```
import spacy

# Load the pre-trained NER model
nlp = spacy.load("en_core_web_sm")

# Input text for NER
text = "Apple is looking at buying U.K. startup for $1 billion"

# Apply NER
doc = nlp(text)

# Print detected entities and their labels
for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Apple ORG
U.K. GPE
$1 billion MONEY
```

```
import nltk
from nltk.corpus import wordnet

# Download WordNet corpus if not already downloaded
nltk.download('wordnet')

# Function to explore word meanings using WordNet
def explore_word_meanings(word):
    synsets = wordnet.synsets(word)

    if not synsets:
        print(f"No synsets found for '{word}'.")
    else:
        print(f"Synsets found for '{word}':")
        for synset in synsets:
            print(f"Synset Name: {synset.name()}")
            print(f"POS Tag: {synset.pos()}")
            print(f"Definition: {synset.definition()}")
            print(f"Examples: {synset.examples()}")
            print()

# Example usage
word = "computer"
explore_word_meanings(word)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
Synsets found for 'computer':
Synset Name: computer.n.01
POS Tag: n
Definition: a machine for performing calculations automatically
Examples: []

Synset Name: calculator.n.01
POS Tag: n
```

```
Definition: an expert at calculation (or at operating calculating machines)
Examples: []
```

```
import re

def parse_fopc(expression):
    pattern = r'(\w+)\((\w+),(\w+)\)'
    match = re.match(pattern, expression)

    if match:
        predicate, arg1, arg2 = match.groups()
        return {"Predicate": predicate, "Argument1": arg1, "Argument2": arg2}
    return None

# Example logical expression
expression = "Loves(Darshan,Mary)"
parsed_expression = parse_fopc(expression)
print(parsed_expression)
```

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

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample documents
documents = [
    "The cat sat on the mat.",
    "The dog barked at the cat.",
    "The bird flew over the cat and dog."
]

# Create a TF-IDF Vectorizer
vectorizer = TfidfVectorizer()

# Compute TF-IDF matrix
tfidf_matrix = vectorizer.fit_transform(documents)

# Display TF-IDF matrix
print(tfidf_matrix.toarray())

# Display the terms in the vocabulary
print("Vocabulary:", vectorizer.get_feature_names_out())
```

```
[[0.         0.         0.         0.         0.27116066 0.
  0.         0.4591149  0.4591149  0.         0.4591149  0.54232132]
 [0.         0.48098405 0.48098405 0.         0.28407693 0.36580076
  0.         0.         0.         0.         0.         0.56815385]
 [0.39769885 0.         0.         0.39769885 0.23488735 0.30246022
  0.39769885 0.         0.         0.39769885 0.         0.46977469]]
Vocabulary: ['and' 'at' 'barked' 'bird' 'cat' 'dog' 'flew' 'mat' 'on' 'over' 'sat'
 'the']
```