#1. WordNet is a hierarchical organization of nouns, verbs, adjectives, and adverbs; listing:

- Glosses: short definitions
- Synsets: synonym sets
- Use examples
- Relations to other words

```
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')
```

#2/#3. Ouput a synset "**Medicine**" and extract its definition, usage examples, and lemmas. Transverse up the Wordnet hierarchy as far as I can.

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

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

i = 1
while synset:
    print(synset)
    if i > 10:
        break
    i += 1
    if synset.hypernyms():
        synset = synset.hypernyms()[0]
```

```
All synsets: [Synset('medicine.n.01'), Synset('medicine.n.02'), Synset('medicine.n.03'), Synset('music.n.05'), Synset('medicate.v.02')]

Synset: Synset('medicine.n.01')
Definition: the branches of medical science that deal with nonsurgical techniques
Examples: []
Lemmas: [Lemma('medicine.n.01.medicine'), Lemma('medicine.n.01.medical_specialty')]
Synset('medicine.n.01')
Synset('medical_science.n.01')
Synset('life_science.n.01')
Synset('natural_science.n.01')
Synset('science.n.01')
Synset('discipline.n.01')
Synset('knowledge_domain.n.01')
Synset('content.n.05')
Synset('cognition.n.01')
```

```
Synset('psychological_feature.n.01')
Synset('abstraction.n.06')
```

#4. Output the following if exist: hypernyms, hyponyms, meronyms, holonyms, antonym

```
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: [Synset('medical_science.n.01')]
Hyponyms: [Synset('allergology.n.01'), Synset('anesthesiology.n.01'), Synset('angiology.n.01'), Synset('bacteriology.n.01'), Synset('bio
Meronyms: []
Holonyms: []
Antonyms: []
```

#5/#6 - Ouput a synset "**Punish**" and extract its definition, usage examples, and lemmas. Transverse up the Wordnet hierarchy as far as I can.

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

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

i = 1
while synset:
    print(synset)
    if i > 8:
        break
    i += 1
    if synset.hypernyms():
        synset = synset.hypernyms()[0]
```

```
All synsets: [Synset('walk.n.01'), Synset('base_on_balls.n.01'), Synset('walk.n.03'), Synset('walk.n.04'), Synset('walk.n.05'), Synset(

Synset: Synset('walk.n.01')
Definition: the act of traveling by foot
Examples: ['walking is a healthy form of exercise']
Lemmas: [Lemma('walk.n.01.walk'), Lemma('walk.n.01.walking')]
Synset('walk.n.01')
Synset('locomotion.n.02')
Synset('motion.n.06')
Synset('change.n.03')
Synset('action.n.01')
Synset('act.n.02')
Synset('event.n.01')
Synset('psychological_feature.n.01')
Synset('abstraction.n.06')
```

#7 - Use morphy to find as many different forms of the word as you can

```
print(wn.morphy('walk', wn.NOUN))
print(wn.morphy('walked'))
print(wn.morphy('walking', wn.ADJ))
```

```
walk
walk
walking
```

#8 - Select 2 words "**many**" and "**numerous**" that you think might be similar. Find the specific synsets you are interested in. Run the Wu-Palmer similarity metric and the Lesk algorithm.

```
hop = wn.synset('hop.v.01')
jump = wn.synset('jump.v.01')

# Wu-Palmer similarity
print("Wu-Palmer: ", wn.wup_similarity(hop, jump))

# Lesk Algorithm
sent = ("I will jump over the rainbow")
print(lesk(sent, 'jump'))
print(lesk(sent, 'jump').definition())
```

```
Wu-Palmer:  0.8
Synset('startle.n.01')
a sudden involuntary movement
```

#9 - Select an emotionally charged word "**anxious**". 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
anxious = 'anxious'
senti_synsets = list(swn.senti_synsets(anxious))
print("sentisynsets: ")
for synset in senti_synsets:
  print(synset)
  print("Positive Score: ", synset.pos_score())
  print("Negative Score: ", synset.neg_score())
  print("Objective Score: ", synset.obj_score())
```

```
sentisynsets:
<anxious.s.01: PosScore=0.125 NegScore=0.0>
Positive Score:  0.125
Negative Score:  0.0
Objective Score:  0.875
<anxious.s.02: PosScore=0.125 NegScore=0.625>
Positive Score:  0.125
Negative Score:  0.625
Objective Score:  0.25
```

```
#word polarity
sent = 'At that moment he was not listening to music, he was living an experience.'
tokens = word_tokenize(sent)
for token in tokens:
  synsets_tok = list(swn.senti_synsets(token))
  if synsets_tok:
    print(synsets_tok[0])
    print('Pos:', synsets_tok[0].pos_score(), "\t")
    print('Neg:', synsets_tok[0].neg_score(), "\t")
    print('Obj:', synsets_tok[0].obj_score(), "\t")
```

```
<astatine.n.01: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
<moment.n.01: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
<helium.n.01: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
<washington.n.02: PosScore=0.0 NegScore=0.0>
```

```
Pos: 0.0
Neg: 0.0
Obj: 1.0
<not.r.01: PosScore=0.0 NegScore=0.625>
Pos: 0.0
Neg: 0.625
Obj: 0.375
<listening.n.01: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
<music.n.01: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
<helium.n.01: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
<washington.n.02: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
<life.n.02: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
<associate_in_nursing.n.01: PosScore=0.0 NegScore=0.125>
Pos: 0.0
Neg: 0.125
Obj: 0.875
<experience.n.01: PosScore=0.0 NegScore=0.0>
Pos: 0.0
Neg: 0.0
Obj: 1.0
```

#10 - 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()

text = ' '.join(text4.tokens)
text[:50]

vocab = len(set(text4))
us = text.count('United States')/vocab
print("Probability of United States:", us)
u = text.count('United')/vocab
print("Probability of United:", u)
s = text.count('States')/vocab
print("Probability of States:", s)
pmi = math.log2(us / (u * s))
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 United States: 0.015860349127182045
Probability of United: 0.0170573566084788
Probability of States: 0.03301745635910224
pmi:  4.815657649820885
```

✓ 0s completed at 12:05 AM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.