1. Introduction

1. Project Purpose and Background: To understand how to use what you have learned in Week 7 by creating a program with a specific purpose.

2. Goal: Create a search_engine that finds similar sentences and implement the functionality of the search_engine using functions.

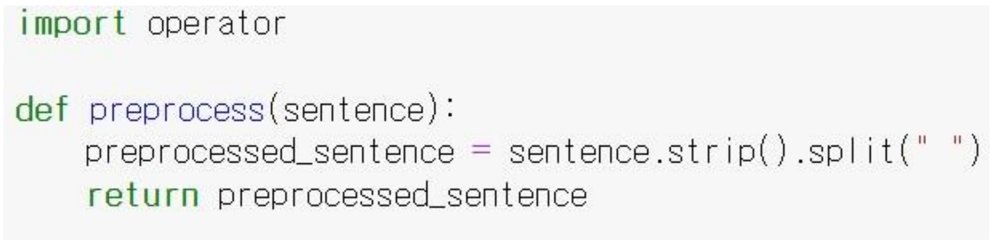
2. Requirements

1. User requirement: For a given sentence, a program that lists 10 similar sentences in the data along with their similarity in order of similarity.

2. functional requirements.

①Preprocess sentences within the search target and store them in a list. ② Receive an input English string (query) from the user and preprocess it. ③ Calculate the similarity between the query and sentences within the search target • Similarity is based on the count of the same "word." ④ Rank the sentences based on similarity. ⑤ Output the top 10 ranked sentences to the user from the ranked sentences

3. Design and Implementation



```python
import operator

def preprocess(sentence):
    preprocessed_sentence = sentence.strip().split(" ")
    return preprocessed_sentence
```

2.input

Sentence: The string you entered

3. Result

Returns: preprocessed_sentence = a list of the words in the Sentence, split by spaces

4.Description.

Returns the input string as a list of words split by spaces.

```
def indexing(file_name):
    file_tokens_pairs = [] #빈 리스트 "file_tokens_pairs" 만들기
    lines = open(file_name, "r", encoding="utf8").readlines()  #"jhe-koen-dev.en"파일 열기
    for line in lines:
        tokens = preprocess(line) #한 문장을 하나의 리스트로 하고 공백을 기준으로 문장을 나누어 원소로 만들기
        file_tokens_pairs.append(tokens) #리스트 "file_tokens_pairs"에 바로 위에서 만들어진 리스트 추가하기
    return file_tokens_pairs
```

2.input file_name: The name of the file where the sentences are

stored

3.result

Returns: file_tokens_pairs = a two-dimensional list storing a group of sentences based on their spacing.

4.Explanation

1. create a file_tokens_pairs list

2. open a file named file_name in read mode

3. read the file line by line and create a list of tokens that are split based on newlines

4 Collect all tokens lists and create and return a two-dimensional list called file_tokens_pairs

```
def calc_similarity(preprocessed_query, preprocessed_sentences):
    score_dict = {} #각 문장의 인덱스와 유사도를 저장하기 위한 딕셔너리 생성
    for i in range(len(preprocessed_sentences)):
        #대소문자 구분을 없에는 코드
        sentence = preprocessed_sentences[i]
        query_str = ' '.join(preprocessed_query).lower()
        sentence_str = ' '.join(sentence).lower()
        preprocessed_query = set(preprocess(query_str))
        preprocessed_sentence = preprocess(sentence_str)

        file_token_set = set(preprocessed_sentence)
        all_tokens = preprocessed_query | file_token_set
        same_tokens = preprocessed_query & file_token_set
        similarity = len(same_tokens) / len(all_tokens) #두 집합의 교집합/합집합 으로 유사도 구하기
        score_dict[i] = similarity
    return score_dict
```

2.input

preprocessed_query: a variable that replaces the string entered by the user with a set of whitespacebased splits.

preprocessed_sentences: = a two-dimensional list that stores the sentences in the read file in a grouped list based on spaces.

3. result

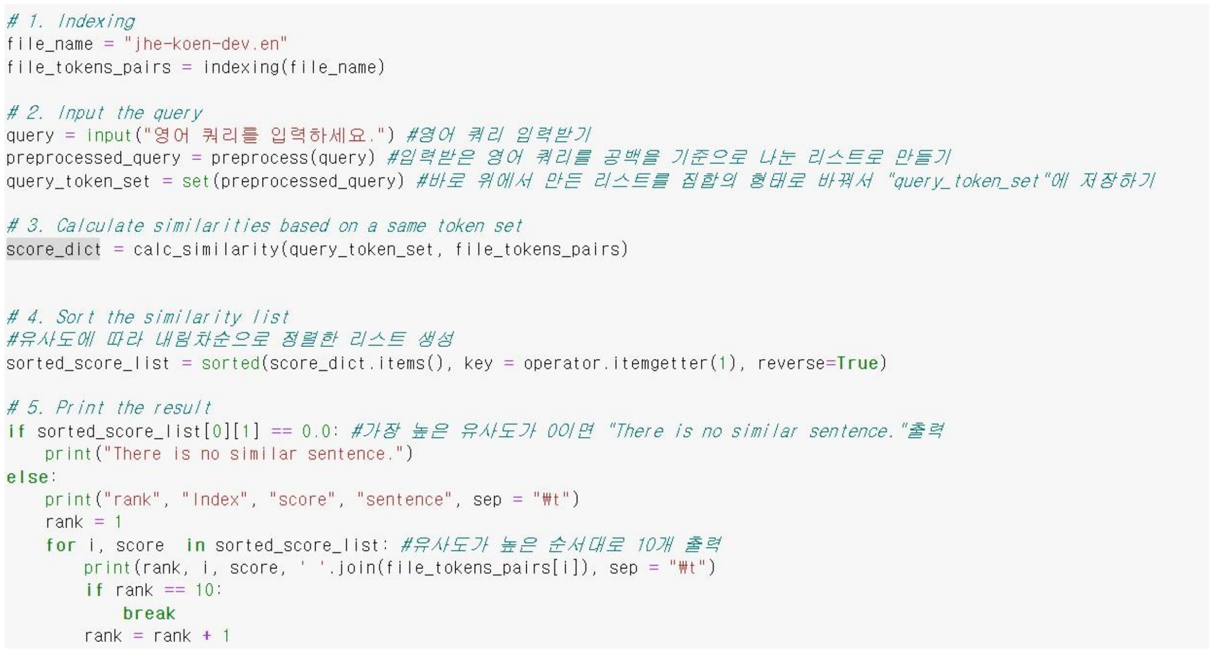
Returns: score_dict = {pure (index) of the strings stored in the file: similarity of the entered string to the strings in the file} A dictionary consisting of

4.Description

1..Create a variable to store the strings in the list and set you entered in all lowercase letters

2. use the two variables created in step 1 to find the similarity by intersection/union of two sentences

3.Store the obtained similarity in score_dict and return it



```python
# 1. Indexing
file_name = "jhe-koen-dev.en"
file_tokens_pairs = indexing(file_name)

# 2. Input the query
query = input("영어 쿼리를 입력하세요.") #영어 쿼리 입력받기
preprocessed_query = preprocess(query) #입력받은 영어 쿼리를 공백을 기준으로 나눈 리스트로 만들기
query_token_set = set(preprocessed_query) #바로 위에서 만든 리스트를 집합의 형태로 바꿔서 "query_token_set"에 저장하기

# 3. Calculate similarities based on a same token set
score_dict = calc_similarity(query_token_set, file_tokens_pairs)


# 4. Sort the similarity list
#유사도에 따라 내림차순으로 정렬한 리스트 생성
sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1), reverse=True)

# 5. Print the result
if sorted_score_list[0][1] == 0.0: #가장 높은 유사도가 0이면 "There is no similar sentence."출력
    print("There is no similar sentence.")
else:
    print("rank", "Index", "score", "sentence", sep = "₩t")
    rank = 1
    for i, score  in sorted_score_list: #유사도가 높은 순서대로 10개 출력
        print(rank, i, score, ' '.join(file_tokens_pairs[i]), sep = "₩t")
        if rank == 10:
            break
        rank = rank + 1
```

2.input query: Ask the user to enter an English

query

3.result

Return value: None because it is not a function

Output: 10 sentences with high similarity to the sentence entered by the user, and the similarity of each sentence sorted in order of similarity

4.Description.

1Preprocess sentences within the search target and store them in a list.

2 Receive an input English string (query) from the user and preprocess it.

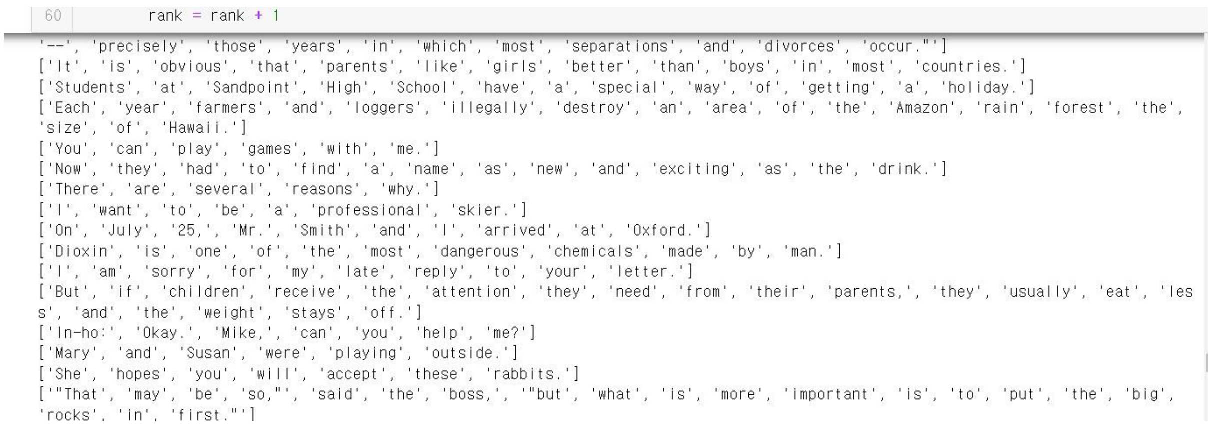
3 Calculate the similarity between the query and sentences within the search target • Similarity is based on the count of the same "word."

4 Rank the sentences based on similarity.

5 Output the top 10 ranked sentences to the user from the ranked sentences

4. Testing

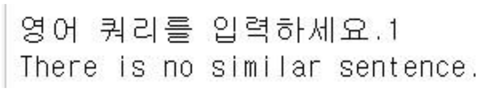
1. Test Results for Each Functionality: 대상 내 문장들을 전처리 후 리스트에 저장



```
        rank = rank + 1
'--', 'precisely', 'those', 'years', 'in', 'which', 'most', 'separations', 'and', 'divorces', 'occur."']
['It', 'is', 'obvious', 'that', 'parents', 'like', 'girls', 'better', 'than', 'boys', 'in', 'most', 'countries.']
['Students', 'at', 'Sandpoint', 'High', 'School', 'have', 'a', 'special', 'way', 'of', 'getting', 'a', 'holiday.']
['Each', 'year', 'farmers', 'and', 'loggers', 'illegally', 'destroy', 'an', 'area', 'of', 'the', 'Amazon', 'rain', 'forest', 'the',
'size', 'of', 'Hawaii.']
['You', 'can', 'play', 'games', 'with', 'me.']
['Now', 'they', 'had', 'to', 'find', 'a', 'name', 'as', 'new', 'and', 'exciting', 'as', 'the', 'drink.']
['There', 'are', 'several', 'reasons', 'why.']
['I', 'want', 'to', 'be', 'a', 'professional', 'skier.']
['On', 'July', '25,', 'Mr.', 'Smith', 'and', 'I', 'arrived', 'at', 'Oxford.']
['Dioxin', 'is', 'one', 'of', 'the', 'most', 'dangerous', 'chemicals', 'made', 'by', 'man.']
['I', 'am', 'sorry', 'for', 'my', 'late', 'reply', 'to', 'your', 'letter.']
['But', 'if', 'children', 'receive', 'the', 'attention', 'they', 'need', 'from', 'their', 'parents,', 'they', 'usually', 'eat', 'les
s', 'and', 'the', 'weight', 'stays', 'off.']
['In-ho:', 'Okay.', 'Mike,', 'can', 'you', 'help', 'me?']
['Mary', 'and', 'Susan', 'were', 'playing', 'outside.']
['She', 'hopes', 'you', 'will', 'accept', 'these', 'rabbits.']
['"That', 'may', 'be', 'so,"', 'said', 'the', 'boss,', '"but', 'what', 'is', 'more', 'important', 'is', 'to', 'put', 'the', 'big',
'rocks', 'in', 'first."']
```

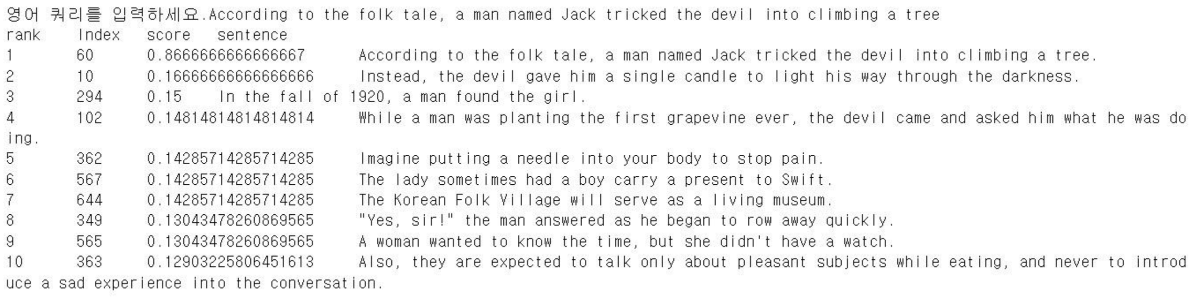
2. Final Test Screenshot –

1) 유사한 문장이 없을 경우



```
영어 쿼리를 입력하세요.1
There is no similar sentence.
```

2) 유사한 문장들이 있을 경우



```
영어 쿼리를 입력하세요.According to the folk tale, a man named Jack tricked the devil into climbing a tree
rank    Index   score   sentence
1       60      0.8666666666666667      According to the folk tale, a man named Jack tricked the devil into climbing a tree.
2       10      0.16666666666666666     Instead, the devil gave him a single candle to light his way through the darkness.
3       294     0.15    In the fall of 1920, a man found the girl.
4       102     0.14814814814814814     While a man was planting the first grapevine ever, the devil came and asked him what he was do
ing.
5       362     0.14285714285714285     Imagine putting a needle into your body to stop pain.
6       567     0.14285714285714285     The lady sometimes had a boy carry a present to Swift.
7       644     0.14285714285714285     The Korean Folk Village will serve as a living museum.
8       349     0.13043478260869565     "Yes, sir!" the man answered as he began to row away quickly.
9       565     0.13043478260869565     A woman wanted to know the time, but she didn't have a watch.
10      363     0.12903225806451613     Also, they are expected to talk only about pleasant subjects while eating, and never to introd
uce a sad experience into the conversation.
```

5. Results and Conclusion

1. Result: The development of the search engine was successfully accomplished.

2. Conclusion: I realized there was a lot more I didn't know than I thought.