Python Programming and Practice

Development of a Personalized Perfume Recommendation System

Final Report

Date : 2023.12.23

Name : Sabin Wang

ID : 213407

1. Introduction

1) Background

The perfume market continues to grow steadily, unaffected by external factors such as economic uncertainty. Consequently, the industry anticipates the perfume market to remain stable for the foreseeable future. Additionally, with the diminishing impact of the COVID-19 pandemic, the reduced usage of masks has led to increased interest in perfumes. However, the expansion of the perfume market has resulted in the release of numerous products, making it challenging for customers to find a fragrance that suits them. Furthermore, exposure to too many scents at once can lead to olfactory fatigue. Therefore, to address these issues, a system recommending customized perfumes to customers is deemed necessary.

2) Project goal

The goal is to develop a system that, when provided with information about a customer's favorite or preferred fragrance, recommends similar perfumes to enhance customer satisfaction.

3) Differences from existing programs

The existing programs recommend new perfumes based on the customer's past purchase history. However, these recommendations may not align with the customer's current preferences as they rely on outdated buying patterns. In this system, the customer directly inputs their favorite fragrance, and the system recommends perfumes with similar characteristics. Additionally, the system takes into account not only the fragrance notes but also the description (feel and features) to provide recommendations, offering a differentiation from existing programs.

2. Functional Requirement

1) Recommendation of Perfumes Similar to Previously Used Ones

- If the user has a previously used perfume, they can enter its name to receive recommendations for similar perfumes.

- Before running the program, similar perfumes are categorized using pre-prepared data.

(1) When the entered perfume name exists in the data

- One similar perfume is displayed, and the user is asked whether they want to save it to the wishlist, get more perfume recommendations, or exit the program.

(2) When the entered perfume name does not exist in the data

- The user is asked whether they want to get other perfume recommendations using the Notes input method, or exit the program.

2) Recommendation of New Perfumes Through Notes/Description Input

- The user is asked whether they want to receive perfume recommendations through Notes input or Description input.

(1) Notes Input

- The user inputs Notes, separating each word with a comma, and the program recommends the perfume most similar to the entered Notes.

(2) Description Input

- The user inputs a Description, separating each word with a space, and the program recommends the perfume most similar to the entered Description.

3) Perfume Recommendation Based on the User's Mood

- The user is asked to input their current mood from the five emotions provided [happiness, sadness, anger, calmness, love], and a perfume matching that mood is recommended.

- Data categorized beforehand according to these five emotions is used.

(1) When there is a perfume matching the emotion entered by the user

- One similar perfume is displayed, and the user is asked whether they want to save it to the wishlist, receive more perfume recommendations, or exit the program.

4) Saving Recommended Perfumes to the Wishlist

- Save the recommended perfumes in a txt format wishlist file.

(1) Add to wishlist

- Add the recommended perfume to the wishlist, or directly enter the name of a perfume to add it to the wishlist.

(2) Display wishlist

- Load the wishlist file and display its contents.

(3) Save wishlist

- Save the wishlist to a file.

(4) Delete wishlist

- Delete a specific item from the wishlist by its number.

3. Implementation

(1) Main Module (Initial Menu Execution) – main.py

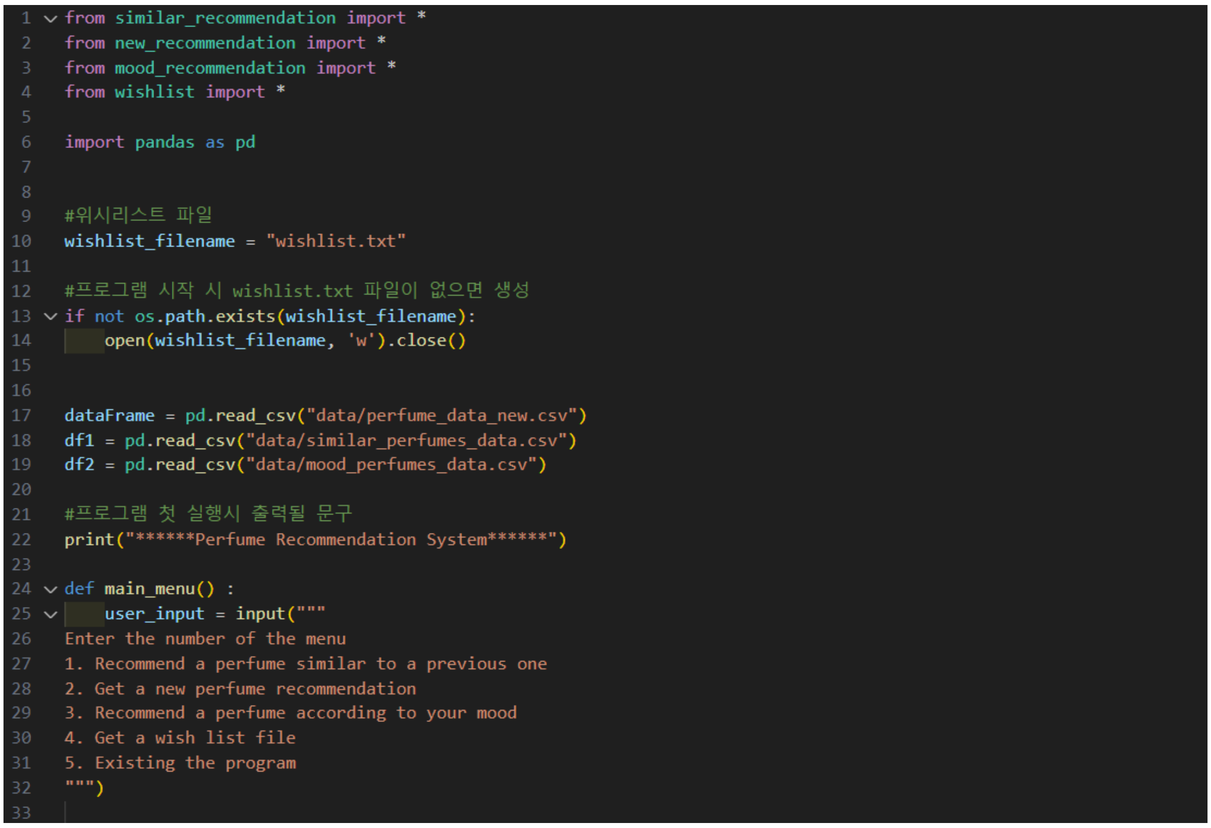
- Input : Menu number desired by the user

- Output : Execution of the respective module based on the entered menu number

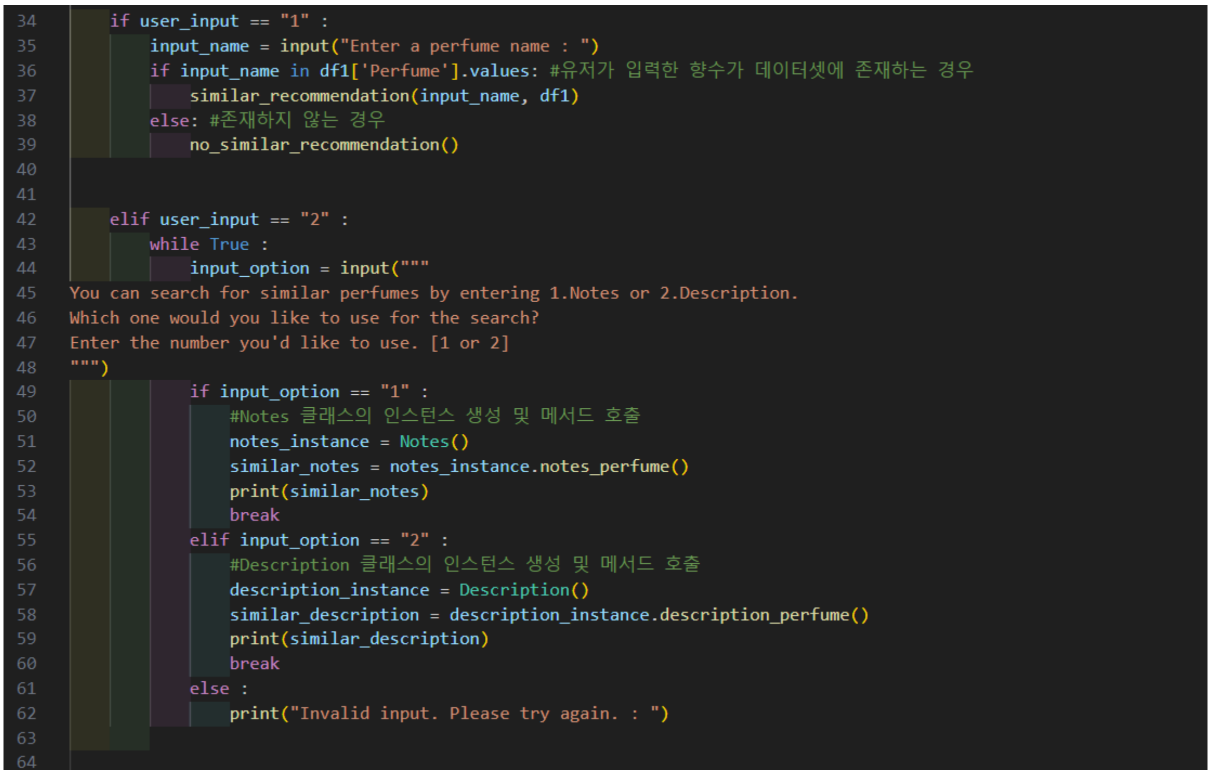
- Explanation : Upon program execution, the initial menu is displayed. When the user inputs the number corresponding to their desired menu, the program calls and executes the function associated with that menu. For efficient code modification and execution, each functionality has been modularized and imported into main.py.

- Applied Concepts : When the program is run, if the wishlist.txt file does not exist, it is created using file input/output operations. During the process where the main function is executed upon program startup, this function is utilized. The user is prompted to enter the number of the desired menu, and the corresponding menu is executed using conditional statements, loops, and modules.

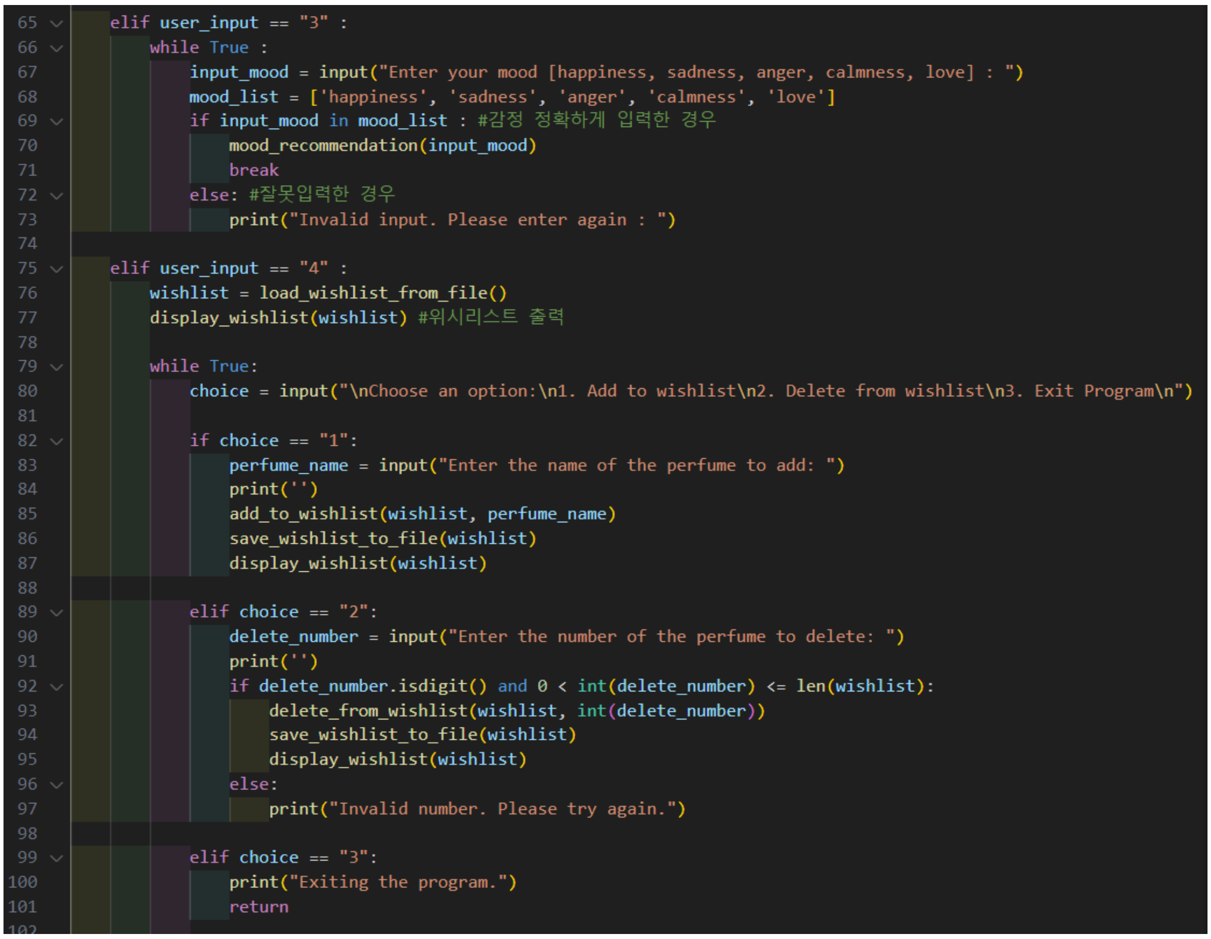
- Code screenshot



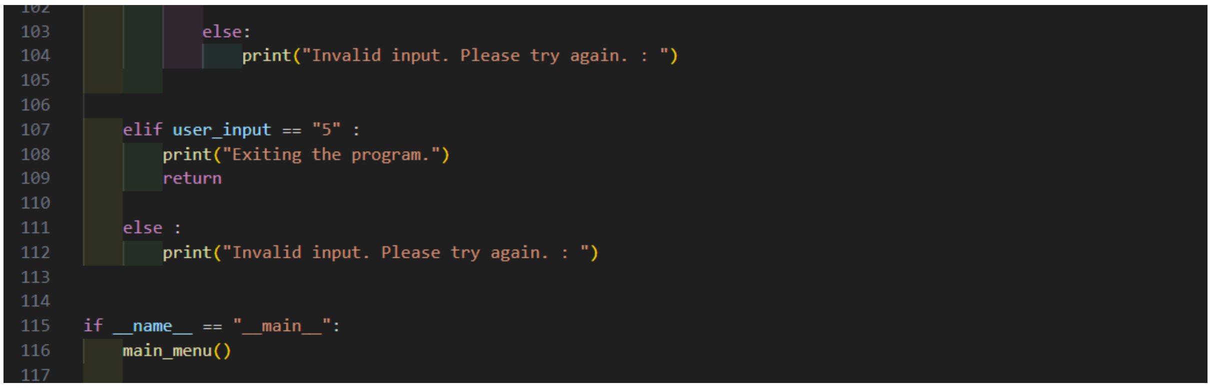
```python
from similar_recommendation import *
from new_recommendation import *
from mood_recommendation import *
from wishlist import *

import pandas as pd


#위시리스트 파일
wishlist_filename = "wishlist.txt"

#프로그램 시작 시 wishlist.txt 파일이 없으면 생성
if not os.path.exists(wishlist_filename):
    open(wishlist_filename, 'w').close()


dataFrame = pd.read_csv("data/perfume_data_new.csv")
df1 = pd.read_csv("data/similar_perfumes_data.csv")
df2 = pd.read_csv("data/mood_perfumes_data.csv")

#프로그램 첫 실행시 출력될 문구
print("******Perfume Recommendation System******")

def main_menu() :
    user_input = input("""
Enter the number of the menu
1. Recommend a perfume similar to a previous one
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
""")

```



```python
    if user_input == "1" :
        input_name = input("Enter a perfume name : ")
        if input_name in df1['Perfume'].values: #유저가 입력한 향수가 데이터셋에 존재하는 경우
            similar_recommendation(input_name, df1)
        else: #존재하지 않는 경우
            no_similar_recommendation()


    elif user_input == "2" :
        while True :
            input_option = input("""
You can search for similar perfumes by entering 1.Notes or 2.Description.
Which one would you like to use for the search?
Enter the number you'd like to use. [1 or 2]
""")
            if input_option == "1" :
                #Notes 클래스의 인스턴스 생성 및 메서드 호출
                notes_instance = Notes()
                similar_notes = notes_instance.notes_perfume()
                print(similar_notes)
                break
            elif input_option == "2" :
                #Description 클래스의 인스턴스 생성 및 메서드 호출
                description_instance = Description()
                similar_description = description_instance.description_perfume()
                print(similar_description)
                break
            else :
                print("Invalid input. Please try again. : ")

```



```python
    elif user_input == "3" :
        while True :
            input_mood = input("Enter your mood [happiness, sadness, anger, calmness, love] : ")
            mood_list = ['happiness', 'sadness', 'anger', 'calmness', 'love']
            if input_mood in mood_list : #감정 정확하게 입력한 경우
                mood_recommendation(input_mood)
                break
            else: #잘못입력한 경우
                print("Invalid input. Please enter again : ")

    elif user_input == "4" :
        wishlist = load_wishlist_from_file()
        display_wishlist(wishlist) #위시리스트 출력

        while True:
            choice = input("\nChoose an option:\n1. Add to wishlist\n2. Delete from wishlist\n3. Exit Program\n")

            if choice == "1":
                perfume_name = input("Enter the name of the perfume to add: ")
                print('')
                add_to_wishlist(wishlist, perfume_name)
                save_wishlist_to_file(wishlist)
                display_wishlist(wishlist)

            elif choice == "2":
                delete_number = input("Enter the number of the perfume to delete: ")
                print('')
                if delete_number.isdigit() and 0 < int(delete_number) <= len(wishlist):
                    delete_from_wishlist(wishlist, int(delete_number))
                    save_wishlist_to_file(wishlist)
                    display_wishlist(wishlist)
                else:
                    print("Invalid number. Please try again.")

            elif choice == "3":
                print("Exiting the program.")
                return
```



```python
            else:
                print("Invalid input. Please try again. : ")

    elif user_input == "5" :
        print("Exiting the program.")
        return

    else :
        print("Invalid input. Please try again. : ")


if __name__ == "__main__":
    main_menu()
```

(2) Change File Encoding – change_file_encoding.py

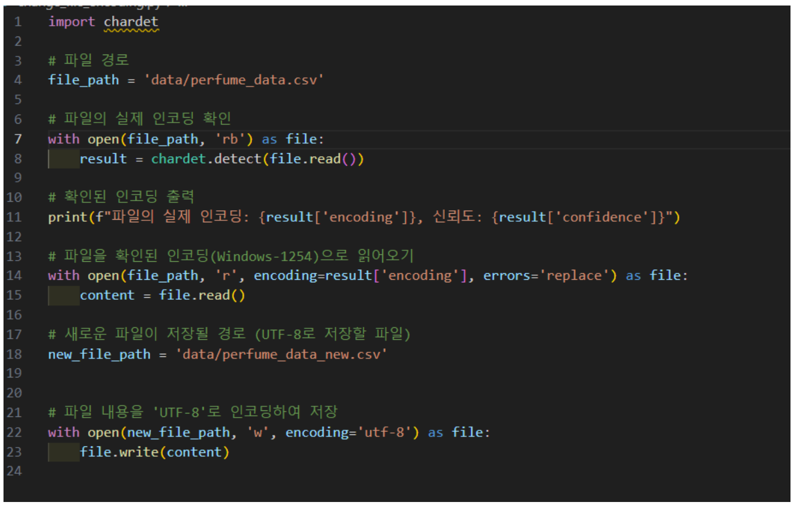
- Input : Original data file downloaded from Kaggle (perfume_data.csv)

- Output : New data file encoded in 'UTF-8' (perfume_data_new.csv)

- Explanation : An issue arose while loading the original data file downloaded from Kaggle due to its encoding being 'Windows-1254.' Therefore, the file contents were encoded in 'UTF-8' and saved.

- Applied Concepts : File input/output operations are used in the process of loading a file and saving it again after changing its encoding.

- Code screenshot



```python
import chardet

# 파일 경로
file_path = 'data/perfume_data.csv'

# 파일의 실제 인코딩 확인
with open(file_path, 'rb') as file:
    result = chardet.detect(file.read())

# 확인된 인코딩 출력
print(f"파일의 실제 인코딩: {result['encoding']}, 신뢰도: {result['confidence']}")

# 파일을 확인된 인코딩(Windows-1254)으로 읽어오기
with open(file_path, 'r', encoding=result['encoding'], errors='replace') as file:
    content = file.read()

# 새로운 파일이 저장될 경로 (UTF-8로 저장할 파일)
new_file_path = 'data/perfume_data_new.csv'


# 파일 내용을 'UTF-8'로 인코딩하여 저장
with open(new_file_path, 'w', encoding='utf-8') as file:
    file.write(content)
```

(3) Recommendation of Perfumes Similar to Previously Used Ones – similar_perfumes_data_generate.py

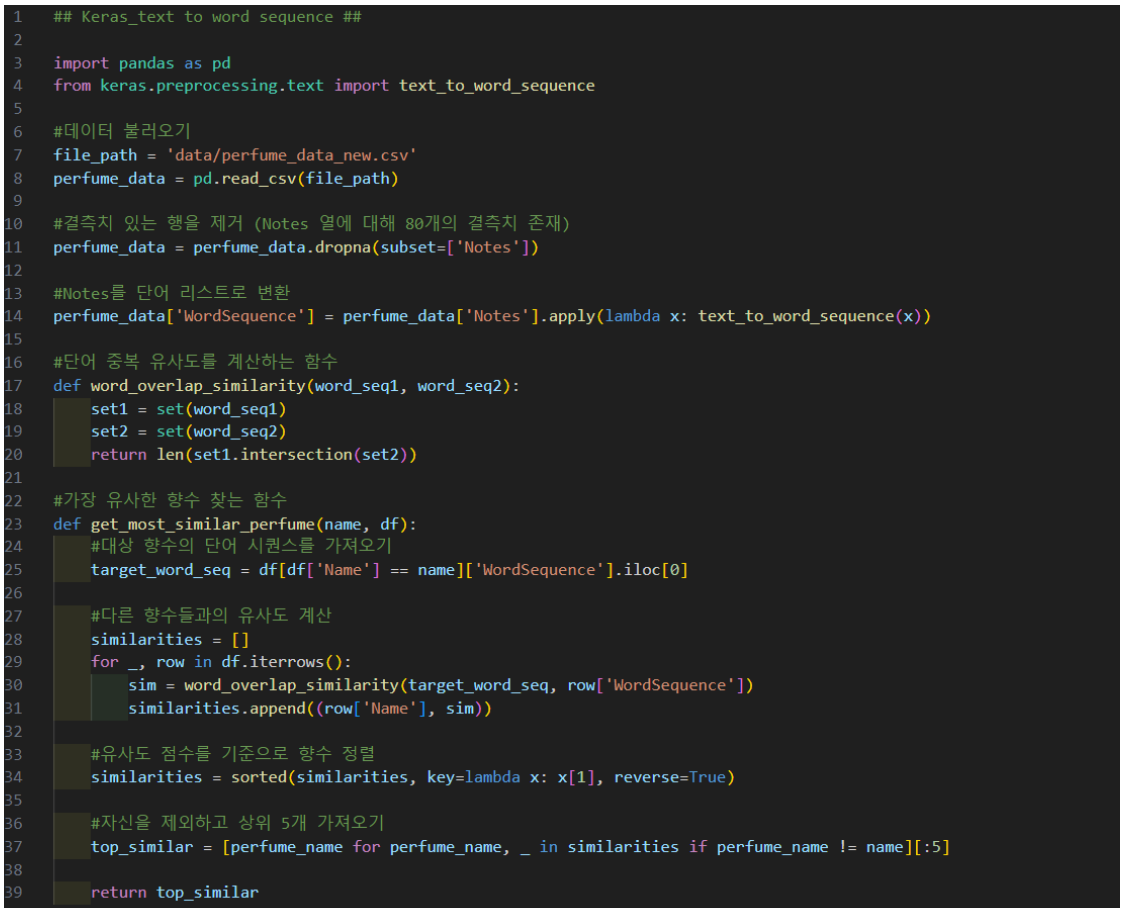
- Input : 'perfume_data_new.csv'

- Output : 'similar_perfumes_data.csv'

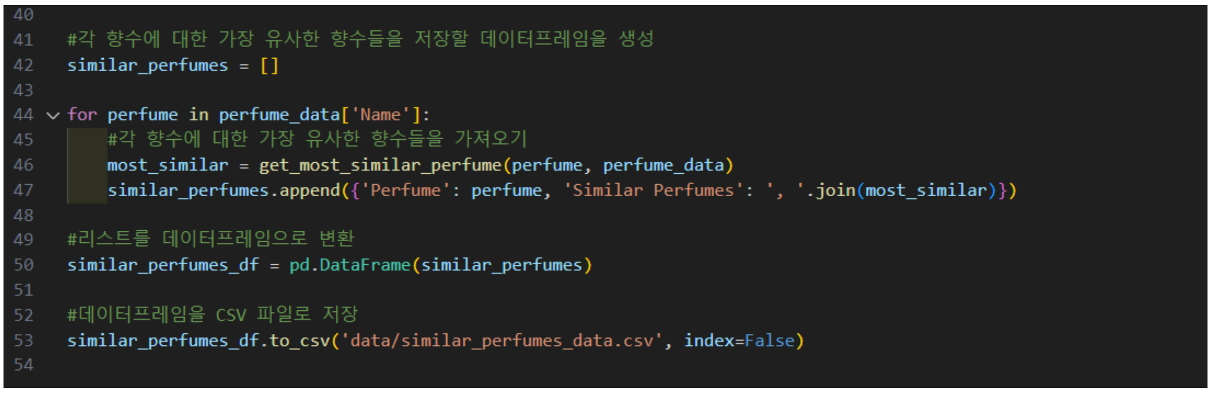
- Explanation : To implement the functionality of recommending perfumes similar to the one entered by the user, a dataset was created where perfumes with similar Notes are clustered. To enhance efficiency, it was deemed more effective to perform the calculation once during the initial phase and store it in a data file for subsequent use, rather than recalculating every time.

- Applied Concepts : Functions, file input/output, and loops are used in the process of loading data files, calculating the similarity of duplicate words, finding the most similar perfume, and then saving the results back to a file.

- Code screenshot



```python
## Keras_text to word sequence ##

import pandas as pd
from keras.preprocessing.text import text_to_word_sequence

#데이터 불러오기
file_path = 'data/perfume_data_new.csv'
perfume_data = pd.read_csv(file_path)

#결측치 있는 행을 제거 (Notes 열에 대해 80개의 결측치 존재)
perfume_data = perfume_data.dropna(subset=['Notes'])

#Notes를 단어 리스트로 변환
perfume_data['WordSequence'] = perfume_data['Notes'].apply(lambda x: text_to_word_sequence(x))

#단어 중복 유사도를 계산하는 함수
def word_overlap_similarity(word_seq1, word_seq2):
    set1 = set(word_seq1)
    set2 = set(word_seq2)
    return len(set1.intersection(set2))

#가장 유사한 향수 찾는 함수
def get_most_similar_perfume(name, df):
    #대상 향수의 단어 시퀀스를 가져오기
    target_word_seq = df[df['Name'] == name]['WordSequence'].iloc[0]

    #다른 향수들과의 유사도 계산
    similarities = []
    for _, row in df.iterrows():
        sim = word_overlap_similarity(target_word_seq, row['WordSequence'])
        similarities.append((row['Name'], sim))

    #유사도 점수를 기준으로 향수 정렬
    similarities = sorted(similarities, key=lambda x: x[1], reverse=True)

    #자신을 제외하고 상위 5개 가져오기
    top_similar = [perfume_name for perfume_name, _ in similarities if perfume_name != name][:5]

    return top_similar
```


```python

#각 향수에 대한 가장 유사한 향수들을 저장할 데이터프레임을 생성
similar_perfumes = []

for perfume in perfume_data['Name']:
    #각 향수에 대한 가장 유사한 향수들을 가져오기
    most_similar = get_most_similar_perfume(perfume, perfume_data)
    similar_perfumes.append({'Perfume': perfume, 'Similar Perfumes': ', '.join(most_similar)})

#리스트를 데이터프레임으로 변환
similar_perfumes_df = pd.DataFrame(similar_perfumes)

#데이터프레임을 csv 파일로 저장
similar_perfumes_df.to_csv('data/similar_perfumes_data.csv', index=False)

```

(4) Recommendation of Perfumes Similar to Previously Used Ones – similar_recommendation.py

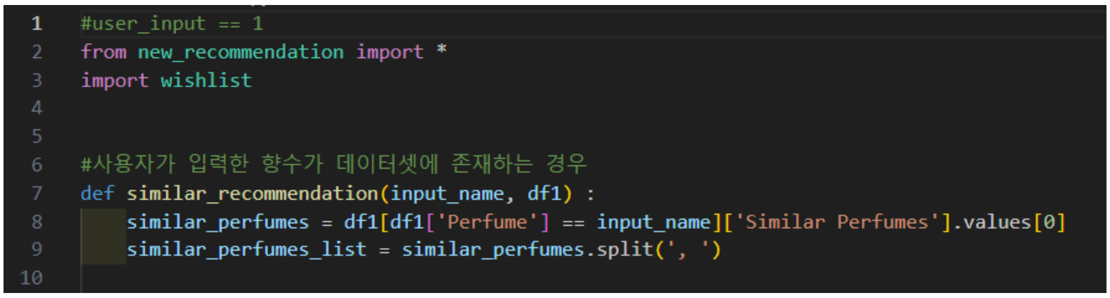
- Input : Name of the perfume entered by the use

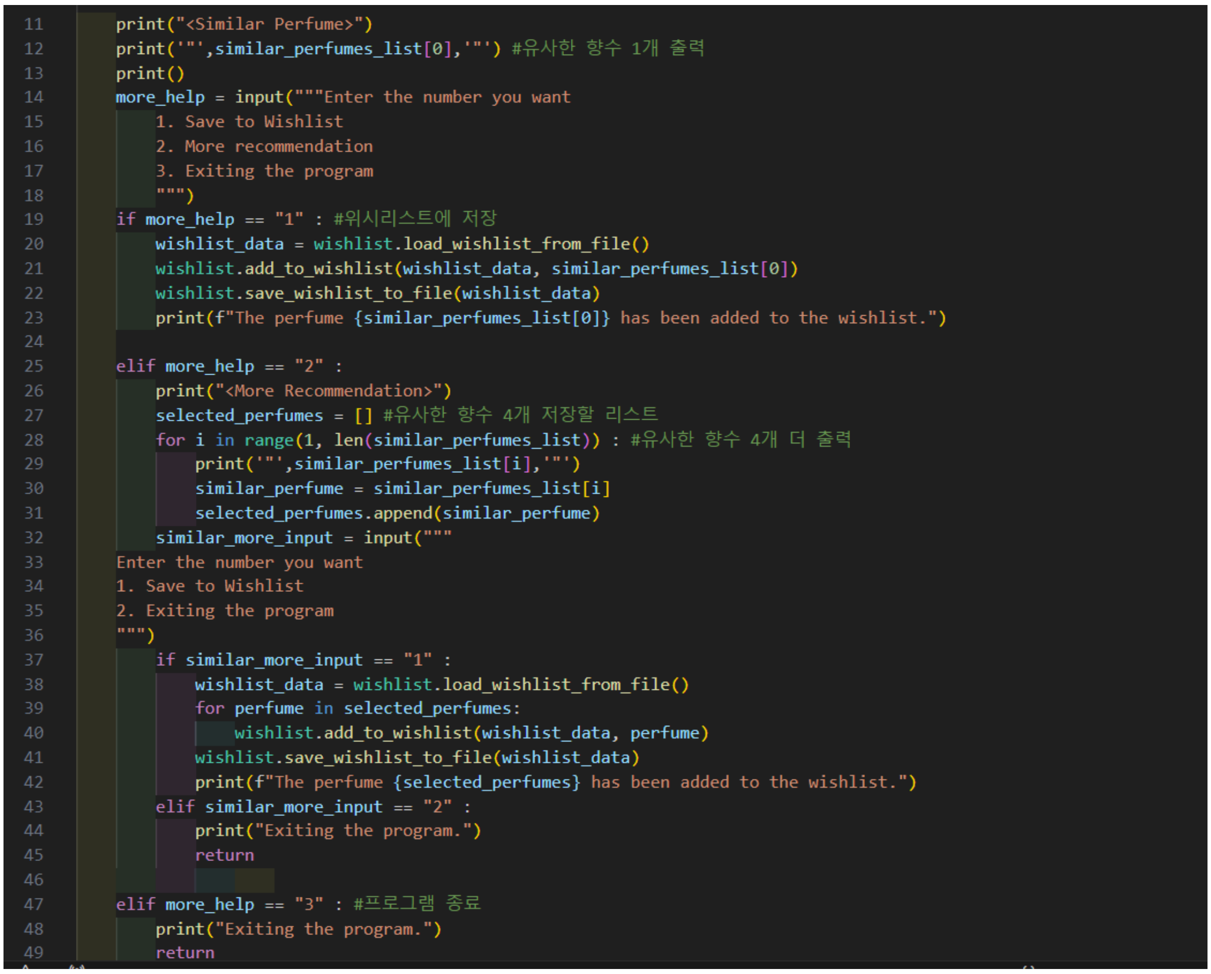
- Output : Display the most similar perfume to the entered one

- Explanation : If the perfume entered by the user exists in 'similar_perfumes_data.csv,' the system outputs the most similar perfume. To provide additional recommendations, the system can also display four more similar perfumes if the user desires more suggestions. If the entered perfume is not found in 'similar_perfumes_data.csv,' the system calls a function to inquire whether the user wants to search for the perfume using a different method or return to the initial menu.

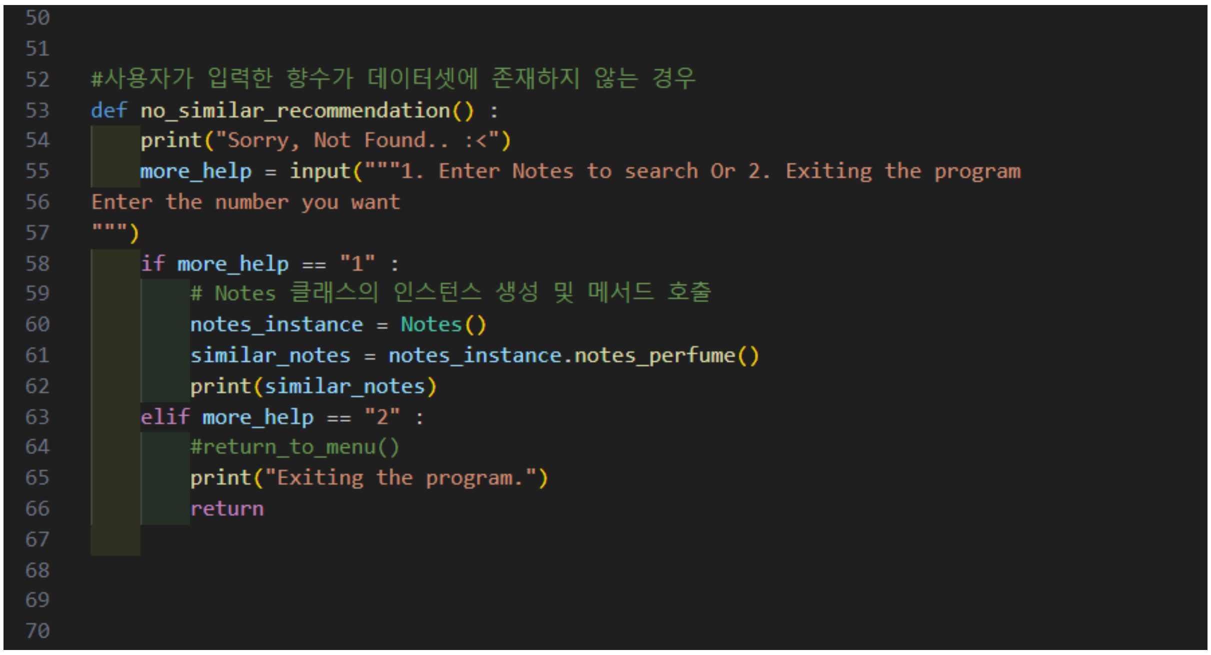
- Applied Concepts : Modules, file input/output, loops, and conditional statements are used in the process of outputting results for cases where the perfume entered by the user exists in the data and where it does not.

- Code screenshot


```python
#user_input == 1
from new_recommendation import *
import wishlist


#사용자가 입력한 향수가 데이터셋에 존재하는 경우
def similar_recommendation(input_name, df1) :
    similar_perfumes = df1[df1['Perfume'] == input_name]['Similar Perfumes'].values[0]
    similar_perfumes_list = similar_perfumes.split(', ')

```



```python
    print("<Similar Perfume>")
    print('"',similar_perfumes_list[0],'"') #유사한 향수 1개 출력
    print()
    more_help = input("""Enter the number you want
        1. Save to Wishlist
        2. More recommendation
        3. Exiting the program
        """)
    if more_help == "1" : #위시리스트에 저장
        wishlist_data = wishlist.load_wishlist_from_file()
        wishlist.add_to_wishlist(wishlist_data, similar_perfumes_list[0])
        wishlist.save_wishlist_to_file(wishlist_data)
        print(f"The perfume {similar_perfumes_list[0]} has been added to the wishlist.")

    elif more_help == "2" :
        print("<More Recommendation>")
        selected_perfumes = [] #유사한 향수 4개 저장할 리스트
        for i in range(1, len(similar_perfumes_list)) : #유사한 향수 4개 더 출력
            print('"',similar_perfumes_list[i],'"')
            similar_perfume = similar_perfumes_list[i]
            selected_perfumes.append(similar_perfume)
        similar_more_input = input("""
    Enter the number you want
    1. Save to Wishlist
    2. Exiting the program
    """)
        if similar_more_input == "1" :
            wishlist_data = wishlist.load_wishlist_from_file()
            for perfume in selected_perfumes:
                wishlist.add_to_wishlist(wishlist_data, perfume)
            wishlist.save_wishlist_to_file(wishlist_data)
            print(f"The perfume {selected_perfumes} has been added to the wishlist.")
        elif similar_more_input == "2" :
            print("Exiting the program.")
            return

    elif more_help == "3" : #프로그램 종료
        print("Exiting the program.")
        return
```



```python


#사용자가 입력한 향수가 데이터셋에 존재하지 않는 경우
def no_similar_recommendation() :
    print("Sorry, Not Found.. :<")
    more_help = input("""1. Enter Notes to search Or 2. Exiting the program
Enter the number you want
""")
    if more_help == "1" :
        # Notes 클래스의 인스턴스 생성 및 메서드 호출
        notes_instance = Notes()
        similar_notes = notes_instance.notes_perfume()
        print(similar_notes)
    elif more_help == "2" :
        #return_to_menu()
        print("Exiting the program.")
        return
```

(5) Recommendation of New Perfumes Through Notes/Description Input – new_recommendation.py

- Input : Notes or Description entered by the user

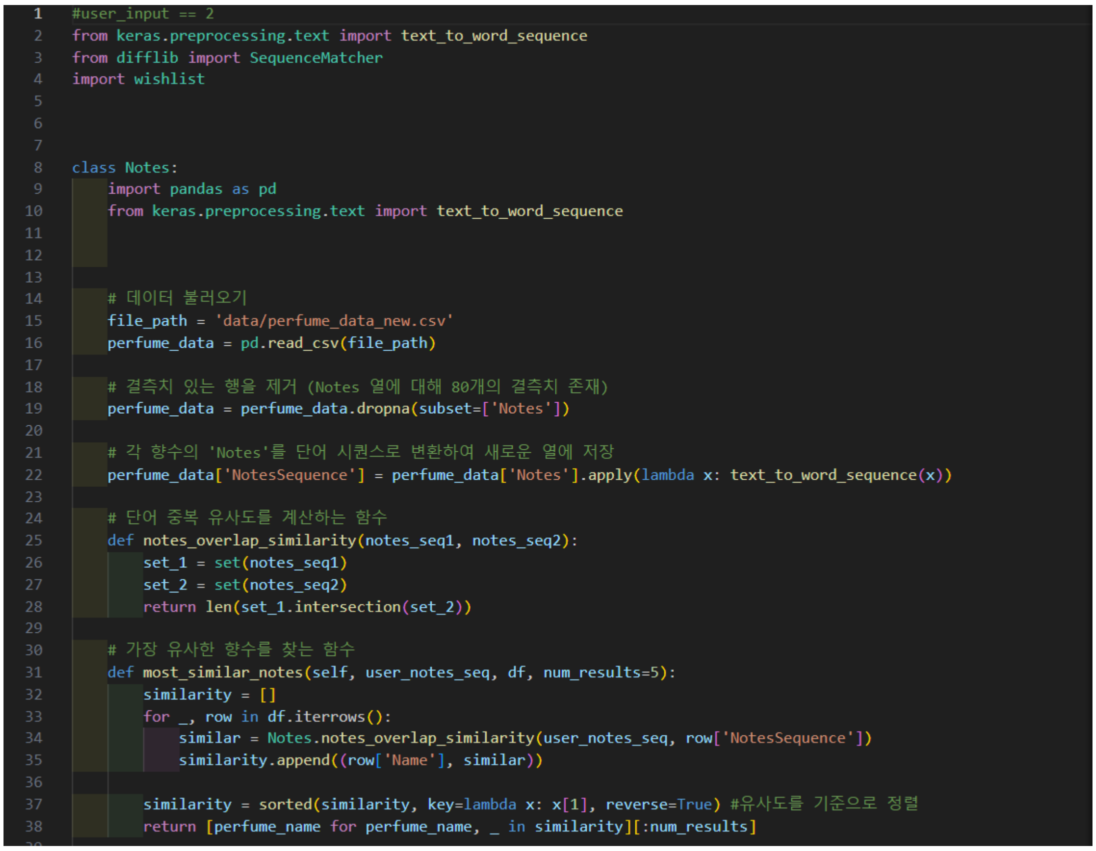
- Output : The perfume most similar to the entered Notes or Description

- Explanation : <Notes> The system takes user input for Notes and calculates similarity by counting the overlapping words with the dataset's Notes and outputs and displays the most similar perfume.

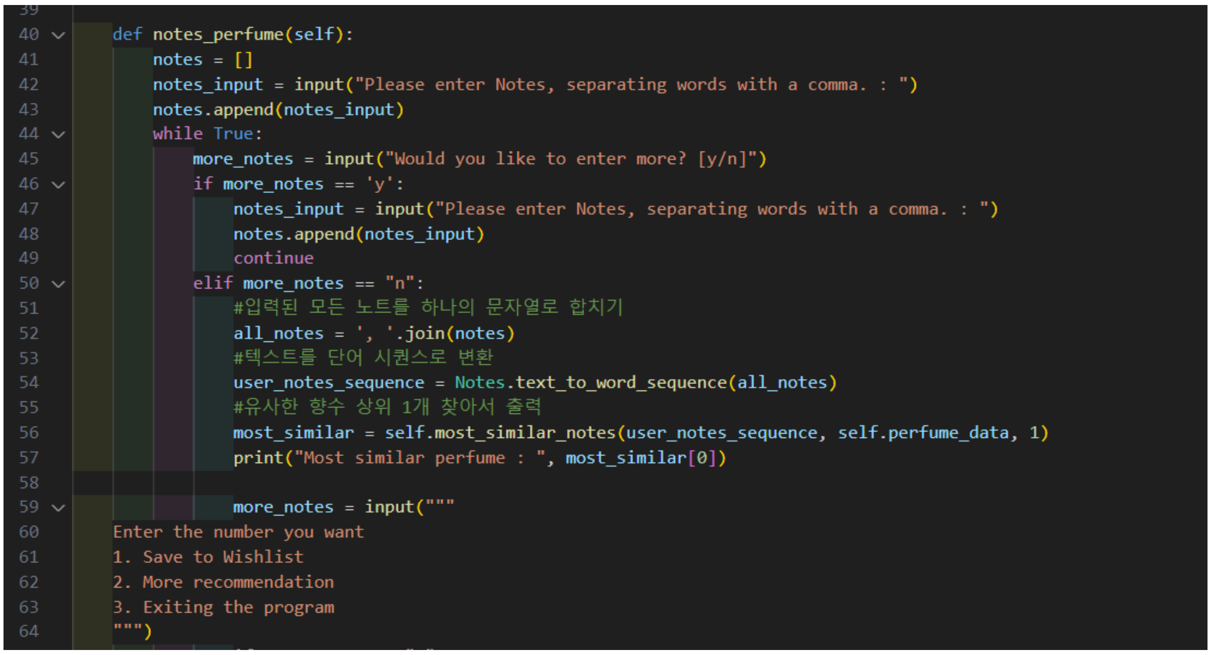
<Description> The system takes user input for Description and calculates similarity by using SequenceMatcher library with the dataset's Description. Subsequently, it outputs and displays the most similar perfume.

- Applied Concepts : The process of finding and outputting the most similar perfume based on the Notes or Description entered by the user involves functions, loops, conditional statements, modules, file input/output, and class.

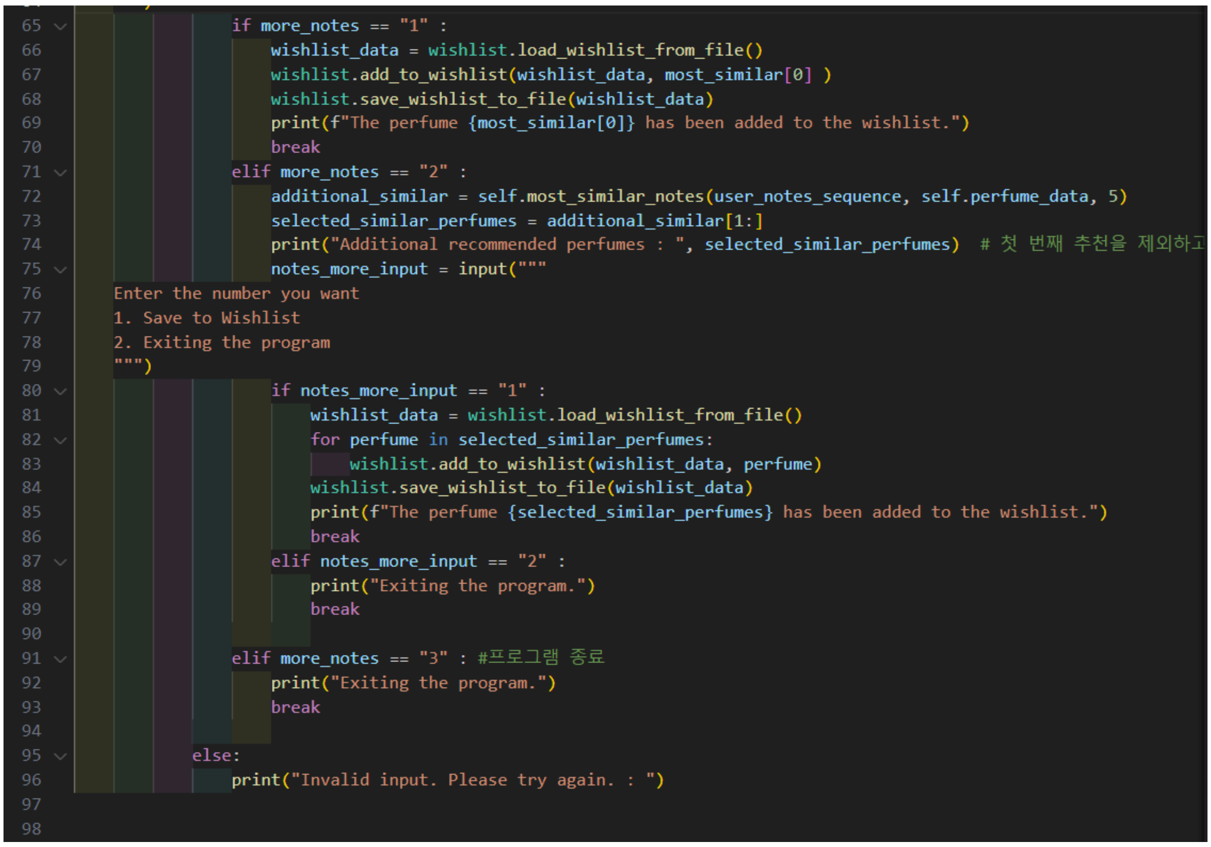
- Code screenshot



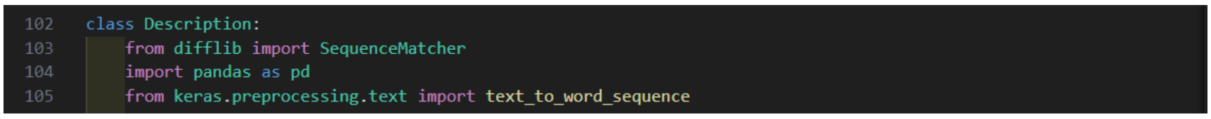
```python
#user_input == 2
from keras.preprocessing.text import text_to_word_sequence
from difflib import SequenceMatcher
import wishlist



class Notes:
    import pandas as pd
    from keras.preprocessing.text import text_to_word_sequence


    # 데이터 불러오기
    file_path = 'data/perfume_data_new.csv'
    perfume_data = pd.read_csv(file_path)

    # 결측치 있는 행을 제거 (Notes 열에 대해 80개의 결측치 존재)
    perfume_data = perfume_data.dropna(subset=['Notes'])

    # 각 향수의 'Notes'를 단어 시퀀스로 변환하여 새로운 열에 저장
    perfume_data['NotesSequence'] = perfume_data['Notes'].apply(lambda x: text_to_word_sequence(x))

    # 단어 중복 유사도를 계산하는 함수
    def notes_overlap_similarity(notes_seq1, notes_seq2):
        set_1 = set(notes_seq1)
        set_2 = set(notes_seq2)
        return len(set_1.intersection(set_2))

    # 가장 유사한 향수를 찾는 함수
    def most_similar_notes(self, user_notes_seq, df, num_results=5):
        similarity = []
        for _, row in df.iterrows():
            similar = Notes.notes_overlap_similarity(user_notes_seq, row['NotesSequence'])
            similarity.append((row['Name'], similar))

        similarity = sorted(similarity, key=lambda x: x[1], reverse=True) #유사도를 기준으로 정렬
        return [perfume_name for perfume_name, _ in similarity][:num_results]
```



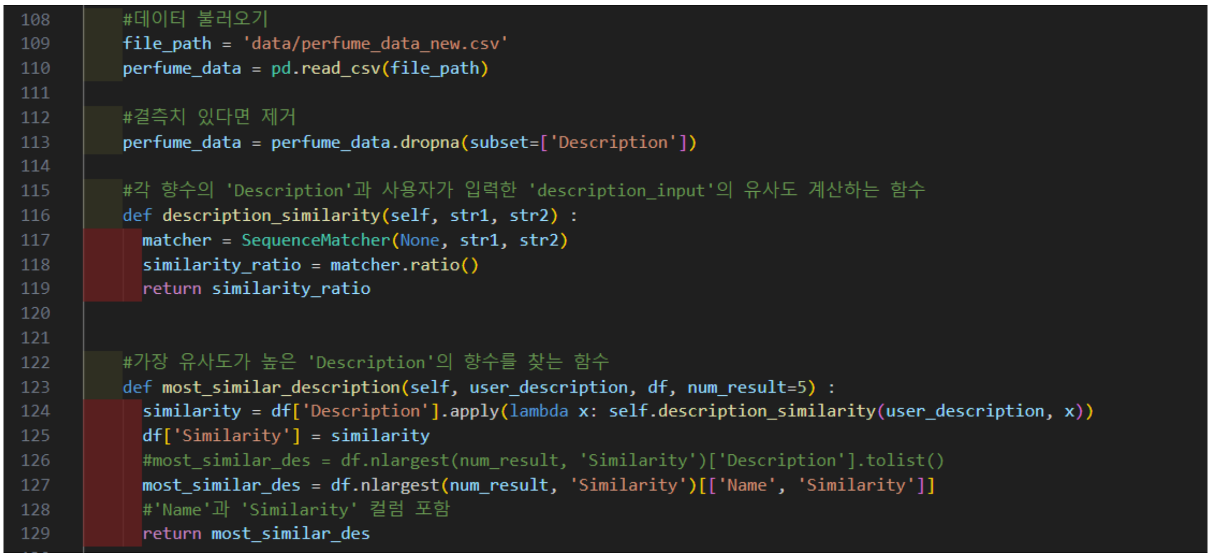
```python
    def notes_perfume(self):
        notes = []
        notes_input = input("Please enter Notes, separating words with a comma. : ")
        notes.append(notes_input)
        while True:
            more_notes = input("Would you like to enter more? [y/n]")
            if more_notes == 'y':
                notes_input = input("Please enter Notes, separating words with a comma. : ")
                notes.append(notes_input)
                continue
            elif more_notes == "n":
                #입력된 모든 노트를 하나의 문자열로 합치기
                all_notes = ', '.join(notes)
                #텍스트를 단어 시퀀스로 변환
                user_notes_sequence = Notes.text_to_word_sequence(all_notes)
                #유사한 향수 상위 1개 찾아서 출력
                most_similar = self.most_similar_notes(user_notes_sequence, self.perfume_data, 1)
                print("Most similar perfume : ", most_similar[0])

                more_notes = input("""
    Enter the number you want
    1. Save to Wishlist
    2. More recommendation
    3. Exiting the program
    """)
```



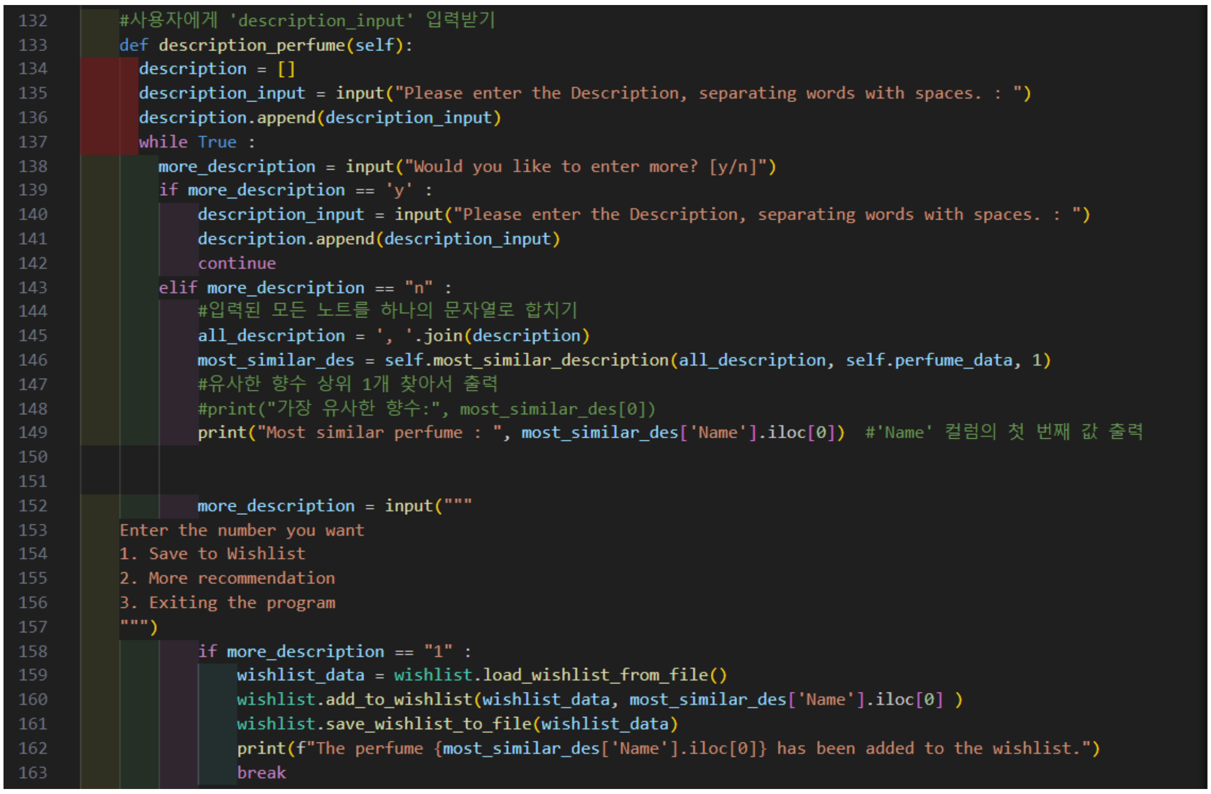
```python
                if more_notes == "1" :
                    wishlist_data = wishlist.load_wishlist_from_file()
                    wishlist.add_to_wishlist(wishlist_data, most_similar[0] )
                    wishlist.save_wishlist_to_file(wishlist_data)
                    print(f"The perfume {most_similar[0]} has been added to the wishlist.")
                    break
                elif more_notes == "2" :
                    additional_similar = self.most_similar_notes(user_notes_sequence, self.perfume_data, 5)
                    selected_similar_perfumes = additional_similar[1:]
                    print("Additional recommended perfumes : ", selected_similar_perfumes)  # 첫 번째 추천을 제외하
                    notes_more_input = input("""
    Enter the number you want
    1. Save to Wishlist
    2. Exiting the program
    """)
                    if notes_more_input == "1" :
                        wishlist_data = wishlist.load_wishlist_from_file()
                        for perfume in selected_similar_perfumes:
                            wishlist.add_to_wishlist(wishlist_data, perfume)
                        wishlist.save_wishlist_to_file(wishlist_data)
                        print(f"The perfume {selected_similar_perfumes} has been added to the wishlist.")
                        break
                    elif notes_more_input == "2" :
                        print("Exiting the program.")
                        break

                elif more_notes == "3" : #프로그램 종료
                    print("Exiting the program.")
                    break

            else:
                print("Invalid input. Please try again. : ")


```



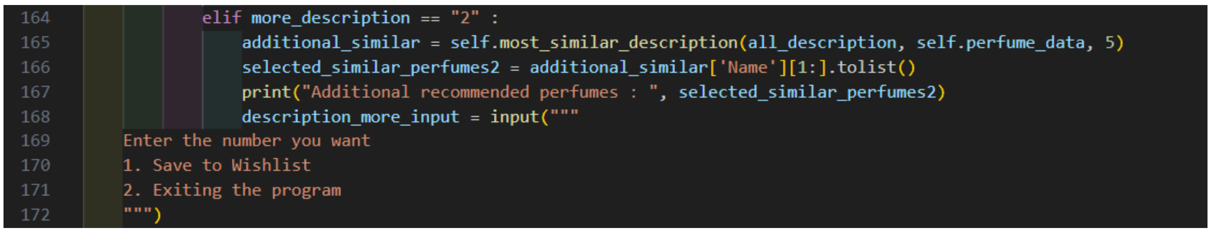
```python
class Description:
    from difflib import SequenceMatcher
    import pandas as pd
    from keras.preprocessing.text import text_to_word_sequence
```



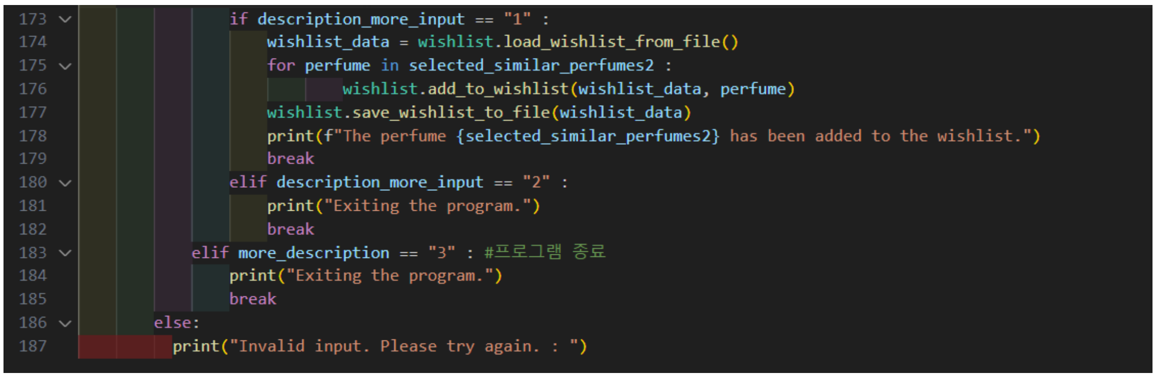
```python
    #데이터 불러오기
    file_path = 'data/perfume_data_new.csv'
    perfume_data = pd.read_csv(file_path)

    #결측치 있다면 제거
    perfume_data = perfume_data.dropna(subset=['Description'])

    #각 향수의 'Description'과 사용자가 입력한 'description_input'의 유사도 계산하는 함수
    def description_similarity(self, str1, str2) :
        matcher = SequenceMatcher(None, str1, str2)
        similarity_ratio = matcher.ratio()
        return similarity_ratio


    #가장 유사도가 높은 'Description'의 향수를 찾는 함수
    def most_similar_description(self, user_description, df, num_result=5) :
        similarity = df['Description'].apply(lambda x: self.description_similarity(user_description, x))
        df['Similarity'] = similarity
        #most_similar_des = df.nlargest(num_result, 'Similarity')['Description'].tolist()
        most_similar_des = df.nlargest(num_result, 'Similarity')[['Name', 'Similarity']]
        #'Name'과 'Similarity' 컬럼 포함
        return most_similar_des
```



```python
    #사용자에게 'description_input' 입력받기
    def description_perfume(self):
        description = []
        description_input = input("Please enter the Description, separating words with spaces. : ")
        description.append(description_input)
        while True :
            more_description = input("Would you like to enter more? [y/n]")
            if more_description == 'y' :
                description_input = input("Please enter the Description, separating words with spaces. : ")
                description.append(description_input)
                continue
            elif more_description == "n" :
                #입력된 모든 노트를 하나의 문자열로 합치기
                all_description = ', '.join(description)
                most_similar_des = self.most_similar_description(all_description, self.perfume_data, 1)
                #유사한 향수 상위 1개 찾아서 출력
                #print("가장 유사한 향수:", most_similar_des[0])
                print("Most similar perfume : ", most_similar_des['Name'].iloc[0])  #'Name' 컬럼의 첫 번째 값 출력


                more_description = input("""
    Enter the number you want
    1. Save to Wishlist
    2. More recommendation
    3. Exiting the program
    """)
                if more_description == "1" :
                    wishlist_data = wishlist.load_wishlist_from_file()
                    wishlist.add_to_wishlist(wishlist_data, most_similar_des['Name'].iloc[0] )
                    wishlist.save_wishlist_to_file(wishlist_data)
                    print(f"The perfume {most_similar_des['Name'].iloc[0]} has been added to the wishlist.")
                    break
```



```python
                elif more_description == "2" :
                    additional_similar = self.most_similar_description(all_description, self.perfume_data, 5)
                    selected_similar_perfumes2 = additional_similar['Name'][1:].tolist()
                    print("Additional recommended perfumes : ", selected_similar_perfumes2)
                    description_more_input = input("""
    Enter the number you want
    1. Save to Wishlist
    2. Exiting the program
    """)
```



```python
                if description_more_input == "1" :
                    wishlist_data = wishlist.load_wishlist_from_file()
                    for perfume in selected_similar_perfumes2 :
                            wishlist.add_to_wishlist(wishlist_data, perfume)
                    wishlist.save_wishlist_to_file(wishlist_data)
                    print(f"The perfume {selected_similar_perfumes2} has been added to the wishlist.")
                    break
                elif description_more_input == "2" :
                    print("Exiting the program.")
                    break
            elif more_description == "3" : #프로그램 종료
                print("Exiting the program.")
                break
        else:
          print("Invalid input. Please try again. : ")
```

(6) Perfume Recommendation Based on the User's Mood – mood_perfumes_data_generated.py

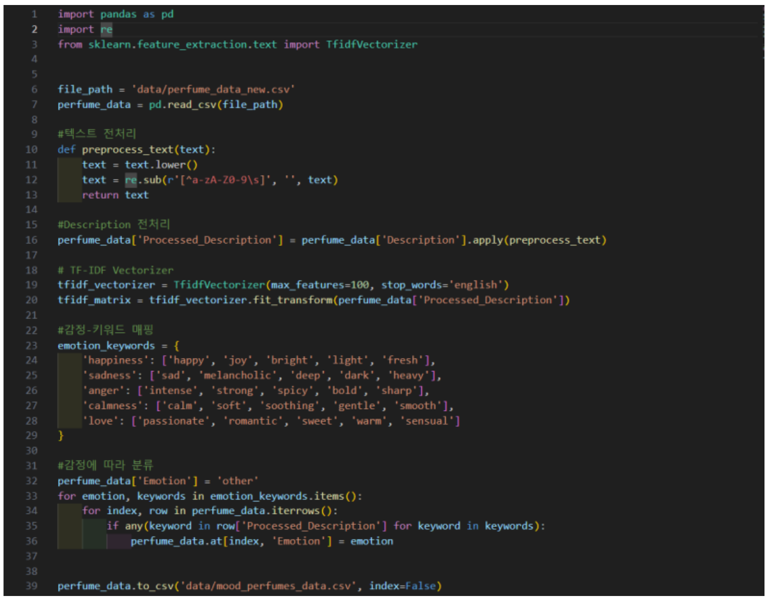
- Input : 'perfume_data_new.csv'

- Output : 'mood_perfumes_data.csv'

- Explanation : In order to recommend perfumes corresponding to emotions entered by the user, we have classified perfumes according to emotions.

- Applied Concepts : Mapping suitable keywords for each emotion and classifying perfumes according to the emotion involves functions, file input/output, dictionaries, and loops.

- Code screenshot



```python
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer


file_path = 'data/perfume_data_new.csv'
perfume_data = pd.read_csv(file_path)

#텍스트 전처리
def preprocess_text(text):
    text = text.lower()
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    return text

#Description 전처리
perfume_data['Processed_Description'] = perfume_data['Description'].apply(preprocess_text)

# TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(max_features=100, stop_words='english')
tfidf_matrix = tfidf_vectorizer.fit_transform(perfume_data['Processed_Description'])

#감정-키워드 매핑
emotion_keywords = {
    'happiness': ['happy', 'joy', 'bright', 'light', 'fresh'],
    'sadness': ['sad', 'melancholic', 'deep', 'dark', 'heavy'],
    'anger': ['intense', 'strong', 'spicy', 'bold', 'sharp'],
    'calmness': ['calm', 'soft', 'soothing', 'gentle', 'smooth'],
    'love': ['passionate', 'romantic', 'sweet', 'warm', 'sensual']
}

#감정에 따라 분류
perfume_data['Emotion'] = 'other'
for emotion, keywords in emotion_keywords.items():
    for index, row in perfume_data.iterrows():
        if any(keyword in row['Processed_Description'] for keyword in keywords):
            perfume_data.at[index, 'Emotion'] = emotion


perfume_data.to_csv('data/mood_perfumes_data.csv', index=False)
```

(7) Perfume Recommendation Based on the User's Mood – mood_recommendation.py

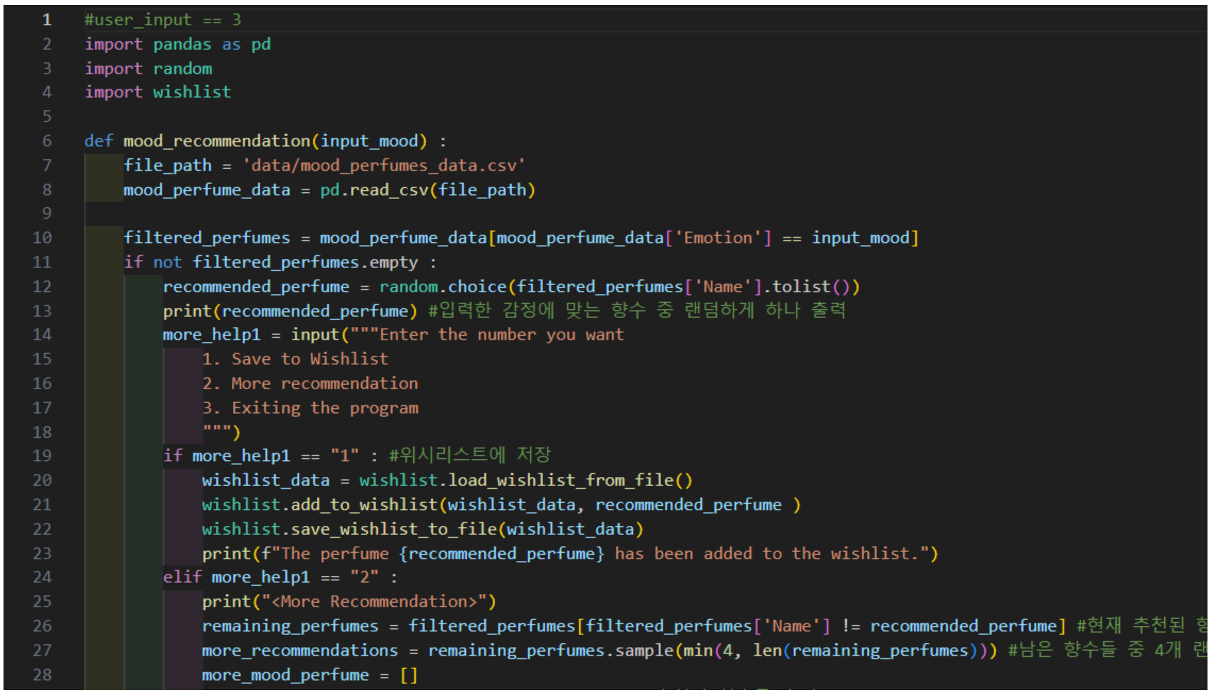
- Input : Emotion entered by the user

- Output : Display a perfume corresponding to the entered emotion

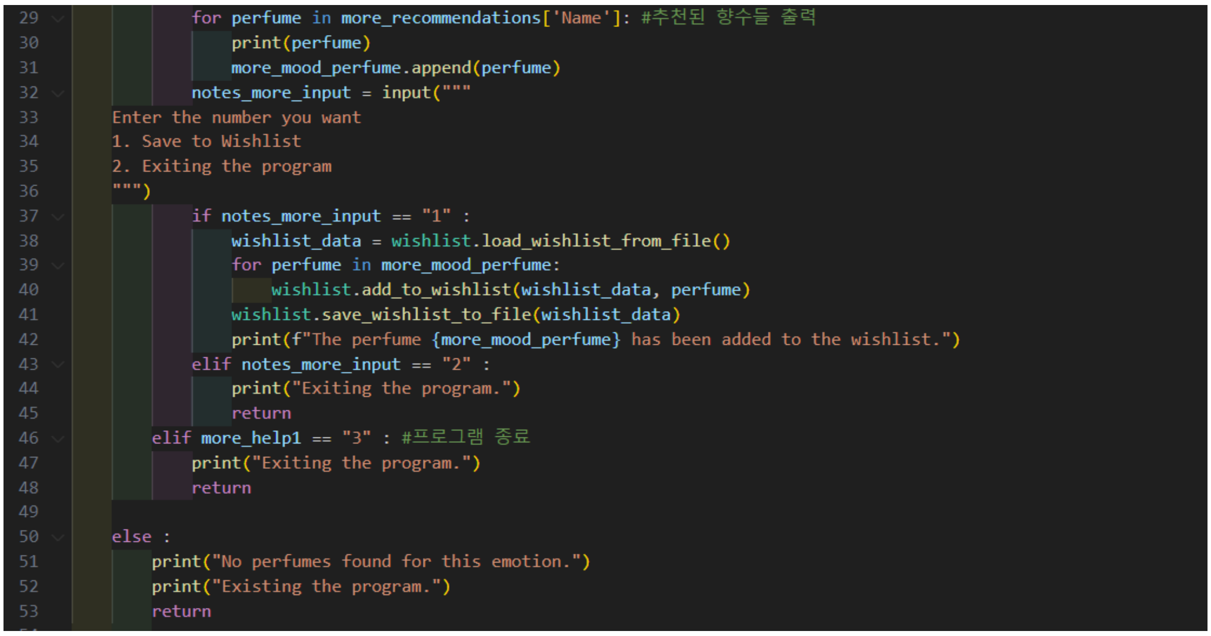
- Explanation : When the user inputs an emotion, one perfume corresponding to the entered emotion is randomly displayed. Afterwards, the user is asked whether they want to save the recommended perfume to the wishlist, receive more recommendations, or exit the program.

- Applied Concepts : File input/output, functions, conditional statements, loops, and modules are used in the process of loading data categorized by emotions and displaying a perfume corresponding to the emotion entered by the user.

- Code screenshot



```python
#user_input == 3
import pandas as pd
import random
import wishlist

def mood_recommendation(input_mood) :
    file_path = 'data/mood_perfumes_data.csv'
    mood_perfume_data = pd.read_csv(file_path)

    filtered_perfumes = mood_perfume_data[mood_perfume_data['Emotion'] == input_mood]
    if not filtered_perfumes.empty :
        recommended_perfume = random.choice(filtered_perfumes['Name'].tolist())
        print(recommended_perfume) #입력한 감정에 맞는 향수 중 랜덤하게 하나 출력
        more_help1 = input("""Enter the number you want
            1. Save to Wishlist
            2. More recommendation
            3. Exiting the program
            """)
        if more_help1 == "1" : #위시리스트에 저장
            wishlist_data = wishlist.load_wishlist_from_file()
            wishlist.add_to_wishlist(wishlist_data, recommended_perfume )
            wishlist.save_wishlist_to_file(wishlist_data)
            print(f"The perfume {recommended_perfume} has been added to the wishlist.")
        elif more_help1 == "2" :
            print("<More Recommendation>")
            remaining_perfumes = filtered_perfumes[filtered_perfumes['Name'] != recommended_perfume] #현재 추천된 향
            more_recommendations = remaining_perfumes.sample(min(4, len(remaining_perfumes))) #남은 향수들 중 4개 랜
            more_mood_perfume = []
```



```python
            for perfume in more_recommendations['Name']: #추천된 향수들 출력
                print(perfume)
                more_mood_perfume.append(perfume)
            notes_more_input = input("""
    Enter the number you want
    1. Save to Wishlist
    2. Exiting the program
    """)
            if notes_more_input == "1" :
                wishlist_data = wishlist.load_wishlist_from_file()
                for perfume in more_mood_perfume:
                    wishlist.add_to_wishlist(wishlist_data, perfume)
                wishlist.save_wishlist_to_file(wishlist_data)
                print(f"The perfume {more_mood_perfume} has been added to the wishlist.")
            elif notes_more_input == "2" :
                print("Exiting the program.")
                return
        elif more_help1 == "3" : #프로그램 종료
            print("Exiting the program.")
            return

    else :
        print("No perfumes found for this emotion.")
        print("Existing the program.")
        return
```

(8) Saving Recommended Perfumes to the Wishlist – wishlist.py

- Input : 'Function.add_to_wishlist' – wishlist, perfume_name

'Function.display_wishlist' – wishlist

'Function.save_wishlist_to_file' – wishist

'Function.delete_from_wishlist' – wishlist, number

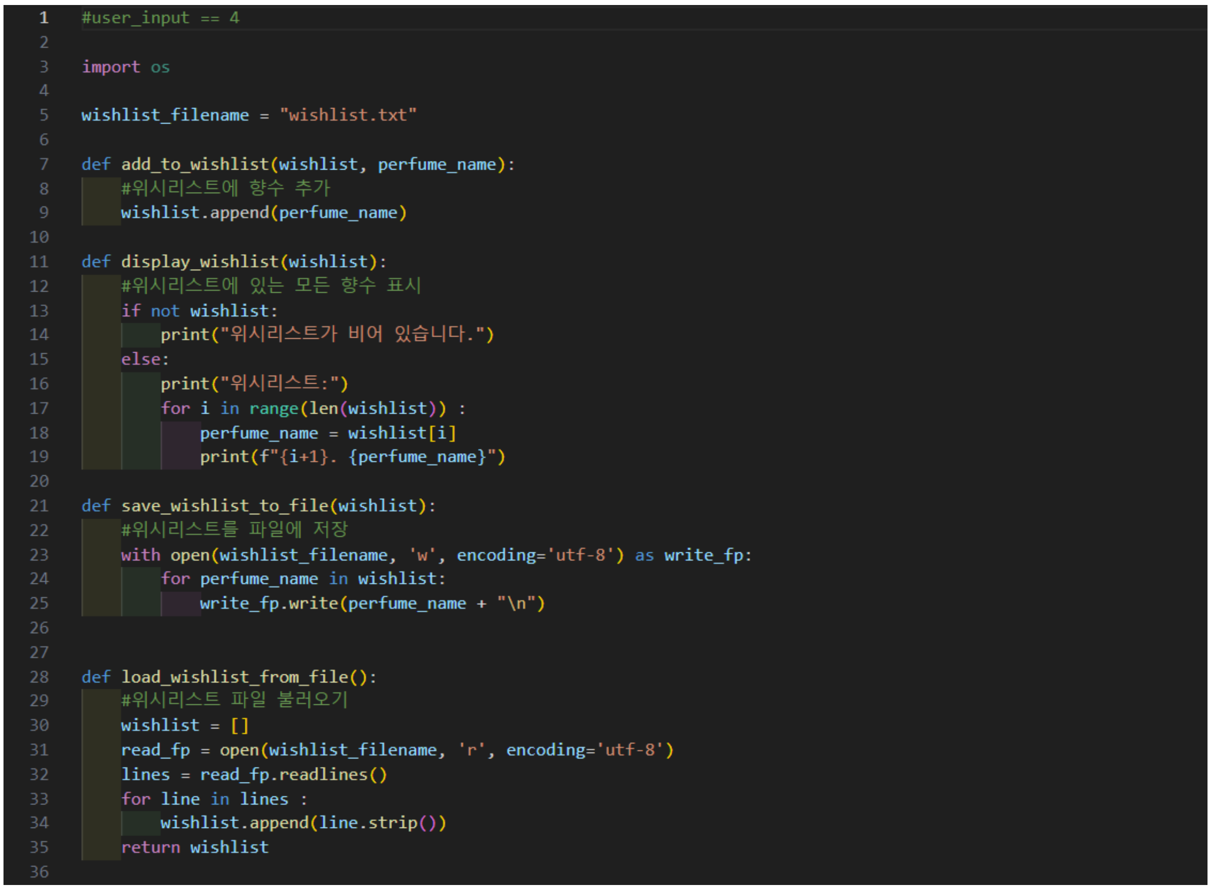
- Output : 'Function.display_wishlist' – wishlist

'Function.load_wishlist_from_file' - wishlist

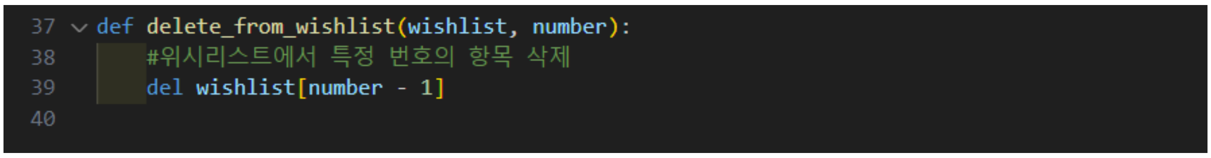
- Explanation : In main.py and other files, functions are created for each feature, allowing the necessary functionalities to be accessed when the file is called.

- Applied Concepts : Functions, loops, and file input/output are used for features such as adding items to the wishlist, deleting items, and displaying the entire content.

- Code screenshot



```python
#user_input == 4

import os

wishlist_filename = "wishlist.txt"

def add_to_wishlist(wishlist, perfume_name):
    #위시리스트에 향수 추가
    wishlist.append(perfume_name)

def display_wishlist(wishlist):
    #위시리스트에 있는 모든 향수 표시
    if not wishlist:
        print("위시리스트가 비어 있습니다.")
    else:
        print("위시리스트:")
        for i in range(len(wishlist)) :
            perfume_name = wishlist[i]
            print(f"{i+1}. {perfume_name}")

def save_wishlist_to_file(wishlist):
    #위시리스트를 파일에 저장
    with open(wishlist_filename, 'w', encoding='utf-8') as write_fp:
        for perfume_name in wishlist:
            write_fp.write(perfume_name + "\n")


def load_wishlist_from_file():
    #위시리스트 파일 불러오기
    wishlist = []
    read_fp = open(wishlist_filename, 'r', encoding='utf-8')
    lines = read_fp.readlines()
    for line in lines :
        wishlist.append(line.strip())
    return wishlist

```



```python
def delete_from_wishlist(wishlist, number):
    #위시리스트에서 특정 번호의 항목 삭제
    del wishlist[number - 1]

```

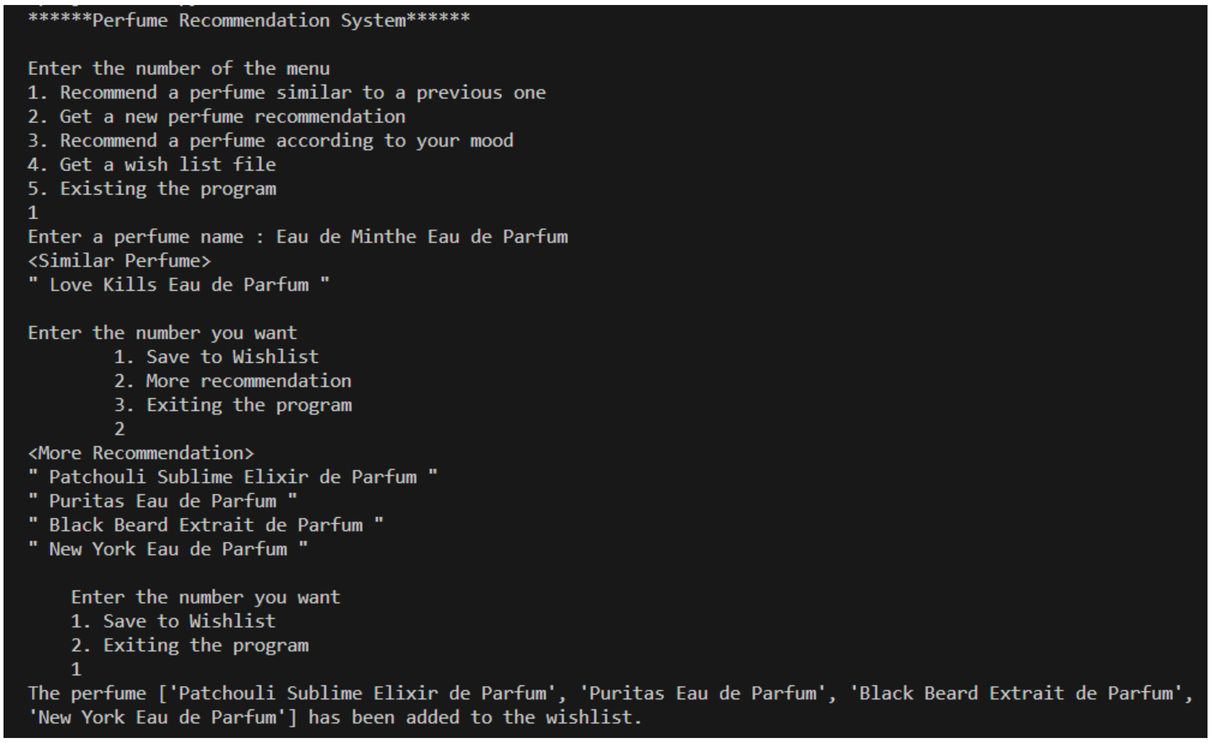
4. Test Result

(1) Recommendation of Perfumes Similar to Previously Used Ones

- A user-entered perfume exists in the dataset

- Explanation : When the user selects “1. Recommend a perfume similar to a previous one” from the initial menu and the entered perfume exists in the dataset, a similar perfume is displayed. The user is then asked whether they want to save it to the wishlist, get recommendations for other perfumes, or exit the program. This test shows the result of receiving more recommendations and saving them to the wishlist.

- Test result screenshot



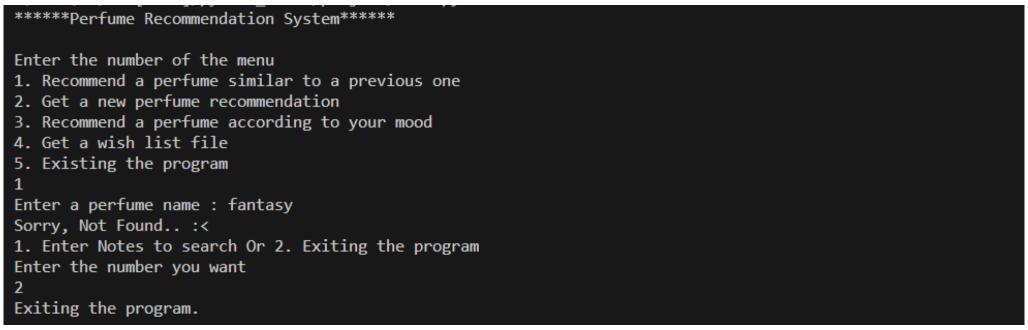
```
******Perfume Recommendation System******

Enter the number of the menu
1. Recommend a perfume similar to a previous one
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
1
Enter a perfume name : Eau de Minthe Eau de Parfum
<Similar Perfume>
" Love Kills Eau de Parfum "

Enter the number you want
        1. Save to Wishlist
        2. More recommendation
        3. Exiting the program
        2
<More Recommendation>
" Patchouli Sublime Elixir de Parfum "
" Puritas Eau de Parfum "
" Black Beard Extrait de Parfum "
" New York Eau de Parfum "

    Enter the number you want
    1. Save to Wishlist
    2. Exiting the program
    1
The perfume ['Patchouli Sublime Elixir de Parfum', 'Puritas Eau de Parfum', 'Black Beard Extrait de Parfum',
'New York Eau de Parfum'] has been added to the wishlist.
```

(2) Recommendation of Perfumes Similar to Previously Used Ones

- A user-entered perfume does not exist in the dataset

- Explanation : If the user selects “1. Recommend a perfume similar to a previous one” from the initial menu and the entered perfume does not exist in the dataset, they are asked whether they want to receive recommendations in a different way or exit the program. This test shows the result of exiting the program.

- Test result screenshot

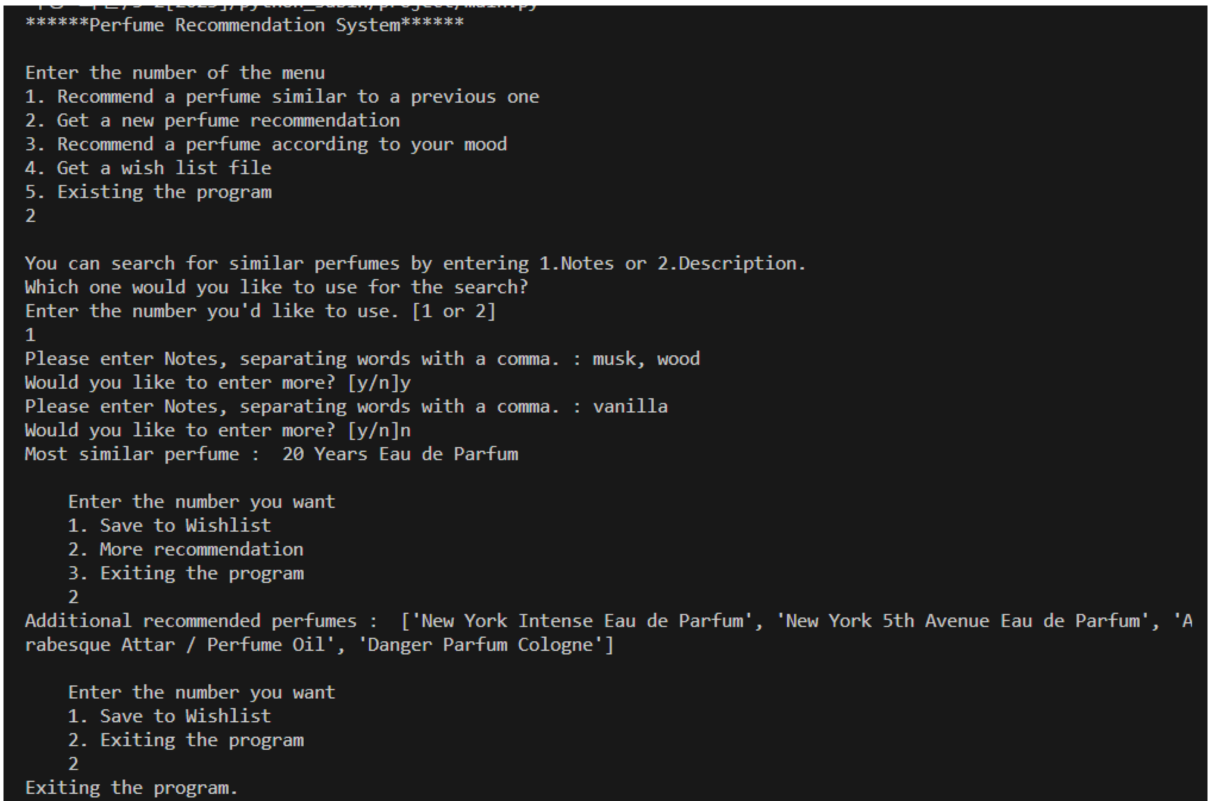


```
******Perfume Recommendation System******

Enter the number of the menu
1. Recommend a perfume similar to a previous one
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
1
Enter a perfume name : fantasy
Sorry, Not Found.. :<
1. Enter Notes to search Or 2. Exiting the program
Enter the number you want
2
Exiting the program.
```

(3) Recommendation of New Perfumes Through Notes Input

- Explanation : When the user selects “2. Get a new perfume recommendation” from the initial menu and chooses to enter Notes, the text entered is compared for similarity with the Notes column in the dataset, and the most similar perfume is displayed. The user is then asked whether they want to save it to the wishlist, get recommendations for other perfumes, or exit the program. This test shows the result of receiving more recommendations and then exiting the program.

- Test result screenshot

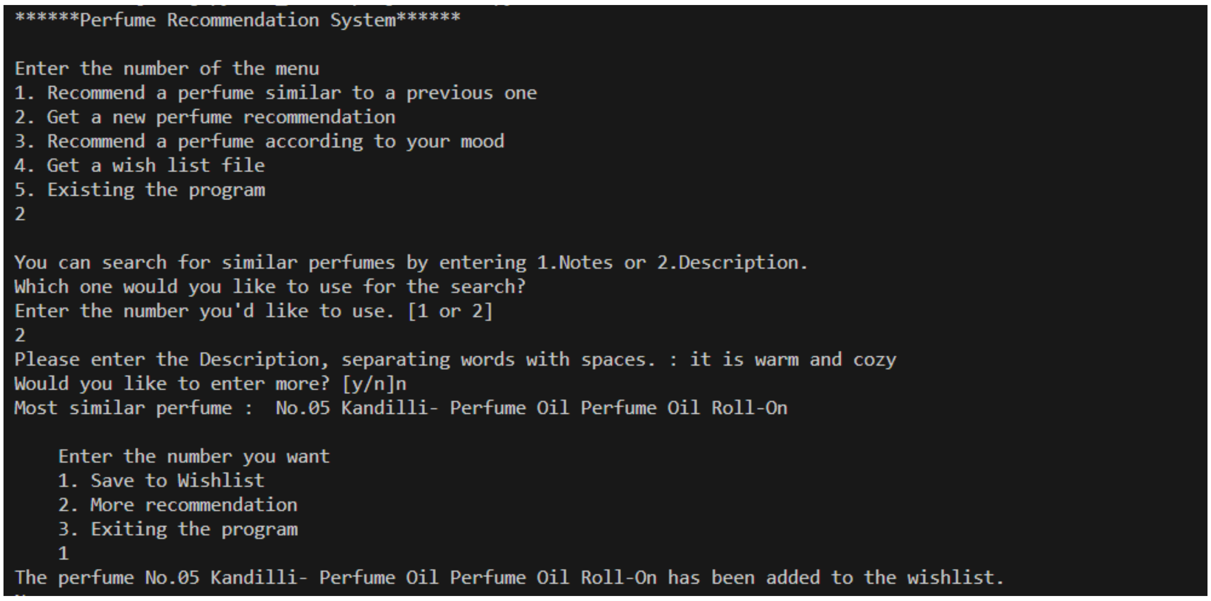


```
******Perfume Recommendation System******

Enter the number of the menu
1. Recommend a perfume similar to a previous one
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
2

You can search for similar perfumes by entering 1.Notes or 2.Description.
Which one would you like to use for the search?
Enter the number you'd like to use. [1 or 2]
1
Please enter Notes, separating words with a comma. : musk, wood
Would you like to enter more? [y/n]y
Please enter Notes, separating words with a comma. : vanilla
Would you like to enter more? [y/n]n
Most similar perfume :  20 Years Eau de Parfum

    Enter the number you want
    1. Save to Wishlist
    2. More recommendation
    3. Exiting the program
    2
Additional recommended perfumes :  ['New York Intense Eau de Parfum', 'New York 5th Avenue Eau de Parfum', 'A
rabesque Attar / Perfume Oil', 'Danger Parfum Cologne']

    Enter the number you want
    1. Save to Wishlist
    2. Exiting the program
    2
Exiting the program.
```

(4) Recommendation of New Perfumes Through Description Input

- Explanation : If the user selects "2. Get a new perfume recommendation" from the initial menu and chooses to enter a Description, the text entered is compared for similarity with the Description column in the dataset, and the most similar perfume is displayed. The user is then asked whether they want to save it to the wishlist, get recommendations for other perfumes, or exit the program. This test shows the result of receiving one perfume recommendation, saving it to the wishlist, and then exiting the program.

- Test result screenshot

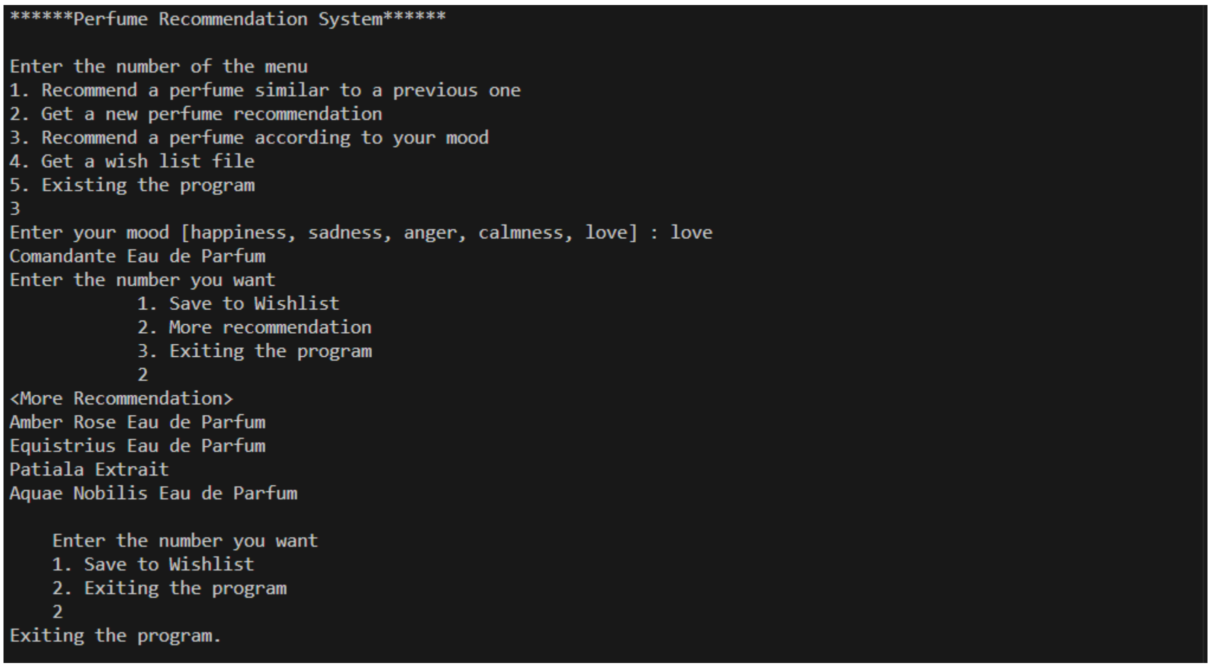


```
******Perfume Recommendation System******

Enter the number of the menu
1. Recommend a perfume similar to a previous one
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
2

You can search for similar perfumes by entering 1.Notes or 2.Description.
Which one would you like to use for the search?
Enter the number you'd like to use. [1 or 2]
2
Please enter the Description, separating words with spaces. : it is warm and cozy
Would you like to enter more? [y/n]n
Most similar perfume :  No.05 Kandilli- Perfume Oil Perfume Oil Roll-On

    Enter the number you want
    1. Save to Wishlist
    2. More recommendation
    3. Exiting the program
    1
The perfume No.05 Kandilli- Perfume Oil Perfume Oil Roll-On has been added to the wishlist.
```

(5) Perfume Recommendation Based on the User's Mood

- Explanation : When the user selects "3. Recommend a perfume according to your mood" from the initial menu and enters one of the five provided emotions, a perfume matching that emotion is displayed. The user is then asked whether they want to save it to the wishlist, get recommendations for other perfumes, or exit the program. This test shows the result of receiving more recommendations and then exiting the program.

- Test result screenshot

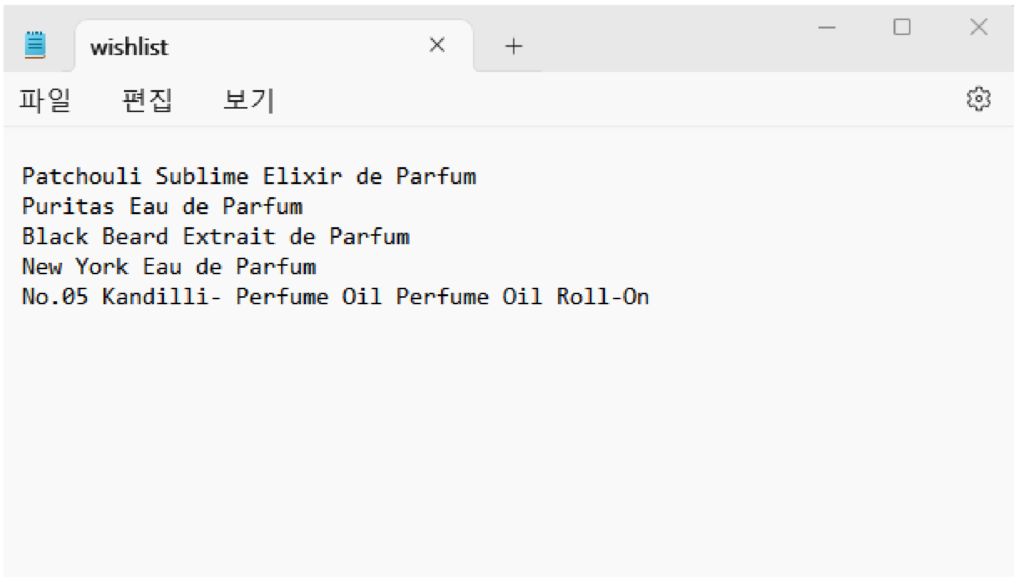


```
******Perfume Recommendation System******

Enter the number of the menu
1. Recommend a perfume similar to a previous one
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
3
Enter your mood [happiness, sadness, anger, calmness, love] : love
Comandante Eau de Parfum
Enter the number you want
            1. Save to Wishlist
            2. More recommendation
            3. Exiting the program
            2
<More Recommendation>
Amber Rose Eau de Parfum
Equistrius Eau de Parfum
Patiala Extrait
Aquae Nobilis Eau de Parfum

    Enter the number you want
    1. Save to Wishlist
    2. Exiting the program
    2
Exiting the program.
```

(6) Saving Recommended Perfumes to the Wishlist – other file

- Explanation : The initial menu options 1, 2, and 3 are used to get perfume recommendations and the test demonstrates saving similar perfumes to the wishlist. The results of the test cases (1) and (4) are shown.

- Test result screenshot

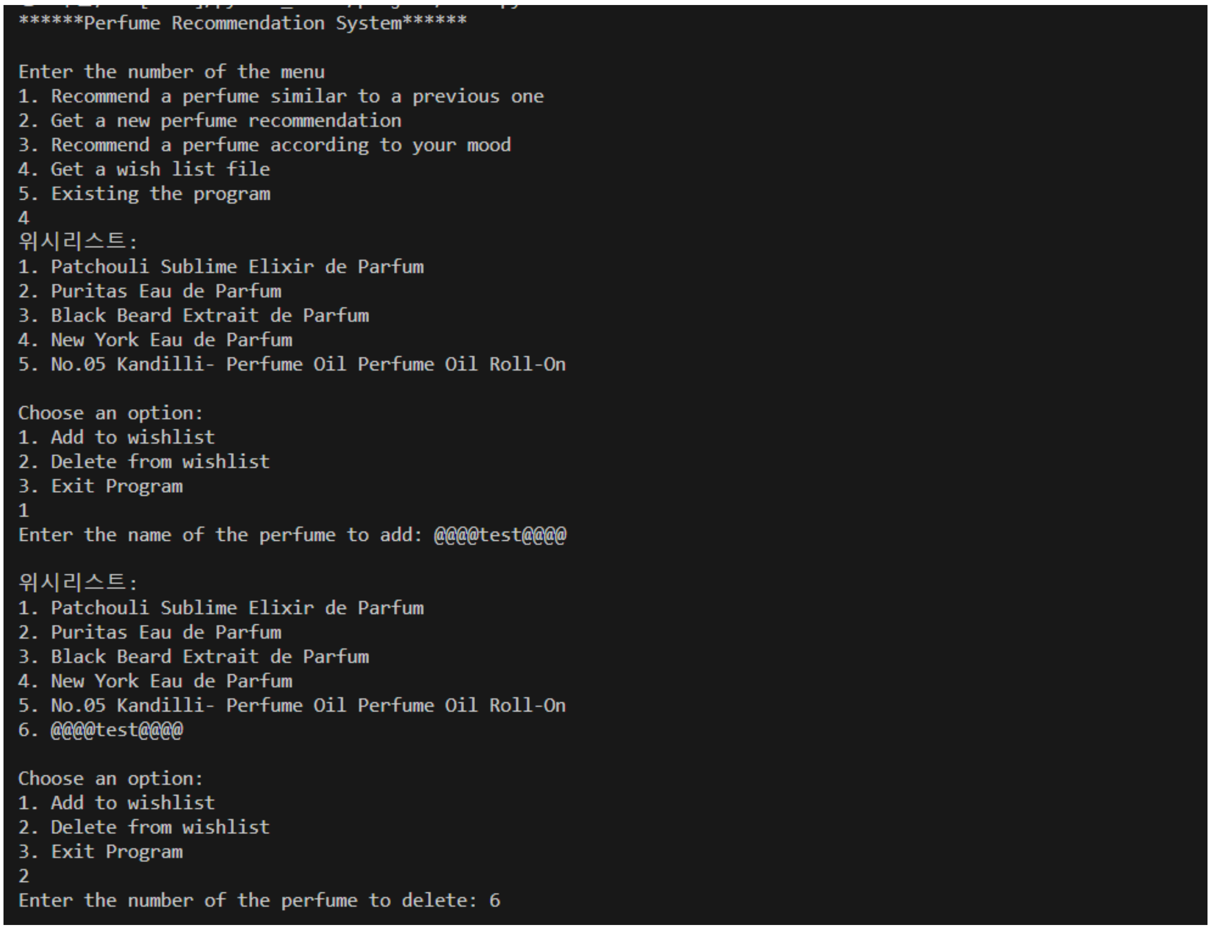


```
Patchouli Sublime Elixir de Parfum
Puritas Eau de Parfum
Black Beard Extrait de Parfum
New York Eau de Parfum
No.05 Kandilli- Perfume Oil Perfume Oil Roll-On
```

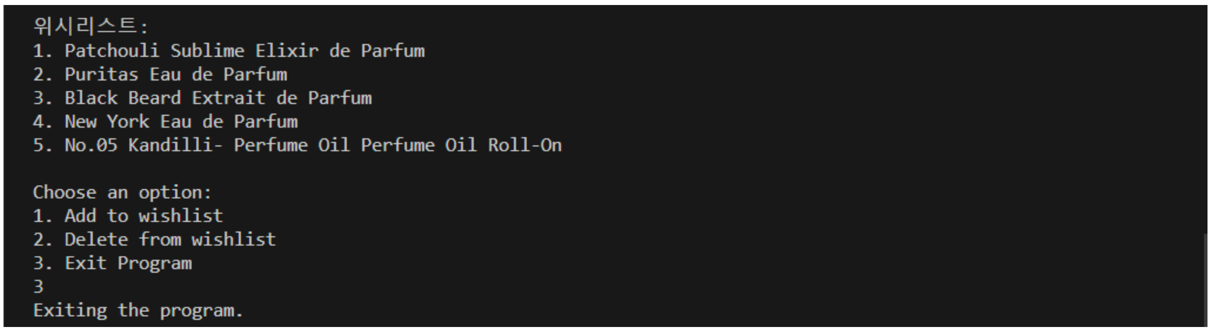
(7) Saving Recommended Perfumes to the Wishlist – main.py

- Explanation : If the user selects “4. Get a wish list file” from the initial menu, the current wishlist is displayed. The user is then asked whether they want to add or delete items, or exit the program. This test shows the result of adding an item to the wishlist and then deleting it.

- Test result screenshot



```
******Perfume Recommendation System******

Enter the number of the menu
1. Recommend a perfume similar to a previous one
2. Get a new perfume recommendation
3. Recommend a perfume according to your mood
4. Get a wish list file
5. Existing the program
4
위시리스트:
1. Patchouli Sublime Elixir de Parfum
2. Puritas Eau de Parfum
3. Black Beard Extrait de Parfum
4. New York Eau de Parfum
5. No.05 Kandilli- Perfume Oil Perfume Oil Roll-On

Choose an option:
1. Add to wishlist
2. Delete from wishlist
3. Exit Program
1
Enter the name of the perfume to add: @@@@test@@@@

위시리스트:
1. Patchouli Sublime Elixir de Parfum
2. Puritas Eau de Parfum
3. Black Beard Extrait de Parfum
4. New York Eau de Parfum
5. No.05 Kandilli- Perfume Oil Perfume Oil Roll-On
6. @@@@test@@@@

Choose an option:
1. Add to wishlist
2. Delete from wishlist
3. Exit Program
2
Enter the number of the perfume to delete: 6
```



```
위시리스트:
1. Patchouli Sublime Elixir de Parfum
2. Puritas Eau de Parfum
3. Black Beard Extrait de Parfum
4. New York Eau de Parfum
5. No.05 Kandilli- Perfume Oil Perfume Oil Roll-On

Choose an option:
1. Add to wishlist
2. Delete from wishlist
3. Exit Program
3
Exiting the program.
```

5. Changes in Comparison to the Plan

1) Feature Addition and Detailed Reorganization

- Before : In the proposal, the functions were listed as "Identifying Similar Fragrances" and "Adding Perfume to Wishlist".

- After : Added the "Perfume Recommendation Based on the User's Mood" feature. The "Identifying Similar Fragrances" feature was further detailed into subcategories like recommending perfumes similar to previously used ones and recommendations based on Notes/Description.

- Reason : This new feature is designed to be more user-friendly for customers who may find it difficult to input their desired fragrance due to a lack of familiarity with perfumes. Unlike the existing recommendation features, this new addition aims to enhance perfume sales by reaching out to customers who are not yet well-acquainted with fragrances. It is also anticipated to provide a novel and enjoyable experience for existing customers.

In addition, we expect that by rearranging the perfume recommendation function in more detail, users will be able to find the function they want more intuitively and quickly.

6. Lessons Learned & Feedback

-Reflections and Feedback on the Course

This semester's course left me with four main impressions

1. I personally appreciated learning not only the basic Python syntax but also about file input/output, web scraping, and machine learning. It was a content-rich course, making me feel that enrolling in it was the best decision of the semester. However, I did find the course content to be quite extensive. Unlike previous practical courses where the day's learning could be fully absorbed during the class, this course presented a larger volume of material and practical exercises that were more challenging than the taught content, which I found a bit overwhelming. Yet, this was

both a drawback and a strength of the course. It was challenging to keep up, but as the professor said, it was beneficial to work on applications beyond simple problems like pyramid-shaped star printing. For instance, I had a vague idea about what web scraping was and how it's done, but I never had the chance to try it until this course, which made it a fun learning experience. Although I mentioned it was challenging, I feel I gained a lot from the course. If you continue to teach this course next year, I believe covering the same amount of material as this semester would satisfy most students.

2. Another aspect I enjoyed was using Git. I had learned about Git in another open-source related course, but it was a brief introduction without practical application. Personally, I had been wanting to use Git, but never got around to it. So, being able to familiarize myself with basic Git functions like commit and clone during the course was great.

3. I really, really appreciated the absence of a team project.

4. It was also good that the midterm exam was done on a computer rather than by hand-writing code.

Thank you for your hard work this semester!