```
import spacy

nlp = spacy.load("en_core_web_sm")

text = "Apple is looking at buying a UK startup for $1 billion."

doc = nlp(text)

for ent in doc.ents:
    print(f"Entity: {ent.text}, Label: {ent.label_}, Explanation: {spacy.explain(ent.label_)}")
```

```
Entity: Apple, Label: ORG, Explanation: Companies, agencies, institutions, etc.
Entity: UK, Label: GPE, Explanation: Countries, cities, states
Entity: $1 billion, Label: MONEY, Explanation: Monetary values, including unit
```

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.corpus import wordnet

word = "bank"

synsets = wordnet.synsets(word)

for syn in synsets:
    print(f"Synset: {syn.name()}")
    print(f"Definition: {syn.definition()}")
    print(f"Examples: {syn.examples()}")
    print("-----")
```

```
Synset: bank.n.05
Definition: a supply or stock held in reserve for future use (especially in emergencies)
Examples: []
-----
Synset: bank.n.06
Definition: the funds held by a gambling house or the dealer in some gambling games
Examples: ['he tried to break the bank at Monte Carlo']
-----
Synset: bank.n.07
Definition: a slope in the turn of a road or track; the outside is higher than the inside in order t
```

```
Synset: bank.v.02
Definition: enclose with a bank
Examples: ['bank roads']
-----
Synset: bank.v.03
Definition: do business with a bank or keep an account at a bank
Examples: ['Where do you bank in this town?']
-----
Synset: bank.v.04
Definition: act as the banker in a game or in gambling
Examples: []
-----
Synset: bank.v.05
Definition: be in the banking business
Examples: []
-----
Synset: deposit.v.02
Definition: put into a bank account
Examples: ['She deposits her paycheck every month']
-----
Synset: bank.v.07
Definition: cover with ashes so to control the rate of burning
Examples: ['bank a fire']
-----
Synset: trust.v.01
Definition: have confidence or faith in
Examples: ['We can trust in God', 'Rely on your friends', 'bank on your good education', "I swear by
-----
```

```
import re

def parse_fopc(expression):
    pattern = r"([A-Za-z]+)\(([^,]+),\s?([^,]+)\)"
    match = re.match(pattern, expression)

    if match:
        predicate = match.group(1)
        arg1 = match.group(2)
        arg2 = match.group(3)
        return {"Predicate": predicate, "Argument1": arg1, "Argument2": arg2}
    else:
        return "Invalid FOPC expression."

expression = "Loves(John, Mary)"
result = parse_fopc(expression)
print(result)
```

```
{'Predicate': 'Loves', 'Argument1': 'John', 'Argument2': 'Mary'}
```

```
from nltk.wsd import lesk
from nltk.corpus import wordnet

sentence = "The bank can guarantee deposits will eventually cover future financial needs."
word = "bank"

best_sense = lesk(sentence.split(), word)

if best_sense:
    print(f"Best sense: {best_sense.name()}")
```

```
        print(f"Definition: {best_sense.definition()}")
    else:
        print("No appropriate sense found.")
```

```
Best sense: depository_financial_institution.n.01
Definition: a financial institution that accepts deposits and channels the money into lending activitie
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

documents = [
    "The car is driven on the road",
    "The truck is driven on the highway",
    "The bike is parked on the sidewalk"
]

query = "drive car"

vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(documents + [query])

query_vector = tfidf_matrix[-1]
similarities = (tfidf_matrix[:-1] * query_vector.T).toarray()

ranked_docs = np.argsort(similarities.flatten())[::-1]
for idx in ranked_docs:
    print(f"Document {idx + 1}: {documents[idx]}, Similarity Score: {similarities[idx][0]}")
```

```
Document 1: The car is driven on the road, Similarity Score: 0.22545378329488952
Document 3: The bike is parked on the sidewalk, Similarity Score: 0.0
Document 2: The truck is driven on the highway, Similarity Score: 0.0
```