Python Programing

# Emotional therapist

Final Report

Date : 12/24

Name : KimSeHee

ID : 233973

# 1. Introduction

## 1) Background

In these times of grayness, many people live without understanding their own emotions and feelings. If they continue without realizing their emotions, it could lead to significant issues for themselves in the future. To address this, I believe an emotional management program is necessary

## 2) Project goal

Analyzing the user's diary to rank emotions Aim to extract the first and second highest emotions and suggest solutions

## 3) Differences from existing programs

The difference is that it directly analyzes the user's emotions and outputs them as graph images, and the user's emotions can be used by accumulating information every week. In addition, you can keep a journal yourself, not just analyze your emotions.

# 2. Functional Requirement

## 1) Retrieve User Diary Functionality

- Description: After receiving diary entries from the user for each day of the week (Monday through Sunday), the "load_diary_from_file" function is used to retrieve the diary text. Subsequently, the "read_diary_word" function is employed to separate the text into words and return them as a list.

### 2) Ability to visualize user diary weekly sentiment graphs and output them to a file

- Description: The user's weekly sentiment score is visualized through a graph and output to the program. You can also save the results by printing out a graph as an image file (png).

### 3) Providing solutions through weekly user sentiment analysis

- Description: Ranking the user's sentiment and returning it to the dictionary to provide a solution based on the overall sentiment score.

### 4) User diary writing feature

- Description: Ability to write and save a diary within the program instead of importing files

### 5) Playing music

- Description: Plays a song so that the user can relax while using the program.

## 3. Implementation

**(1) Ablilty to import user diary**

① Input

- Enter the file name of the user diary

② Output

- If the user file is not found, use the -> "Error: {filename} could not be found. Exit the program." Output

③ Description

- Prompt the user to input a total of 7 diaries for Monday through Sunday using the 'diary_input' function, which returns the diary texts. Use the 'load_diary_from_file' function to retrieve the user's diaries inputted by the 'diary_input' function, and display an error if the user's diaries do not exist. Extract words from the loaded diary text using the 'read_diary_word' function and return them as a list.

④ Applied Lessons Learned

- List: I utilized the 'days_of_week' variable to store a list of days of the week.I also created a variable named 'diaries' to store the user's diaries in a list.
- Built-in function

  ⑴ split: I utilized a function that uses the 'split' method to separate the diary text into words and return them as a list.

  ⑵ input: I gathered user diaries through the 'input' function.

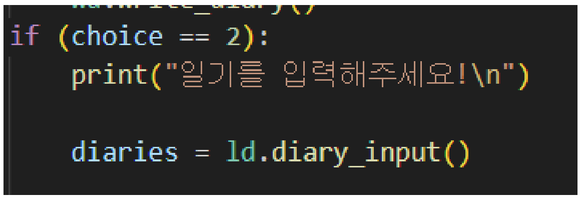
  ⑶ append: I added the user's diary to the list using the 'append' function.

- For loops: I iterated through the 'days_of_week' list using a for loop to collect user diaries for each day of the week.
- While loops: I enhanced stability by implementing an infinite loop, allowing the program to continue running regardless of the number of errors when the user inputs the diary name.
- Conditional Statements: I utilized a conditional statement to check if the diary was added to the list, and in such case, I used 'break' to exit the infinite loop.

- Function: I improved the code for better readability by utilizing the functions 'load_diary_from_file,' 'read_diary_word,' and 'diary_input.'
- Try-except: I used a 'try' block to attempt opening the file. If the file cannot be opened, various 'except' blocks are employed. If the file is not found, an error message is displayed, instructing the user to input the file name again. Additionally, if a different error occurs other than the file not being found, the program prints the specific error and exits.
- File: I read the user's diary file and returned the text as a string. Additionally, within the 'try' block, I simplified the function using the 'write' function.
- Module: I modularized the functions by separating the main file from the 'load_diary.py' file.

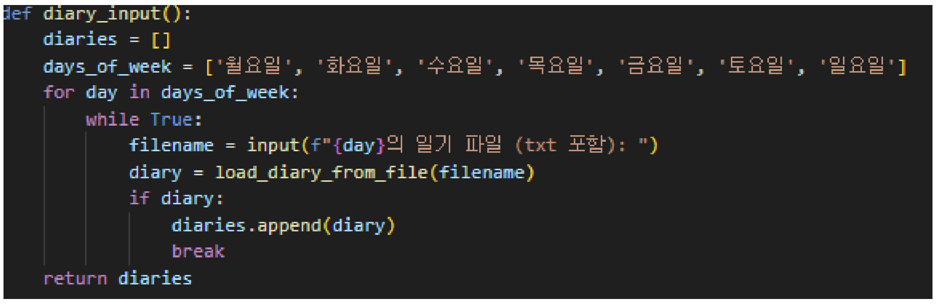
⑤ Code screenshot

- Function calls in the main section of the code.

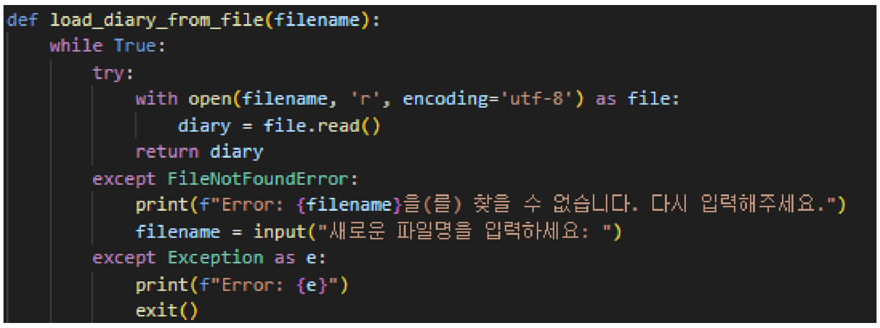


```
if (choice == 2):
    print("일기를 입력해주세요!\n")

    diaries = ld.diary_input()
```

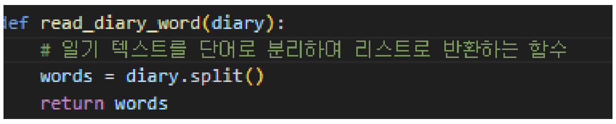
- The 'load_diary.py' file.



```
def diary_input():
    diaries = []
    days_of_week = ['월요일', '화요일', '수요일', '목요일', '금요일', '토요일', '일요일']
    for day in days_of_week:
        while True:
            filename = input(f"{day}의 일기 파일 (txt 포함): ")
            diary = load_diary_from_file(filename)
            if diary:
                diaries.append(diary)
                break
    return diaries
```



```python
def load_diary_from_file(filename):
    while True:
        try:
            with open(filename, 'r', encoding='utf-8') as file:
                diary = file.read()
            return diary
        except FileNotFoundError:
            print(f"Error: {filename}을(를) 찾을 수 없습니다. 다시 입력해주세요.")
            filename = input("새로운 파일명을 입력하세요: ")
        except Exception as e:
            print(f"Error: {e}")
            exit()
```



```python
lef read_diary_word(diary):
    # 일기 텍스트를 단어로 분리하여 리스트로 반환하는 함수
    words = diary.split()
    return words
```

**(2) Ability to visualize user diary weekly sentiment graphs and output them to a file**

① Input

- The text where the user's diary is divided word by word, The user inputs the filename for the graph image.

② Output:

- I generate an image file depicting the analysis of the user's weekly emotions in a graph and output the calculated ranking of the user's weekly emotions.

③ Description

- I import the sentiment analysis from 'nlit.sentiment'. Then, I perform sentiment analysis on the user's text. After analyzing, I classify the emotions into four categories: negative, positive, mixed, and neutral. I assign numerical values to these categories. I visualize the list containing weekly emotions using 'matplotlib.pyplot'. I set X as the days of the week and Y as the corresponding numerical values of emotions for each day. After that, I save the graph as an image file.

④ Applied Lessons Learned

- For Loops: I iterate through the list containing all the user's diaries one by one.
- Built-in-Function

  ① join: I concatenate the words in 'day_diary_words' into a single sentence using the 'join' function, and store the result in the 'day_diary_text' variable.

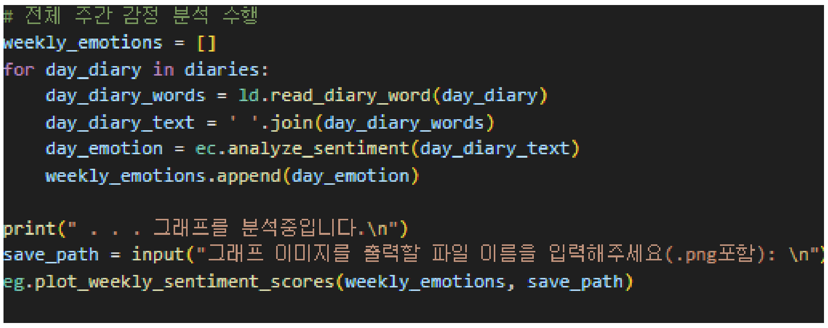
  ② append: I add the analyzed emotional information to the

  'weekly_emotions' list using the 'append' function.

  ③ input: I use the 'input' function to prompt the user to input the filename for the graph image.
- Machine learning: I use machine learning with 'nltk.sentiment' to perform sentiment analysis on the entered text.
- Text Mining: I utilize text mining techniques to tokenize the user's diary text into words and then perform sentiment analysis.
- List: I use a loop to perform sentiment analysis for each day of the week, and store the results of the analysis in the 'weekly_emotions' list.
- Pyplot: I import the 'matplotlib.pyplot' module and visualize the weekly emotion scores as a graph.
- Try-except: I use a 'try' block to attempt saving the graph as an image file, and if an error occurs, I print an error message.
- Module: I modularize the code by creating the 'emotion_graph.py' file for better readability.
- Function: I organize the code for better readability by using functions such as 'analyze_sentiment', 'plot_weekly_sentiment_scores', and 'save_graph'.

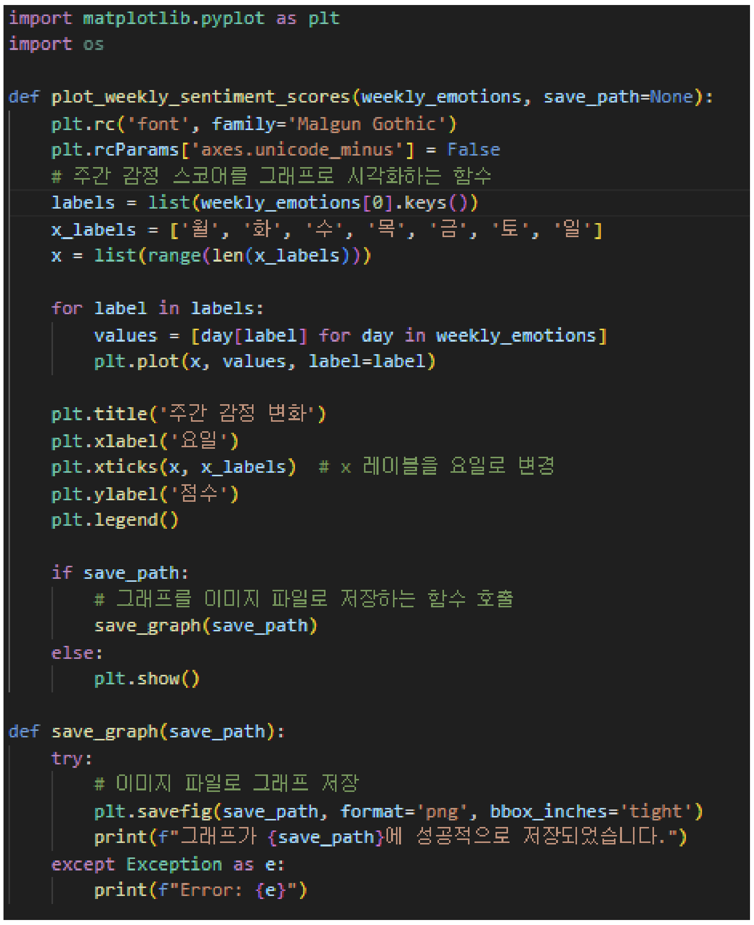
⑤ Code Screenshot

- The part of the main file where functions are called.



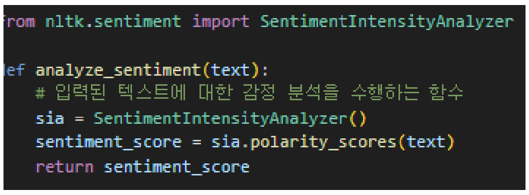
```
# 전체 주간 감정 분석 수행
weekly_emotions = []
for day_diary in diaries:
    day_diary_words = ld.read_diary_word(day_diary)
    day_diary_text = ' '.join(day_diary_words)
    day_emotion = ec.analyze_sentiment(day_diary_text)
    weekly_emotions.append(day_emotion)

print(" . . . 그래프를 분석중입니다.\n")
save_path = input("그래프 이미지를 출력할 파일 이름을 입력해주세요(.png포함): \n"
eg.plot_weekly_sentiment_scores(weekly_emotions, save_path)
```

- The ‘emotion_graph.py’ File



```
import matplotlib.pyplot as plt
import os

def plot_weekly_sentiment_scores(weekly_emotions, save_path=None):
    plt.rc('font', family='Malgun Gothic')
    plt.rcParams['axes.unicode_minus'] = False
    # 주간 감정 스코어를 그래프로 시각화하는 함수
    labels = list(weekly_emotions[0].keys())
    x_labels = ['월', '화', '수', '목', '금', '토', '일']
    x = list(range(len(x_labels)))

    for label in labels:
        values = [day[label] for day in weekly_emotions]
        plt.plot(x, values, label=label)

    plt.title('주간 감정 변화')
    plt.xlabel('요일')
    plt.xticks(x, x_labels)  # x 레이블을 요일로 변경
    plt.ylabel('점수')
    plt.legend()

    if save_path:
        # 그래프를 이미지 파일로 저장하는 함수 호출
        save_graph(save_path)
    else:
        plt.show()

def save_graph(save_path):
    try:
        # 이미지 파일로 그래프 저장
        plt.savefig(save_path, format='png', bbox_inches='tight')
        print(f"그래프가 {save_path}에 성공적으로 저장되었습니다.")
    except Exception as e:
        print(f"Error: {e}")
```

- The 'emotion_score.py' file for performing sentiment analysis.



```
rom nltk.sentiment import SentimentIntensityAnalyzer

ef analyze_sentiment(text):
   # 입력된 텍스트에 대한 감정 분석을 수행하는 함수
   sia = SentimentIntensityAnalyzer()
   sentiment_score = sia.polarity_scores(text)
   return sentiment_score
```

**(3) Providing solutions through weekly user sentiment analysis**

① Input

- I receive the list containing the analysis of the user's emotions.

② Output

- I provide the user with solutions for the emotions that are ranked high, along with the numerical representation of the emotional ranking for the week.

③ Description

- I rank the user's weekly emotions using the 'rank_emotions' function and show the user the emotional ranking for the current week. After that, using the 'provide_solution' function, I offer solutions for the emotions with the highest rank for the week and organize the solutions into a file for output.

④ Applied Lessons Learned

- File: I output to a file the organized solutions for the top emotions and the emotions for the current week, which are presented to the user.
- Conditional Statements: I use 'if' statements to differentiate between positive, negative, or other top emotions, and accordingly, output solutions and display messages. Additionally, I enhance stability by adding an 'else' statement to handle cases where the file is not provided.
- List Array: I utilize array functionalities with lists to easily extract the top emotions.
- Dictionary: I simplify the organization of emotions and their numerical values using a dictionary. Additionally, I use the 'keys' function to extract a list of numerical 'keys' arranged in order.

- Built-in-Function

  ⑴ sorted: I arrange the emotion scores using the 'sorted' function and then rank them.

  ⑵ dict() : I return the emotion scores as a dictionary.

  ⑶ get: I use the 'get' function to obtain the 'value' of the corresponding key in the dictionary.

  ⑷ Input: The user inputs the filename for the emotion scores and solutions to be organized.

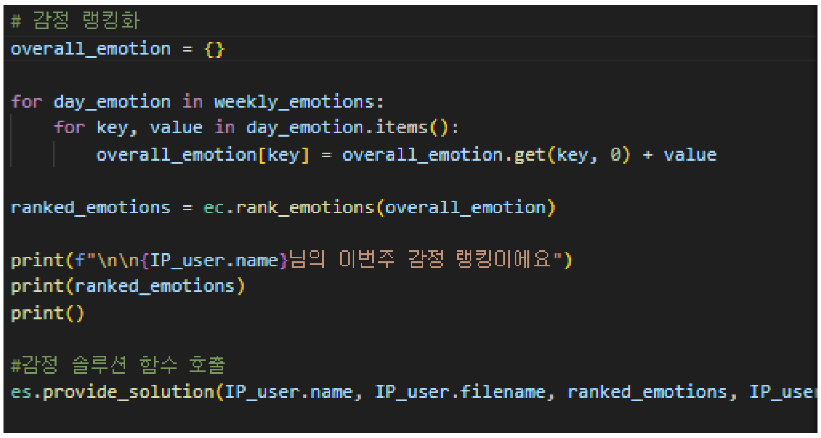
- For-Loops: I iterate through the 'weekly_emotions' list, convert it into a dictionary, and then rank it.

- Module: I separate the main file from 'emotion_solution.py' and 'emotion_score.py,' making the code more readable.

- Function: I organize the code more neatly by using the 'rank_emotions' and 'provide_solution' functions.

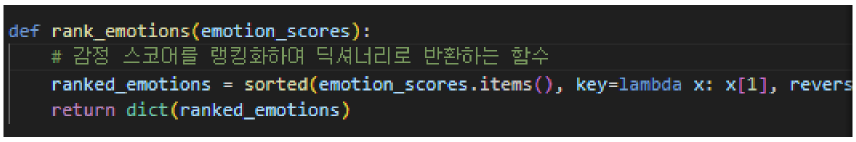
⑤ Code Screenshot

- The function calls in the main file.



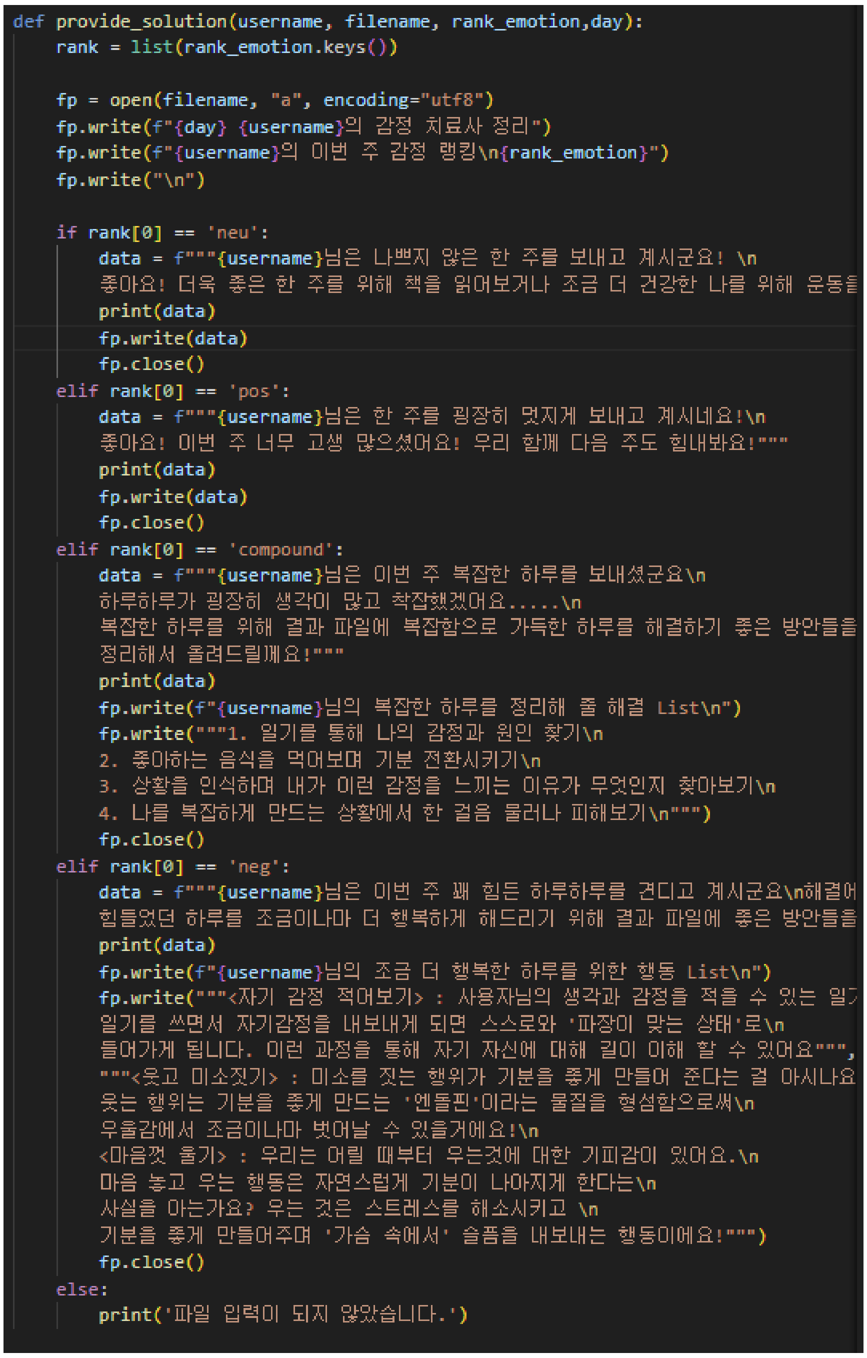
```
# 감정 랭킹화
overall_emotion = {}

for day_emotion in weekly_emotions:
    for key, value in day_emotion.items():
        overall_emotion[key] = overall_emotion.get(key, 0) + value

ranked_emotions = ec.rank_emotions(overall_emotion)

print(f"\n\n{IP_user.name}님의 이번주 감정 랭킹이에요")
print(ranked_emotions)
print()

#감정 솔루션 함수 호출
es.provide_solution(IP_user.name, IP_user.filename, ranked_emotions, IP_use
```

- rank the emotion scores and return them as a dictionary.



```python
def rank_emotions(emotion_scores):
    # 감정 스코어를 랭킹화하여 딕셔너리로 반환하는 함수
    ranked_emotions = sorted(emotion_scores.items(), key=lambda x: x[1], revers
    return dict(ranked_emotions)
```

- The file providing solutions.



```python
def provide_solution(username, filename, rank_emotion,day):
    rank = list(rank_emotion.keys())

    fp = open(filename, "a", encoding="utf8")
    fp.write(f"{day} {username}의 감정 치료사 정리")
    fp.write(f"{username}의 이번 주 감정 랭킹\n{rank_emotion}")
    fp.write("\n")

    if rank[0] == 'neu':
        data = f"""{username}님은 나쁘지 않은 한 주를 보내고 계시군요! \n
        좋아요! 더욱 좋은 한 주를 위해 책을 읽어보거나 조금 더 건강한 나를 위해 운동
        print(data)
        fp.write(data)
        fp.close()
    elif rank[0] == 'pos':
        data = f"""{username}님은 한 주를 굉장히 멋지게 보내고 계시네요!\n
        좋아요! 이번 주 너무 고생 많으셨어요! 우리 함께 다음 주도 힘내봐요!"""
        print(data)
        fp.write(data)
        fp.close()
    elif rank[0] == 'compound':
        data = f"""{username}님은 이번 주 복잡한 하루를 보내셨군요\n
        하루하루가 굉장히 생각이 많고 착잡했겠어요.....\n
        복잡한 하루를 위해 결과 파일에 복잡함으로 가득한 하루를 해결하기 좋은 방안들을
        정리해서 올려드릴께요!"""
        print(data)
        fp.write(f"{username}님의 복잡한 하루를 정리해 줄 해결 List\n")
        fp.write("""1. 일기를 통해 나의 감정과 원인 찾기\n
        2. 좋아하는 음식을 먹어보며 기분 전환시키기\n
        3. 상황을 인식하며 내가 이런 감정을 느끼는 이유가 무엇인지 찾아보기\n
        4. 나를 복잡하게 만드는 상황에서 한 걸음 물러나 피해보기\n""")
        fp.close()
    elif rank[0] == 'neg':
        data = f"""{username}님은 이번 주 꽤 힘든 하루하루를 견디고 계시군요\n해결에
        힘들었던 하루를 조금이나마 더 행복하게 해드리기 위해 결과 파일에 좋은 방안들을
        print(data)
        fp.write(f"{username}님의 조금 더 행복한 하루를 위한 행동 List\n")
        fp.write("""<자기 감정 적어보기> : 사용자님의 생각과 감정을 적을 수 있는 일
        일기를 쓰면서 자기감정을 내보내게 되면 스스로와 '파장이 맞는 상태'로\n
        들어가게 됩니다. 이런 과정을 통해 자기 자신에 대해 깊이 이해 할 수 있어요""",
        """<웃고 미소짓기> : 미소를 짓는 행위가 기분을 좋게 만들어 준다는 걸 아시나요
        웃는 행위는 기분을 좋게 만드는 '엔돌핀'이라는 물질을 형성함으로써\n
        우울감에서 조금이나마 벗어날 수 있을거에요!\n
        <마음껏 울기> : 우리는 어릴 때부터 우는것에 대한 기피감이 있어요.\n
        마음 놓고 우는 행동은 자연스럽게 기분이 나아지게 한다는\n
        사실을 아는가요? 우는 것은 스트레스를 해소시키고 \n
        기분을 좋게 만들어주며 '가슴 속에서' 슬픔을 내보내는 행동이에요!""")
        fp.close()
    else:
        print('파일 입력이 되지 않았습니다.')
```

**(4) User diary writing feature**

① Input

- I collect the user's diary and ask for the day of the week when the user writes it.

② Output

- The diary entered by the user is automatically saved as a txt file, If the user enters an incorrect day of the week, a prompt is displayed, asking them to enter it again.

③ Description

- The user writes a diary, and when they finish, they input '일기 끝' (end of the diary). Upon entering '일기 끝,' the diary writing process automatically concludes, and the user is prompted to input the current day of the week. For example, if the user enters '월요일' (Monday), the user's diary is automatically saved as 'monday_diary.txt.'

④ Applied Lessons Learned

- While Loops: Through an infinite loop in the 'write_diary' function, the user can continue entering diary entries until they input '일기 끝' (end of the diary).

- Conditional Statements: I use 'if' statements to set conditions based on the user's chosen day of the week (from Monday to Sunday). The program saves the diary to the corresponding file. Additionally, in the infinite loop, if the user inputs '일기 끝' (end of the diary), the loop is terminated.

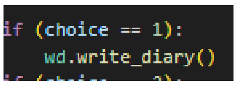
- File: The user writes a diary, and the program creates a file with the corresponding day of the week using the 'with' statement. It writes the diary content to the file and then closes the file.

- Module: I separate 'write_diary.py' from the main file, thereby modularizing the code.
- Function: I create the 'write_diary' function to make the code more readable.
- Built-in-Fucntion

  ① input: I gather both the diary entry and the day of the week from the user.

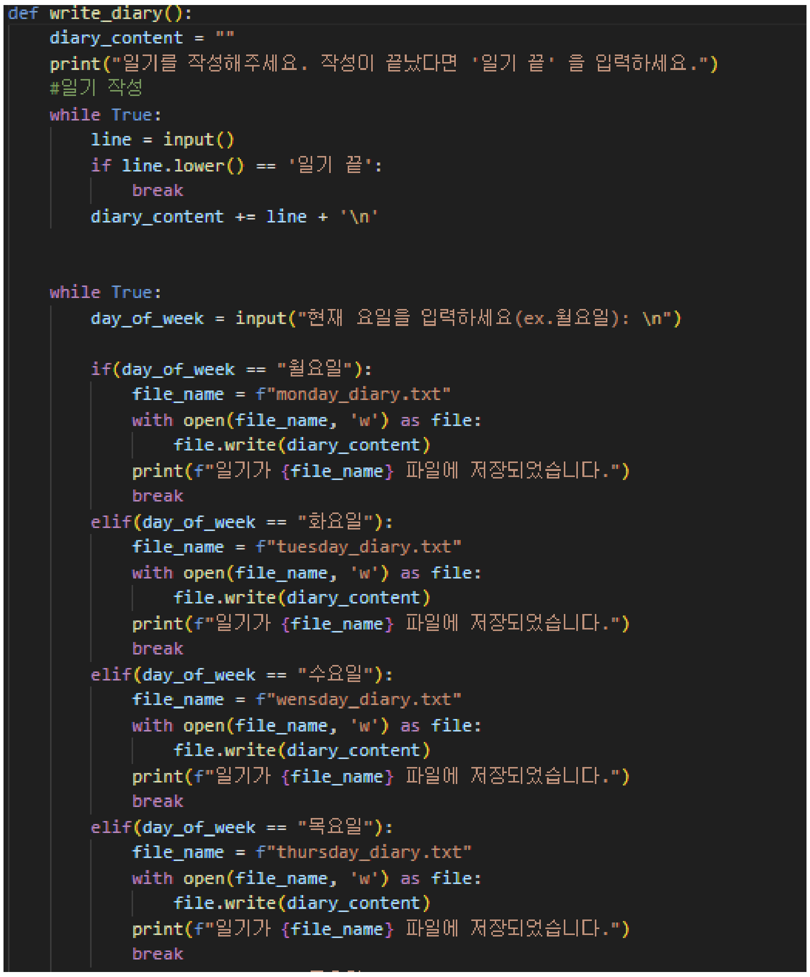
⑤ Code Screenshot

- The function calls in the main file.

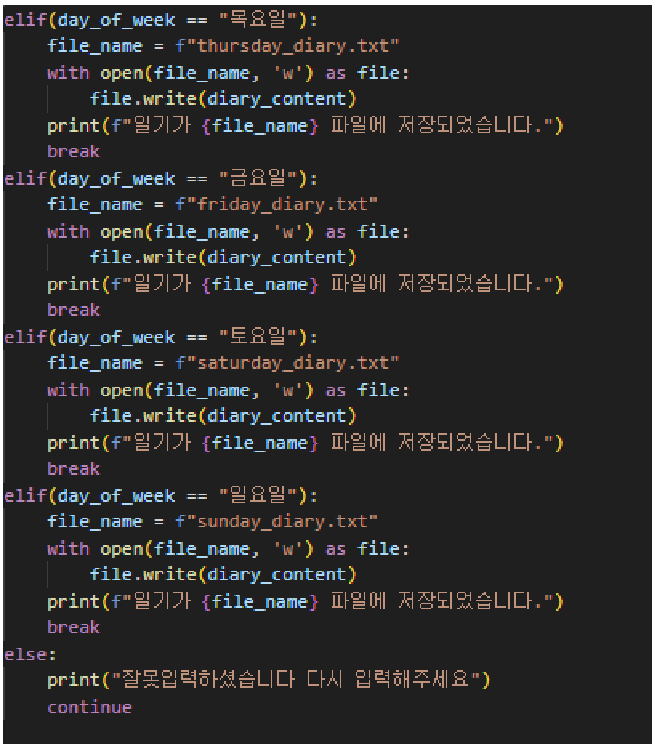


```python
if (choice == 1):
    wd.write_diary()
```

- The function for writing diary entries.



```python
def write_diary():
    diary_content = ""
    print("일기를 작성해주세요. 작성이 끝났다면 '일기 끝' 을 입력하세요.")
    #일기 작성
    while True:
        line = input()
        if line.lower() == '일기 끝':
            break
        diary_content += line + '\n'


    while True:
        day_of_week = input("현재 요일을 입력하세요(ex.월요일): \n")

        if(day_of_week == "월요일"):
            file_name = f"monday_diary.txt"
            with open(file_name, 'w') as file:
                file.write(diary_content)
            print(f"일기가 {file_name} 파일에 저장되었습니다.")
            break
        elif(day_of_week == "화요일"):
            file_name = f"tuesday_diary.txt"
            with open(file_name, 'w') as file:
                file.write(diary_content)
            print(f"일기가 {file_name} 파일에 저장되었습니다.")
            break
        elif(day_of_week == "수요일"):
            file_name = f"wensday_diary.txt"
            with open(file_name, 'w') as file:
                file.write(diary_content)
            print(f"일기가 {file_name} 파일에 저장되었습니다.")
            break
        elif(day_of_week == "목요일"):
            file_name = f"thursday_diary.txt"
            with open(file_name, 'w') as file:
                file.write(diary_content)
            print(f"일기가 {file_name} 파일에 저장되었습니다.")
            break
```



```python
elif(day_of_week == "목요일"):
    file_name = f"thursday_diary.txt"
    with open(file_name, 'w') as file:
        file.write(diary_content)
    print(f"일기가 {file_name} 파일에 저장되었습니다.")
    break
elif(day_of_week == "금요일"):
    file_name = f"friday_diary.txt"
    with open(file_name, 'w') as file:
        file.write(diary_content)
    print(f"일기가 {file_name} 파일에 저장되었습니다.")
    break
elif(day_of_week == "토요일"):
    file_name = f"saturday_diary.txt"
    with open(file_name, 'w') as file:
        file.write(diary_content)
    print(f"일기가 {file_name} 파일에 저장되었습니다.")
    break
elif(day_of_week == "일요일"):
    file_name = f"sunday_diary.txt"
    with open(file_name, 'w') as file:
        file.write(diary_content)
    print(f"일기가 {file_name} 파일에 저장되었습니다.")
    break
else:
    print("잘못입력하셨습니다 다시 입력해주세요")
    continue
```

## (5) Playing Music

① Input

- I ask the user whether they want to stop the music.

② Output

- I play the music.

③ Description

- I play music automatically when the program is executed. However, if the user wants to stop listening to the music, I terminate the music playback.

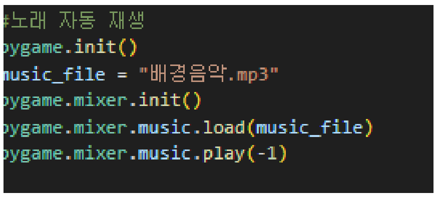
④ Applied Lessons Learned

- If-continue: In the conditional statement, if the user wishes to turn off the music, I use 'continue' to re-enter the infinite loop.

- Import: I import the 'pygame' module and use it to play music.

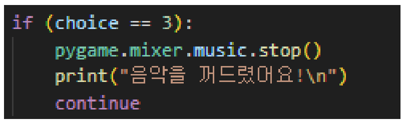
⑤ Code Screenshot

- The code for playing music.



```
#노래 자동 재생
pygame.init()
music_file = "배경음악.mp3"
pygame.mixer.init()
pygame.mixer.music.load(music_file)
pygame.mixer.music.play(-1)
```

- The code for stopping music playback.



```
if (choice == 3):
    pygame.mixer.music.stop()
    print("음악을 꺼드렸어요!\n")
    continue
```

## (6) Collecting other user information.

① Input

- I collect the user's name, file name, and current date from the user.

② Output

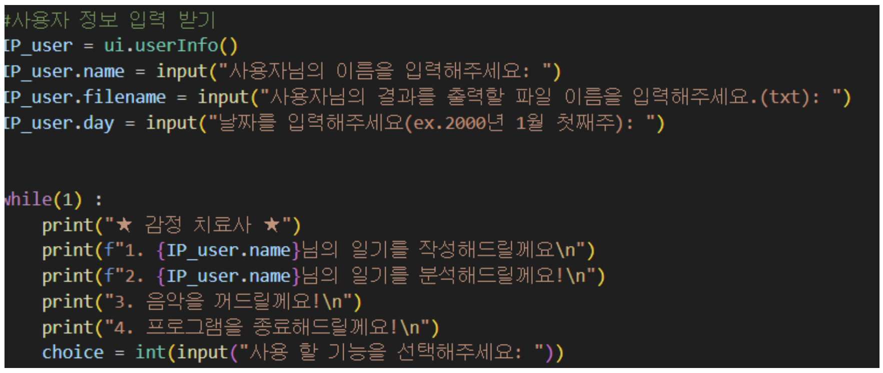
- The program prints the user's name and displays the file name as entered by the user. Additionally, the current date is included in the output file.

③ Applied Lessons Learned

- Class: I encapsulate the user input information in a class for more effective and organized use.

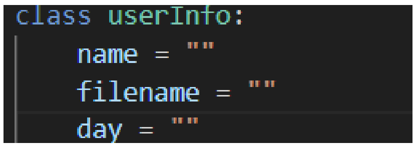
④ Code Screenshot

- Main File



```
#사용자 정보 입력 받기
IP_user = ui.userInfo()
IP_user.name = input("사용자님의 이름을 입력해주세요: ")
IP_user.filename = input("사용자님의 결과를 출력할 파일 이름을 입력해주세요.(txt): ")
IP_user.day = input("날짜를 입력해주세요(ex.2000년 1월 첫째주): ")


while(1) :
    print("★ 감정 치료사 ★")
    print(f"1. {IP_user.name}님의 일기를 작성해드릴께요\n")
    print(f"2. {IP_user.name}님의 일기를 분석해드릴께요!\n")
    print("3. 음악을 꺼드릴께요!\n")
    print("4. 프로그램을 종료해드릴께요!\n")
    choice = int(input("사용 할 기능을 선택해주세요: "))
```

- user_Info.py File



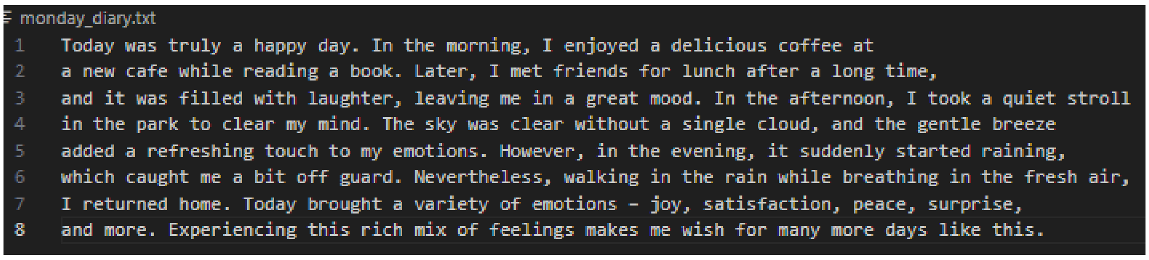
```
class userInfo:
    name = ""
    filename = ""
    day = ""
```

## 4. Test Result

### (1) Ability to import user diary

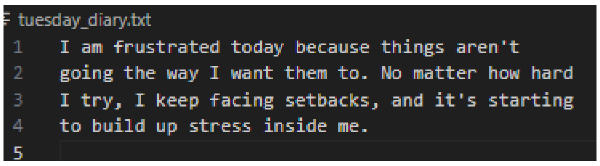
- Description: As an example of a user diary, the result is 7 inputs and the output to a dictionary appraisal list.

① 7 user diary examples



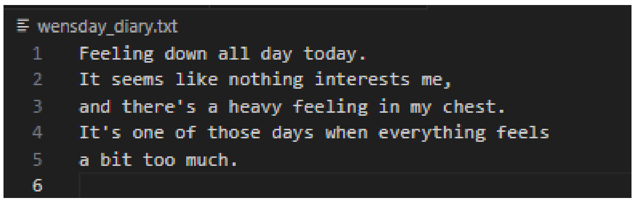
monday_diary.txt

```
Today was truly a happy day. In the morning, I enjoyed a delicious coffee at
a new cafe while reading a book. Later, I met friends for lunch after a long time,
and it was filled with laughter, leaving me in a great mood. In the afternoon, I took a quiet stroll
in the park to clear my mind. The sky was clear without a single cloud, and the gentle breeze
added a refreshing touch to my emotions. However, in the evening, it suddenly started raining,
which caught me a bit off guard. Nevertheless, walking in the rain while breathing in the fresh air,
I returned home. Today brought a variety of emotions - joy, satisfaction, peace, surprise,
and more. Experiencing this rich mix of feelings makes me wish for many more days like this.
```



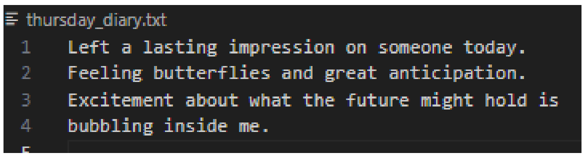
tuesday_diary.txt

```
I am frustrated today because things aren't
going the way I want them to. No matter how hard
I try, I keep facing setbacks, and it's starting
to build up stress inside me.

```

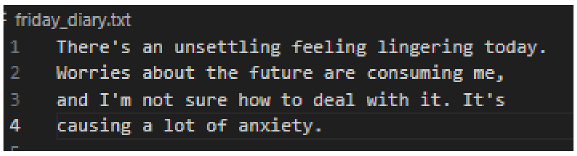


wensday_diary.txt

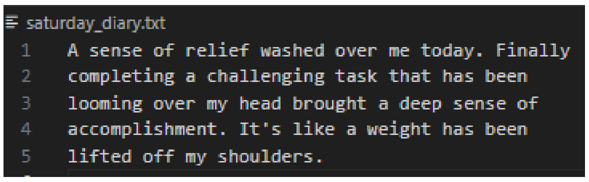
```
Feeling down all day today.
It seems like nothing interests me,
and there's a heavy feeling in my chest.
It's one of those days when everything feels
a bit too much.

```


thursday_diary.txt

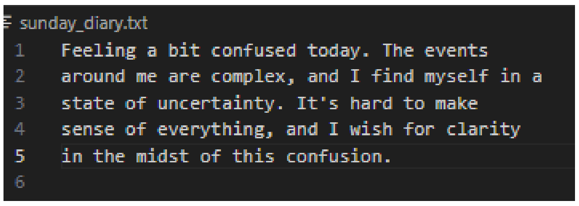
```
Left a lasting impression on someone today.
Feeling butterflies and great anticipation.
Excitement about what the future might hold is
bubbling inside me.
```


friday_diary.txt

```
There's an unsettling feeling lingering today.
Worries about the future are consuming me,
and I'm not sure how to deal with it. It's
causing a lot of anxiety.
```

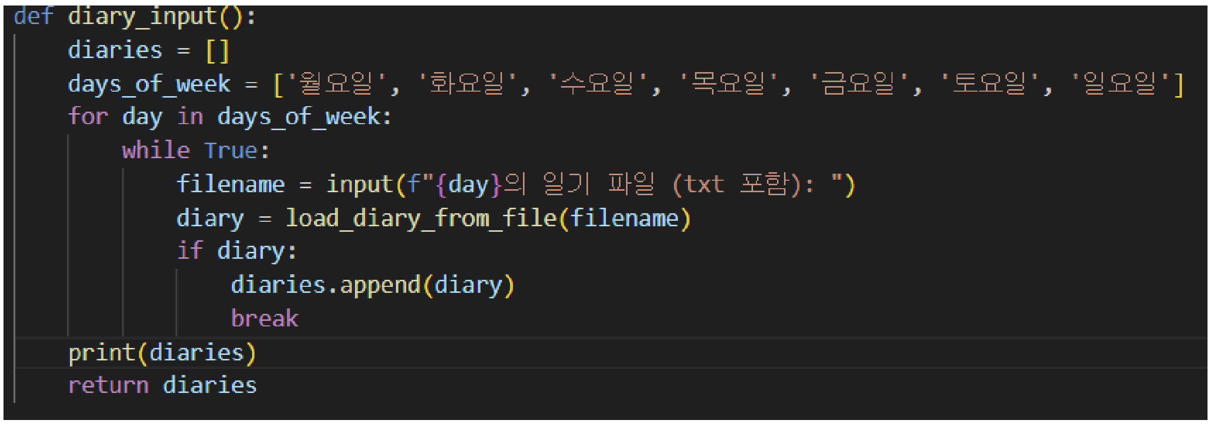

saturday_diary.txt

```
A sense of relief washed over me today. Finally
completing a challenging task that has been
looming over my head brought a deep sense of
accomplishment. It's like a weight has been
lifted off my shoulders.
```

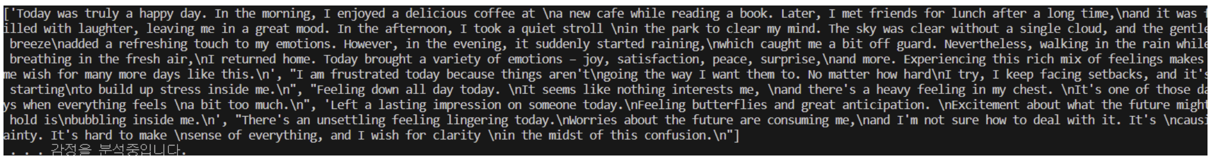

sunday_diary.txt

```
Feeling a bit confused today. The events
around me are complex, and I find myself in a
state of uncertainty. It's hard to make
sense of everything, and I wish for clarity
in the midst of this confusion.
```

② Calling the user's diary list.

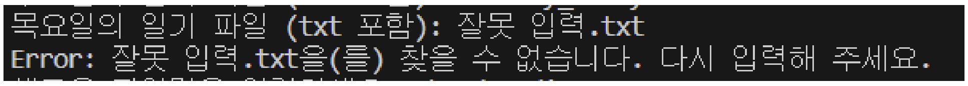


```python
def diary_input():
    diaries = []
    days_of_week = ['월요일', '화요일', '수요일', '목요일', '금요일', '토요일', '일요일']
    for day in days_of_week:
        while True:
            filename = input(f"{day}의 일기 파일 (txt 포함): ")
            diary = load_diary_from_file(filename)
            if diary:
                diaries.append(diary)
                break
    print(diaries)
    return diaries
```



```
['Today was truly a happy day. In the morning, I enjoyed a delicious coffee at \na new cafe while reading a book. Later, I met friends for lunch after a long time,\nand it was f
illed with laughter, leaving me in a great mood. In the afternoon, I took a quiet stroll \nin the park to clear my mind. The sky was clear without a single cloud, and the gentle
 breeze\nadded a refreshing touch to my emotions. However, in the evening, it suddenly started raining,\nwhich caught me a bit off guard. Nevertheless, walking in the rain while
 breathing in the fresh air,\nI returned home. Today brought a variety of emotions - joy, satisfaction, peace, surprise,\nand more. Experiencing this rich mix of feelings makes
me wish for many more days like this.\n', "I am frustrated today because things aren't\ngoing the way I want them to. No matter how hard\nI try, I keep facing setbacks, and it's
 starting\nto build up stress inside me.\n", "Feeling down all day today. \nIt seems like nothing interests me, \nand there's a heavy feeling in my chest. \nIt's one of those da
ys when everything feels \na bit too much.\n", 'Left a lasting impression on someone today.\nFeeling butterflies and great anticipation. \nExcitement about what the future might
 hold is\nbubbling inside me.\n', "There's an unsettling feeling lingering today.\nWorries about the future are consuming me,\nand I'm not sure how to deal with it. It's \ncausi
ainty. It's hard to make \nsense of everything, and I wish for clarity \nin the midst of this confusion.\n"]
. . . 감정을 분석중입니다.
```

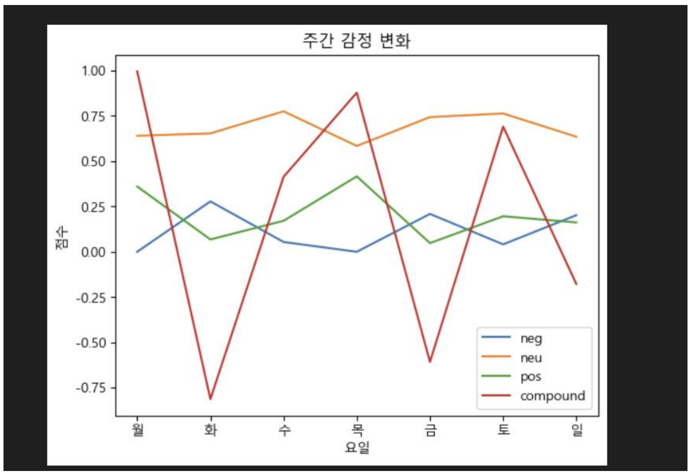
③ If the user entered the day of the week incorrectly.

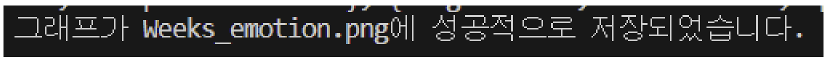


```
목요일의 일기 파일 (txt 포함): 잘못 입력.txt
Error: 잘못 입력.txt을(를) 찾을 수 없습니다. 다시 입력해 주세요.
```

**(2) Ability to visualize user diary weekly sentiment graphs and output them to a file**

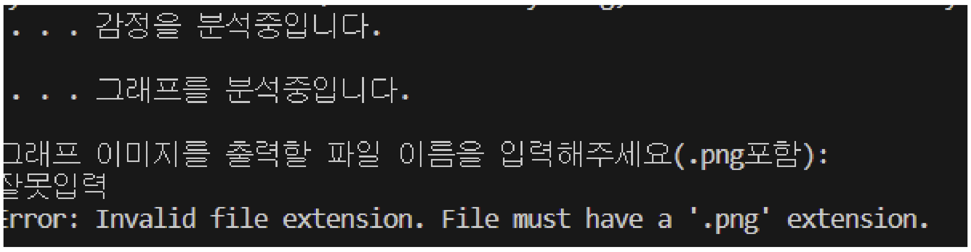
① Files saved as images





```
그래프가 Weeks_emotion.png에 성공적으로 저장되었습니다.
```

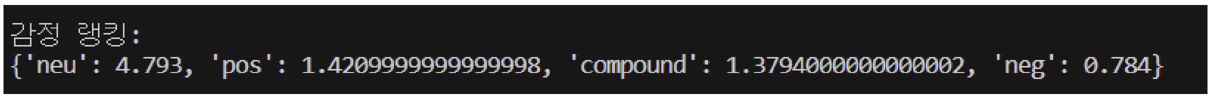
② If the user entered the graph file name incorrectly



```
. . . 감정을 분석중입니다.

. . . 그래프를 분석중입니다.

그래프 이미지를 출력할 파일 이름을 입력해주세요(.png포함):
잘못입력
Error: Invalid file extension. File must have a '.png' extension.
```

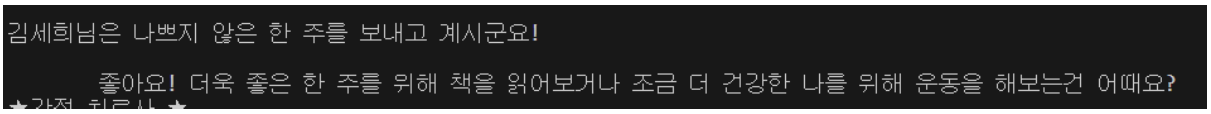
**(3) Providing solutions through weekly user sentiment analysis**

① Sentiment Ranking



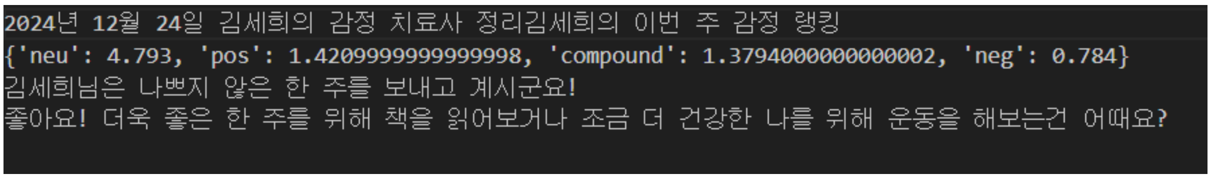
```
감정 랭킹:
{'neu': 4.793, 'pos': 1.4209999999999998, 'compound': 1.3794000000000002, 'neg': 0.784}
```

② Top Sentiment Solutions



```
김세희님은 나쁘지 않은 한 주를 보내고 계시군요!

        좋아요! 더욱 좋은 한 주를 위해 책을 읽어보거나 조금 더 건강한 나를 위해 운동을 해보는건 어때요?
```

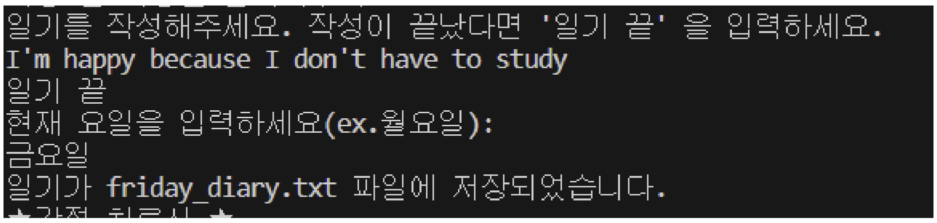
③ user.txt File



```
2024년 12월 24일 김세희의 감정 치료사 정리김세희의 이번 주 감정 랭킹
{'neu': 4.793, 'pos': 1.4209999999999998, 'compound': 1.3794000000000002, 'neg': 0.784}
김세희님은 나쁘지 않은 한 주를 보내고 계시군요!
좋아요! 더욱 좋은 한 주를 위해 책을 읽어보거나 조금 더 건강한 나를 위해 운동을 해보는건 어때요?
```

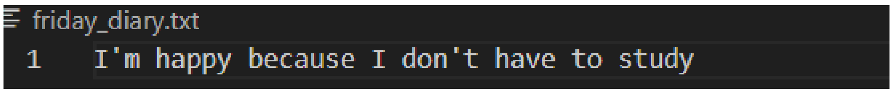
### (4) User diary writing feature

① Writing a diary and entering the day of the week.



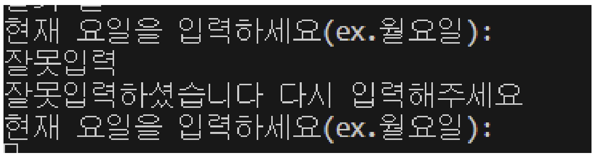
```
일기를 작성해주세요. 작성이 끝났다면 '일기 끝' 을 입력하세요.
I'm happy because I don't have to study
일기 끝
현재 요일을 입력하세요(ex.월요일):
금요일
일기가 friday_diary.txt 파일에 저장되었습니다.
```

② (e.g., Friday) Diary entry for Friday saved as 'friday_diary.txt'.



```
friday_diary.txt
I'm happy because I don't have to study
```

③ If the day of the week is entered incorrectly

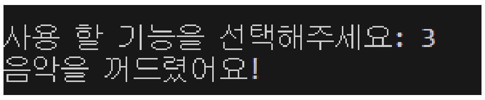


```
현재 요일을 입력하세요(ex.월요일):
잘못입력
잘못입력하셨습니다 다시 입력해주세요
현재 요일을 입력하세요(ex.월요일):
```

### (5) Playing Music

① The music plays well when the functionality is executed.

② If you want to stop the music



```
사용 할 기능을 선택해주세요: 3
음악을 꺼드렸어요!
```

## 5. Changes made after submitting the program proposal.

### 1) Changing the number of top emotions.

- Previous: Top 2 emotions.

- After: Top 1 emotion.

- reason: Since we are counting emotions for each week, it seems more appropriate to focus on providing solutions for one emotion rather than two.

### 2) Adding a feature.

- Previous: Two features - reading the user's diary and providing solutions.

- After: Added features for playing music, displaying a graph image, and writing a diary.

- Reason: Added more features to make the program more diverse, allowing users to better understand and delve deeper into their emotions.

## 6. Lessons Learned & Feedback

It was really challenging to create the program. Juggling it with assignments and exams added a lot of stress, especially during the later part of the course when I felt compared to students in the AI department. As a student from another department, I often criticized myself for not keeping up. However, as I gradually completed the program, I developed a deep affection for it, and I wanted to make it even better.

If I had more time, I believe I could have created a higher-quality program. Unfortunately, due to exams, I started a bit late, and time became a constraint. It's a bit disappointing. During the vacation, I'm considering further developing my programs. Also, I'd like to express my gratitude to the professor for always providing quick and kind responses to my questions. Programming was challenging, and I felt like giving up, but with the professor's help, I was fortunate enough to finish the program. It was a difficult journey, but I learned a lot. Thank you