: 1.0

<rage.v.02: PosScore=0.0 NegScore=0.5>
Pos: 0.0 Neg: 0.5 Obj: 0.5

<rage.v.03: PosScore=0.0 NegScore=0.5>
Pos: 0.0 Neg: 0.5 Obj: 0.5
```

```
#word polarity
sent = 'The quick brown fox jumped over the lazy dog.'
tokens = word_tokenize(sent)
for token in tokens:
  token_synsets = list(swn.senti_synsets(token))
  if token_synsets:
    print(token_synsets[0], 'Pos:', token_synsets[0].pos_score(), 'Neg:', token_synsets[0].ne
```

```
<quick.n.01: PosScore=0.0 NegScore=0.0> Pos: 0.0 Neg: 0.0 Obj: 1.0
<brown.n.01: PosScore=0.0 NegScore=0.375> Pos: 0.0 Neg: 0.375 Obj: 0.625
<fox.n.01: PosScore=0.0 NegScore=0.0> Pos: 0.0 Neg: 0.0 Obj: 1.0
<jump.v.01: PosScore=0.0 NegScore=0.0> Pos: 0.0 Neg: 0.0 Obj: 1.0
<over.n.01: PosScore=0.0 NegScore=0.0> Pos: 0.0 Neg: 0.0 Obj: 1.0
<lazy.s.01: PosScore=0.0 NegScore=0.0> Pos: 0.0 Neg: 0.0 Obj: 1.0
<dog.n.01: PosScore=0.0 NegScore=0.0> Pos: 0.0 Neg: 0.0 Obj: 1.0
```

Output collocations for text4, the Inaugural corpus. Select one of the collocations identified by NLTK. Calculate mutual

information

```
colloc = text4.collocations()
print(colloc)

#prob of vice pres
print()
length = len(set(text4))
vicePres = text.count('Vice President') / length
print('Probability of vice president:', vicePres)
justVice = text.count('Vice') / length
print('Probability of vice:', justVice)
justPres = text.count('President') / length
print('Probability of president:', justPres)

print()
pmi = math.log2(vicePres / (justVice * justPres))
print('PMI:', pmi)
```

```
United States; fellow citizens; years ago; four years; Federal
Government; General Government; American people; Vice President; God
bless; Chief Justice; one another; fellow Americans; Old World;
Almighty God; Fellow citizens; Chief Magistrate; every citizen; Indian
tribes; public debt; foreign nations
None

Probability of vice president: 0.0017955112219451373
Probability of vice: 0.0018952618453865336
Probability of president: 0.010773067331670824

PMI: 6.458424602064904
```