1. Introduction

    1. **Project Purpose and Background:** Continue with practical exercises to apply what I've learned in Python so far.
    2. **Goal:** Creating a simple search engine based on similarity scores.

2. Requirements

    1. **User requirements:** A system that, when a user enters a specific phrase, searches and ranks sentences based on their similarity scores to the input phrase.
    2. **Functional Requirements:**

        1. **preprocess(sentence):**

        Input: The search query or each sentence in the search target.
        Output: Preprocessed search query or sentence (a set of tokens within the query or sentence).

        2. **indexing(file_name):**

        Input: The name and path of the search target file.
        Output: A set of tokens for each sentence within the file.

        3. **calc_similarity(preprocessed_query, preprocessed_sentences):**

        Input: Preprocessed search query and preprocessed sentences in the search target.
        Output: A dictionary containing file IDs and similarity scores between the query and each sentence.

        4. **sorted_score_list:**

        Input: Output the sentences ranked based on similarity.
        Output: Present the top 10 ranked sentences to the user.

        5. **if sorted_score_list[0][1] == 0.0:**

        Input: Display sentences ranked based on similarity.
        Output: Show the top 10 ranked sentences to the user.

3. Design and Implementation

1. Implementation Details:

1. Code block or function screenshot

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

2. Input

preprocess: A preprocessing function that divides each sentence in the search query or search target by spaces.

3. Output

Return Value: The preprocessed sentence.

It returns the preprocessed search query or sentence, which is a set of tokens within the query or sentence.

4. Explanation

After processing the input query sentence by dividing it based on spaces, each processed sentence is stored in preprocessed_sentence and then returned.

1. Code block or function screenshot

```
# file indexing
def indexing(file_name):
    file_tokens_pairs = []
    lines = open(file_name, "r", encoding="utf8").readlines()
    for line in lines:
        tokens = preprocess(line)
        file_tokens_pairs.append(tokens)
    return file_tokens_pairs
```

2. Input

indexing: A function that loads the name and path of the search target file, reads the file, and performs an indexing process by splitting it into lines.

3. Output

Return Value: The indexed sentences.
It returns a set of tokens for each sentence in the file.

4. Explanation

It takes the file name and path of the search target, adds each sentence that has gone through the indexing process, line by line, to the newly created file_token_pairs list.

It returns the file_token_pairs list containing the sentences that have undergone the indexing process.

1. Code block or function screenshot

```
# calculate similarity using
def calc_similarity(preprocessed_query, preprocessed_sentences):
    score_dict = {}
    for i in range(len(preprocessed_sentences)):
        # start : code for case-insensitive token set
        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)
        # end : code for case-insensitive token set

        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)
        score_dict[i] = similarity
    return score_dict
```

2. Input

calc_similarity: A function that compares preprocessed sentences in the search target with the preprocessed query sentence to calculate similarity.

3. Output

Return Value: The calculated similarity score.

It returns a dictionary containing the similarity scores between the sentences in the search target and the query sentence.

4. Explanation

It takes preprocessed sentences from the search target and the query sentence as input and calculates the similarity scores for each sentence, adding them to the newly created score_dict dictionary.

It also includes a process to measure similarity in a case-insensitive manner using the join function.

It returns the similarity scores stored in score_dict.

1. Code block or function screenshot

```
# 4. Sort the similarity list
sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1), reverse=True)
```

2. Input

sorted_score_list: A function that ranks sentences based on the calculated similarity scores and stores them in sorted_score_list.

3. Output

Return Value: None.

It ranks the calculated similarity scores and stores them in sorted_score_list.

4. Explanation

It obtains key-value pairs from the score_dict dictionary using the items() function and ranks them in descending order based on the key (score) to store them in sorted_score_list.

1. Code block or function screenshot

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

2. Input

sorted_score_list[0][1] == 0.0: A list containing the sorted similarity scores for each sentence.

3. Output

Return Value: None.

It prints the sentences and scores for the top 10 sentences with the highest similarity scores.

If there are no similar sentences, it prints a message indicating that there are none.

4. Explanation

If the score of the first item in the sorted list is 0, it determines that there are no similar sentences and prints a message to that effect.

Otherwise, it prints the rank, sentence number, and sentence for the top 1 to 10 sentences.

4. Testing

1. Test Results for Each Functionality:

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

```
# query processing
def preprocess(sentence):
    preprocessed_sentence = sentence.strip().split(" ")
    print(preprocessed_sentence)
    return preprocessed_sentence
```

```
["You'll", 'be', 'picking', 'fruit', 'and', 'generally', 'helping', 'us', 'do', 'all', 'the', 'usual', 'farm', 'work.']
['In', 'the', 'Middle', 'Ages,', 'cities', 'were', 'not', 'very', 'clean,', 'and', 'the', 'streets', 'were', 'filled', 'with', 'garb
age.']
['For', 'the', 'moment', 'they', 'may', 'yet', 'be', 'hiding', 'behind', 'their', 'apron', 'strings,', 'but', 'sooner', 'or', 'late
r', 'their', 'society', 'will', 'catch', 'up', 'with', 'the', 'progressive', 'world.']
['Do', 'you', 'know', 'what', 'the', 'cow', 'answered?"', 'said', 'the', 'minister.']
['Poland', 'and', 'Italy', 'may', 'seem', 'like', 'very', 'different', 'countries.']
['Mr.', 'Smith', 'and', 'I', 'stayed', 'the', 'whole', 'day', 'in', 'Oxford.']
['The', 'sight', 'of', 'a', 'red', 'traffic', 'signal', 'gave', 'him', 'an', 'idea.']
['So', 'they', 'used', 'pumpkins', 'instead.']
['2.', 'a', 'particular', 'occasion', 'of', 'state', 'of', 'affairs:', 'They', 'might', 'not', 'offer', 'me', 'much', 'money.']
["I'm", 'especially', 'interested', 'in', 'learning,', 'horse-riding', 'skills,', 'so', 'I', 'hope', "you'll", 'include', 'informati
on', 'about', 'this.']
['Instead,', 'the', 'devil', 'gave', 'him', 'a', 'single', 'candle', 'to', 'light', 'his', 'way', 'through', 'the', 'darkness.']
['It', 'shines', 'over', 'the', 'sea.']
['He,', 'too,', 'was', 'arrested,', 'and', 'a', 'bomb', 'was', 'thrown', 'at', 'his', 'house.']
['It', 'seems', 'that', 'the', 'high', 'temperature', 'and', 'pressure', 'on', 'the', 'star', 'made', 'its', 'carbon', 'surface', 't
```

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

```
# file indexing
def indexing(file_name):
    file_tokens_pairs = []
    lines = open(file_name, "r", encoding="utf8").readlines()
    for line in lines:
        tokens = preprocess(line)
        file_tokens_pairs.append(tokens)
    print(file_tokens_pairs)
    return file_tokens_pairs
```

```
[["You'll", 'be', 'picking', 'fruit', 'and', 'generally', 'helping', 'us', 'do', 'all', 'the', 'usual', 'farm', 'work.'], ['In', 'th
e', 'Middle', 'Ages,', 'cities', 'were', 'not', 'very', 'clean,', 'and', 'the', 'streets', 'were', 'filled', 'with', 'garbage.'],
['For', 'the', 'moment', 'they', 'may', 'yet', 'be', 'hiding', 'behind', 'their', 'apron', 'strings,', 'but', 'sooner', 'or', 'late
r', 'their', 'society', 'will', 'catch', 'up', 'with', 'the', 'progressive', 'world.'], ['Do', 'you', 'know', 'what', 'the', 'cow',
'answered?"', 'said', 'the', 'minister.'], ['Poland', 'and', 'Italy', 'may', 'seem', 'like', 'very', 'different', 'countries.'], ['M
r.', 'Smith', 'and', 'I', 'stayed', 'the', 'whole', 'day', 'in', 'Oxford.'], ['The', 'sight', 'of', 'a', 'red', 'traffic', 'signal',
'gave', 'him', 'an', 'idea.'], ['So', 'they', 'used', 'pumpkins', 'instead.'], ['2.', 'a', 'particular', 'occasion', 'of', 'state',
'of', 'affairs:', 'They', 'might', 'not', 'offer', 'me', 'much', 'money.'], ["I'm", 'especially', 'interested', 'in', 'learning,',
'horse-riding', 'skills,', 'so', 'I', 'hope', "you'll", 'include', 'information', 'about', 'this.'], ['Instead,', 'the', 'devil', 'g
ave', 'him', 'a', 'single', 'candle', 'to', 'light', 'his', 'way', 'through', 'the', 'darkness.'], ['It', 'shines', 'over', 'the',
'sea.'], ['He,', 'too,', 'was', 'arrested,', 'and', 'a', 'bomb', 'was', 'thrown', 'at', 'his', 'house.'], ['It', 'seems', 'that', 't
he', 'high', 'temperature', 'and', 'pressure', 'on', 'the', 'star', 'made', 'its', 'carbon', 'surface', 'turn', 'to', 'diamond.'],
['"The', 'pig', 'was', 'unpopular', 'while', 'the', 'cow', 'was', 'loved', 'by', 'everyone.'], ['Books', 'give', 'a', 'lot', 'of',
'things', 'to', 'us.'], ['Jimmy', 'and', 'Timmy', 'were', 'identical', 'twins.'], ['It', 'is', 'a', 'chemical', 'that', 'cause', 'ca
ncer.'], ['Ziege', 'from', 'Germany', 'and', 'Brazilian', 'superstars', 'Ronaldo', 'and', 'Roberto', 'Carlos', 'belonged', 'to', 'th
e', 'bald', 'club.'], ['Now', 'the', 'Taliban', 'are', 'gone', 'and', 'things', 'have', 'begun', 'to', 'change.'], ['Is', 'your', 's
kin', 'clear,', 'smooth,', 'and', 'shining', 'with', 'health?'], ['As', 'a', 'result,', 'there', 'is', 'a', 'great', 'deal', 'of',
```

3. Calculate the similarity between the input English string and the sentences in the search target

- Similarity is based on the number of common 'words'.

```
# calculate similarity using
def calc_similarity(preprocessed_query, preprocessed_sentences):
    score_dict = {}
    for i in range(len(preprocessed_sentences)):
        # start : code for case-insensitive token set
        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)
        # end : code for case-insensitive token set

        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)
        score_dict[i] = similarity
    print(score_dict)
    return score_dict
```

```
영어 쿼리를 입력하세요.hello my name is misoo
{0: 0.0, 1: 0.0, 2: 0.0, 3: 0.0, 4: 0.0, 5: 0.0, 6: 0.0, 7: 0.0, 8: 0.0, 9: 0.0, 10: 0.0, 11: 0.0, 12: 0.0, 13: 0.0, 14: 0.0, 15: 0.0,
16: 0.0, 17: 0.09090909090909091, 18: 0.0, 19: 0.0, 20: 0.07692307692307693, 21: 0.041666666666666664, 22: 0.0, 23: 0.0588235294117647
05, 24: 0.05555555555555555, 25: 0.0, 26: 0.0, 27: 0.0625, 28: 0.0, 29: 0.0, 30: 0.0, 31: 0.1, 32: 0.0, 33: 0.0, 34: 0.0, 35: 0.0, 36:
0.0, 37: 0.0, 38: 0.0, 39: 0.0, 40: 0.0, 41: 0.0, 42: 0.0, 43: 0.0, 44: 0.0, 45: 0.1111111111111111, 46: 0.0, 47: 0.0, 48: 0.0, 49: 0.
0, 50: 0.08333333333333333, 51: 0.08333333333333333, 52: 0.0, 53: 0.0, 54: 0.0, 55: 0.05263157894736842, 56: 0.0, 57: 0.0, 58: 0.0, 59:
0.0, 60: 0.0, 61: 0.0, 62: 0.0, 63: 0.0, 64: 0.0, 65: 0.05263157894736842, 66: 0.0, 67: 0.0, 68: 0.0, 69: 0.03571428571428571, 70: 0.06
666666666666667, 71: 0.0, 72: 0.0, 73: 0.07692307692307693, 74: 0.0, 75: 0.06666666666666667, 76: 0.0, 77: 0.08333333333333333, 78: 0.0
7692307692307693, 79: 0.0, 80: 0.047619047619047616, 81: 0.0, 82: 0.06666666666666667, 83: 0.0, 84: 0.0, 85: 0.0, 86: 0.0, 87: 0.0, 88:
0.0, 89: 0.0, 90: 0.0, 91: 0.0, 92: 0.0, 93: 0.0, 94: 0.0, 95: 0.0, 96: 0.05, 97: 0.0, 98: 0.041666666666666664, 99: 0.0, 100: 0.0, 10
1: 0.0, 102: 0.0, 103: 0.0, 104: 0.0, 105: 0.0, 106: 0.05, 107: 0.1111111111111111, 108: 0.0, 109: 0.0, 110: 0.0, 111: 0.07142857142857
142, 112: 0.07142857142857142, 113: 0.0, 114: 0.0, 115: 0.0, 116: 0.0, 117: 0.0, 118: 0.0, 119: 0.06666666666666667, 120: 0.07142857142
857142, 121: 0.0, 122: 0.0, 123: 0.0, 124: 0.0, 125: 0.0, 126: 0.0, 127: 0.0, 128: 0.038461538461538464, 129: 0.05, 130: 0.0, 131: 0.0,
132: 0.0, 133: 0.0, 134: 0.0, 135: 0.0, 136: 0.0, 137: 0.0, 138: 0.09090909090909091, 139: 0.0, 140: 0.0, 141: 0.0, 142: 0.0, 143: 0.0,
144: 0.0, 145: 0.04, 146: 0.0, 147: 0.0, 148: 0.0, 149: 0.0, 150: 0.0, 151: 0.0, 152: 0.0, 153: 0.0, 154: 0.0, 155: 0.0, 156: 0.0, 157:
0.0, 158: 0.0, 159: 0.0, 160: 0.0, 161: 0.0, 162: 0.0, 163: 0.05263157894736842, 164: 0.0, 165: 0.0, 166: 0.0, 167: 0.0, 168: 0.0, 169:
0.0, 170: 0.0, 171: 0.0, 172: 0.0, 173: 0.09090909090909091, 174: 0.043478260869565216, 175: 0.0, 176: 0.0, 177: 0.0, 178: 0.0, 179: 0.
0, 180: 0.0, 181: 0.0, 182: 0.0, 183: 0.0, 184: 0.0, 185: 0.0625, 186: 0.0, 187: 0.0, 188: 0.0, 189: 0.0, 190: 0.14285714285714285, 19
1: 0.0, 192: 0.0, 193: 0.0, 194: 0.0, 195: 0.1, 196: 0.0, 197: 0.0, 198: 0.0, 199: 0.0, 200: 0.0, 201: 0.0, 202: 0.0, 203: 0.0714285714
2857142, 204: 0.0, 205: 0.0, 206: 0.0, 207: 0.0, 208: 0.0, 209: 0.0, 210: 0.0, 211: 0.0, 212: 0.2, 213: 0.0625, 214: 0.0, 215: 0.0, 21
6: 0.038461538461538464, 217: 0.0, 218: 0.0, 219: 0.0, 220: 0.09090909090909091, 221: 0.0, 222: 0.0, 223: 0.0, 224: 0.0, 225: 0.0, 226:
0.0, 227: 0.0, 228: 0.0, 229: 0.0, 230: 0.0, 231: 0.0, 232: 0.0, 233: 0.0, 234: 0.0, 235: 0.0, 236: 0.0, 237: 0.0, 238: 0.0, 239: 0.0,
240: 0.0, 241: 0.2222222222222222, 242: 0.0, 243: 0.09090909090909091, 244: 0.05263157894736842, 245: 0.0, 246: 0.0, 247: 0.0, 248: 0.0
5, 249: 0.0, 250: 0.0, 251: 0.0, 252: 0.0, 253: 0.0, 254: 0.0, 255: 0.0625, 256: 0.0, 257: 0.0, 258: 0.0, 259: 0.0625, 260: 0.0, 261:
0.0, 262: 0.0, 263: 0.0, 264: 0.0, 265: 0.05263157894736842, 266: 0.08333333333333333, 267: 0.0588235294117647
05, 268: 0.0, 269: 0.0, 2
70: 0.0, 271: 0.0, 272: 0.0, 273: 0.05, 274: 0.0, 275: 0.0, 276: 0.1, 277: 0.0, 278: 0.0, 279: 0.0, 280: 0.047619047619047616, 281: 0.0
7692307692307693, 282: 0.0, 283: 0.0, 284: 0.06666666666666667, 285: 0.0, 286: 0.0, 287: 0.0, 288: 0.0, 289: 0.0, 290: 0.0, 291: 0.0, 2
92: 0.0, 293: 0.1111111111111111, 294: 0.0, 295: 0.0, 296: 0.0, 297: 0.0, 298: 0.0, 299: 0.0, 300: 0.04, 301: 0.0, 302: 0.0, 303: 0.0,
304: 0.07692307692307693, 305: 0.0, 306: 0.0, 307: 0.0625, 308: 0.0, 309: 0.0, 310: 0.0, 311: 0.0, 312: 0.0, 313: 0.0, 314: 0.142857142
85714285, 315: 0.0, 316: 0.03125, 317: 0.0, 318: 0.0, 319: 0.0, 320: 0.0, 321: 0.0, 322: 0.0, 323: 0.0, 324: 0.0, 325: 0.0, 326: 0.1, 3
27: 0.0, 328: 0.0, 329: 0.0, 330: 0.09090909090909091, 331: 0.0, 332: 0.0, 333: 0.0, 334: 0.06666666666666667, 335: 0.0, 336: 0.2222222
222222222, 337: 0.0, 338: 0.0, 339: 0.0, 340: 0.08333333333333333, 341: 0.0, 342: 0.0, 343: 0.0, 344: 0.0, 345: 0.0, 346: 0.0, 347: 0.
0, 348: 0.0, 349: 0.0, 350: 0.0, 351: 0.0, 352: 0.0, 353: 0.0, 354: 0.0, 355: 0.0, 356: 0.05, 357: 0.0, 358: 0.07692307692307693, 359:
0.0, 360: 0.0, 361: 0.0625, 362: 0.0, 363: 0.0, 364: 0.0, 365: 0.04, 366: 0.0, 367: 0.041666666666666664, 368: 0.047619047619047616, 36
```

4. Rank sentences based on similarity.

```
# 4. Sort the similarity list
sorted_score_list = sorted(score_dict.items(), key = operator.itemgetter(1), reverse=True)
print(sorted_score_list)
```

```
영어 쿼리를 입력하세요.hello my name is misoo
[(679, 0.5), (526, 0.2857142857142857), (538, 0.2857142857142857), (453, 0.25), (241, 0.2222222222222222), (336, 0.2222222222222222),
(212, 0.2), (505, 0.18181818181818182), (610, 0.15384615384615385), (190, 0.14285714285714285), (314, 0.14285714285714285), (710, 0.142
85714285714285), (45, 0.1111111111111111), (107, 0.1111111111111111), (293, 0.1111111111111111), (519, 0.1111111111111111), (597, 0.111
1111111111111), (667, 0.1111111111111111), (31, 0.1), (195, 0.1), (276, 0.1), (326, 0.1), (388, 0.1), (544, 0.1), (559, 0.1), (564, 0.
1), (671, 0.1), (712, 0.1), (17, 0.09090909090909091), (138, 0.09090909090909091), (173, 0.09090909090909091), (220, 0.090909090909090
1), (243, 0.09090909090909091), (330, 0.09090909090909091), (412, 0.09090909090909091), (570, 0.09090909090909091), (693, 0.09090909090
909091), (50, 0.08333333333333333), (51, 0.08333333333333333), (77, 0.08333333333333333), (266, 0.08333333333333333), (340, 0.083333333
33333333), (425, 0.08333333333333333), (436, 0.08333333333333333), (463, 0.08333333333333333), (20, 0.07692307692307693), (73, 0.076923
07692307693), (78, 0.07692307692307693), (281, 0.07692307692307693), (304, 0.07692307692307693), (358, 0.07692307692307693), (386, 0.07
692307692307693), (419, 0.07692307692307693), (111, 0.07142857142857142), (112, 0.07142857142857142), (120, 0.07142857142857142), (203,
0.07142857142857142), (432, 0.07142857142857142), (449, 0.07142857142857142), (452, 0.07142857142857142), (469, 0.07142857142857142),
(558, 0.07142857142857142), (592, 0.07142857142857142), (623, 0.07142857142857142), (675, 0.07142857142857142), (70, 0.066666666666666
7), (75, 0.06666666666666667), (82, 0.06666666666666667), (119, 0.06666666666666667), (284, 0.06666666666666667), (334, 0.066666666666
6667), (601, 0.06666666666666667), (622, 0.06666666666666667), (635, 0.06666666666666667), (705, 0.06666666666666667), (27, 0.0625), (1
85, 0.0625), (213, 0.0625), (255, 0.0625), (259, 0.0625), (307, 0.0625), (361, 0.0625), (515, 0.0625), (577, 0.0625), (655, 0.0625), (6
87, 0.0625), (23, 0.058823529411764705), (267, 0.058823529411764705), (435, 0.058823529411764705), (568, 0.058823529411764705), (614,
0.058823529411764705), (618, 0.058823529411764705), (24, 0.05555555555555555), (426, 0.05555555555555555), (579, 0.05555555555555555),
(695, 0.05555555555555555), (55, 0.05263157894736842), (65, 0.05263157894736842), (163, 0.05263157894736842), (244, 0.052631578947368
42), (265, 0.05263157894736842), (486, 0.05263157894736842), (543, 0.05263157894736842), (552, 0.05263157894736842), (587, 0.05263157894
736842), (96, 0.05), (106, 0.05), (129, 0.05), (248, 0.05), (273, 0.05), (356, 0.05), (403, 0.05), (572, 0.05), (583, 0.05), (713, 0.0
```

5. Output the top 10 sentences among the ranked sentences to the user.

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

```
영어 쿼리를 입력하세요.hello my name is misoo
rank    Index   score   sentence
1       679     0.5     My name is Mike.
2       526     0.2857142857142857      Bob is my brother.
3       538     0.2857142857142857      My hobby is traveling.
4       453     0.25    My mother is sketching them.
5       241     0.2222222222222222      My father is running with So-ra.
6       336     0.2222222222222222      My family is at the park.
7       212     0.2     My sister Betty is waiting for me.
8       505     0.18181818181818182     My little sister Annie is five years old.
9       610     0.15384615384615385     I would raise my voice and yell, "LUNCH IS READY!"
10      190     0.14285714285714285     It is Sunday.

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

2. Final Test Screenshot::

1. In case there are no similar sentences.

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

2. In case there are similar sentences.

```
영어 쿼리를 입력하세요.hello my name is misoo
rank    Index   score   sentence
1       679     0.5     My name is Mike.
2       526     0.2857142857142857      Bob is my brother.
3       538     0.2857142857142857      My hobby is traveling.
4       453     0.25    My mother is sketching them.
5       241     0.2222222222222222      My father is running with So-ra.
6       336     0.2222222222222222      My family is at the park.
7       212     0.2     My sister Betty is waiting for me.
8       505     0.18181818181818182     My little sister Annie is five years old.
9       610     0.15384615384615385     I would raise my voice and yell, "LUNCH IS READY!"
10      190     0.14285714285714285     It is Sunday.
```

5. Results and Conclusion

1. Result: I've created a simple search engine based on similarity scores.

2. Conclusion: I was able to input each piece of code separately based on their functions, but I struggled with understanding the overall flow and structuring the algorithm. Before writing the code, I will take some time to analyze it further to gain a better understanding of the overall algorithmic flow.