Python programming and practice

Lunch menu recommendation

Progress Report : 3

Date : 2023.12.22

Name : 정승현

ID : 221186

1. Introduction

1) Background

These days, college students have budget constraints, and college students always eat at limited restaurants during lunch break, so choosing a lunch menu every day has become such a difficult task. To solve this difficult situation, a program is needed to recommend lunch menus according to the user's budget and cooking needs on the day.

2) Project goal

If college student users enter the food category and budget they want to eat, the goal is to randomly inform them of food and restaurants that fit the food category and budget.

3) Differences from existing programs

Existing programs simply enter the food category that users want to eat and randomly recommend menus, so there is a problem of recommending menus at ridiculous prices or difficult-to-eat menus. We ask users to enter not only the food category they want to eat but also the user's budget, and randomly recommend menus that take into account categories and budgets. And considering college student users who are sensitive to calories, writing calories next to random menus helps them select lunch menus. In this regard, menus can be recommended to users more realistically, and there is a difference from other programs.

2. Functional Requirement

1) Function 1 (Recommended function of menu)

It randomly recommends food within the food category that users want.

(1) Detailed function 1 (User's desired food category input)

Users will choose the category of food they want to eat.

For example, choose between Korean, Chinese, Japanese and Western food.

In addition, it randomly recommends food and restaurants according to the user's selected category.

(2) Detailed function 2 (Randomly display calories on the recommended menu)

For those who are sensitive to calories, write calories next to the randomly recommended menu to help you select the menu.

2) Function 2 (Recommended features based on price range)

The ability to recommend menus within the price range that users want

(1) Detailed function 1 (User enters maximum price)

When a user enters the maximum price he or she can pay, they randomly recommend restaurants and menus that fit below the maximum price.

3) Function 3 (Review and rating functions)

Users may leave reviews and ratings at the restaurant after finishing their food, and may not recommend or recommend the restaurant to other users.

4) Function 4 (Recommendation by reflecting the user's past eating history)

The ability to save the menu and reviews that the user ate when writing a review and randomly recommend one menu based on the saved review when the user writes the next review

5) Function 5 (Weather-based food recommendation function)

The ability to receive input for today's weather and randomly recommend food that suits the weather according to the weather

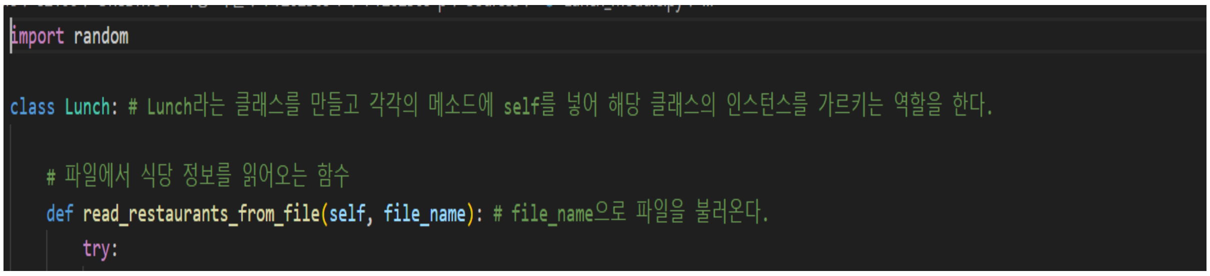
6) Function 6 (The ability to count the number of menus the user has eaten so far and show the five most eaten)

This feature helps users choose their lunch menu by showing them what food they've eaten the most so far.

3. Progress

1) Design and Implementation

(1) Function: class Lunch

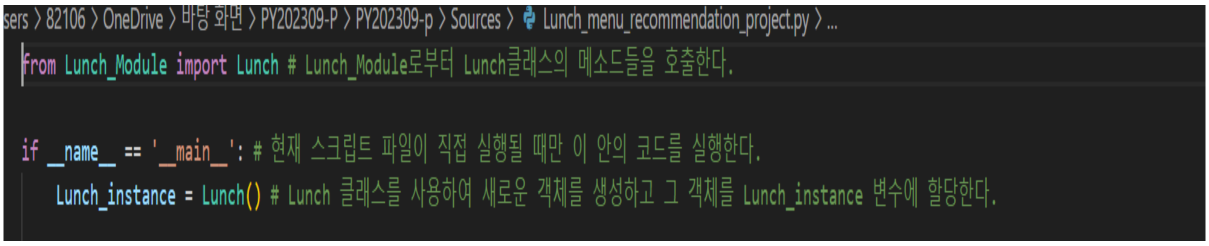


```
import random

class Lunch: # Lunch라는 클래스를 만들고 각각의 메소드에 self를 넣어 해당 클래스의 인스턴스를 가르키는 역할을 한다.

    # 파일에서 식당 정보를 읽어오는 함수
    def read_restaurants_from_file(self, file_name): # file_name으로 파일을 불러온다.
        try:
```

- Description: Create a class called Lunch and put itself in each method to point to an instance of that class.

- Applied Learning: A class was created and the functions and codes were concisely divided into two.

(2) Function: import Lunch

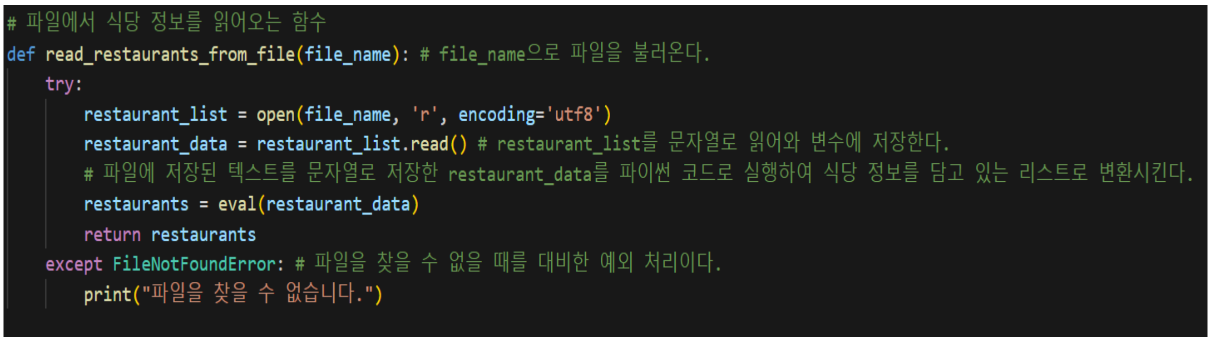


```
sers > 82106 > OneDrive > 바탕 화면 > PY202309-P > PY202309-p > Sources > Lunch_menu_recommendation_project.py > ...
from Lunch_Module import Lunch # Lunch_Module로부터 Lunch클래스의 메소드들을 호출한다.

if __name__ == '__main__': # 현재 스크립트 파일이 직접 실행될 때만 이 안의 코드를 실행한다.
    Lunch_instance = Lunch() # Lunch 클래스를 사용하여 새로운 객체를 생성하고 그 객체를 Lunch_instance 변수에 할당한다.
```

- Description: Call the mesos of the Lunch class from the Lunch_Module because the file name containing the Lunch class was made into Lunch_Module. And make sure that the code is executed only when the current script file is executed correctly. Create a new object using the Lunch class and assign that object to the Lunch_instance variable.

- Applied Learning:class was called. Module '__main__' was used.

(3) Function: read_restaurants_from_file(file_name)



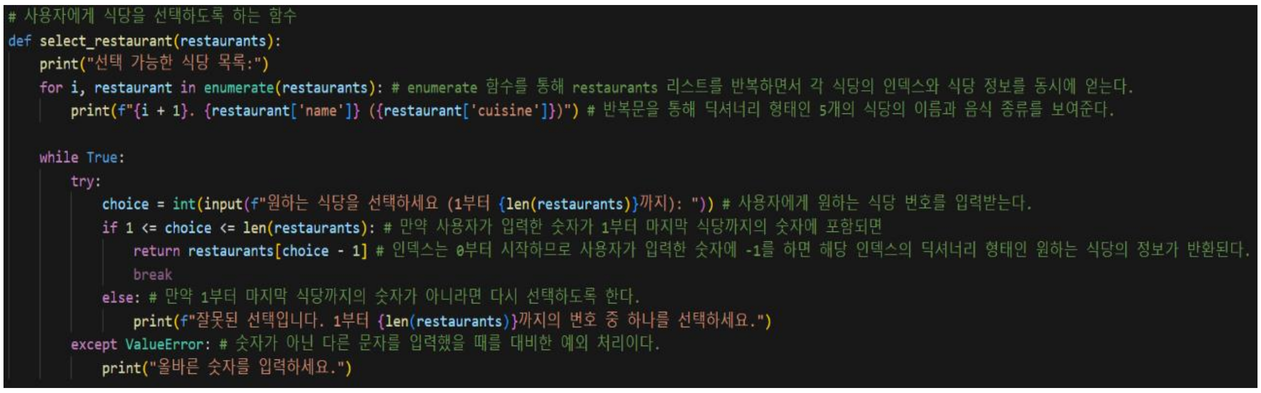
```python
# 파일에서 식당 정보를 읽어오는 함수
def read_restaurants_from_file(file_name): # file_name으로 파일을 불러온다.
    try:
        restaurant_list = open(file_name, 'r', encoding='utf8')
        restaurant_data = restaurant_list.read() # restaurant_list를 문자열로 읽어와 변수에 저장한다.
        # 파일에 저장된 텍스트를 문자열로 저장한 restaurant_data를 파이썬 코드로 실행하여 식당 정보를 담고 있는 리스트로 변환시킨다.
        restaurants = eval(restaurant_data)
        return restaurants
    except FileNotFoundError: # 파일을 찾을 수 없을 때를 대비한 예외 처리이다.
        print("파일을 찾을 수 없습니다.")
```

- I/O: If you import a file as a parameter, convert the restaurant information into a list and return the list.

- Description: Load the file with the parameter file_name and save it to the restaurant_list. In addition, the restaurant_list is read as a string in the restaurant_data and stored as a variable. Then, the restaurant_data storing the text stored in the file as a string is executed with Python code to convert it into a list containing restaurant information.

- Applied Learning: 1) The process of retrieving restaurant information from the file was saved as a function. 2) Exceptional processing was used to prepare for when the file did not open. 3) The file imported through the parameter was read and converted into a character string through file reading and writing.

(4) **Function: select_restaurant(restaurants)**

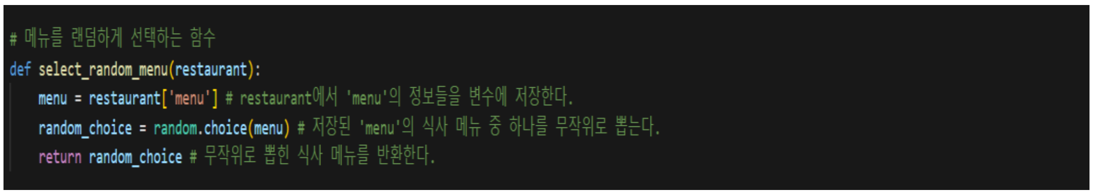


```python
# 사용자에게 식당을 선택하도록 하는 함수
def select_restaurant(restaurants):
    print("선택 가능한 식당 목록:")
    for i, restaurant in enumerate(restaurants): # enumerate 함수를 통해 restaurants 리스트를 반복하면서 각 식당의 인덱스와 식당 정보를 동시에 얻는다.
        print(f"{i + 1}. {restaurant['name']} ({restaurant['cuisine']})") # 반복문을 통해 딕셔너리 형태인 5개의 식당의 이름과 음식 종류를 보여준다.

    while True:
        try:
            choice = int(input(f"원하는 식당을 선택하세요 (1부터 {len(restaurants)}까지): ")) # 사용자에게 원하는 식당 번호를 입력받는다.
            if 1 <= choice <= len(restaurants): # 만약 사용자가 입력한 숫자가 1부터 마지막 식당까지의 숫자에 포함되면
                return restaurants[choice - 1] # 인덱스는 0부터 시작하므로 사용자가 입력한 숫자에 -1를 하면 해당 인덱스의 딕셔너리 형태인 원하는 식당의 정보가 반환된다.
                break
            else: # 만약 1부터 마지막 식당까지의 숫자가 아니라면 다시 선택하도록 한다.
                print(f"잘못된 선택입니다. 1부터 {len(restaurants)}까지의 번호 중 하나를 선택하세요.")
        except ValueError: # 숫자가 아닌 다른 문자를 입력했을 때를 대비한 예외 처리이다.
            print("올바른 숫자를 입력하세요.")
```

- Description: By repeating the list of restaurants through the enumeration function, the index of each restaurant and restaurant information are obtained at the same time. The names and food types of five restaurants in the form of dictionaries are shown through iterations of the index and restaurant information of each restaurant. In addition, the user receives the desired restaurant number through the repetition statement. If the number entered by the user is included in the number from 1 to the last restaurant, the index starts from 0, so if -1 is applied to the number entered by the user, the information of the desired restaurant, in the form of a dictionary of the index, is returned. If it is not a number from 1 to the last restaurant, choose again. The last exception handling is in preparation for when the user enters a character other than a number.

- Applied Learning: 1) I created a function called select_restaurant(). 2) Restaurant information was obtained using a repeat. 3) The user was asked to enter the restaurant number in the repetition. 4) Exception handling was used. 5) The restaurant information of the dictionary was returned through a conditional statement.

(5) Function: select_random_menu(restauants)



```python
# 메뉴를 랜덤하게 선택하는 함수
def select_random_menu(restaurant):
    menu = restaurant['menu'] # restaurant에서 'menu'의 정보들을 변수에 저장한다.
    random_choice = random.choice(menu) # 저장된 'menu'의 식사 메뉴 중 하나를 무작위로 뽑는다.
    return random_choice # 무작위로 뽑힌 식사 메뉴를 반환한다.
```

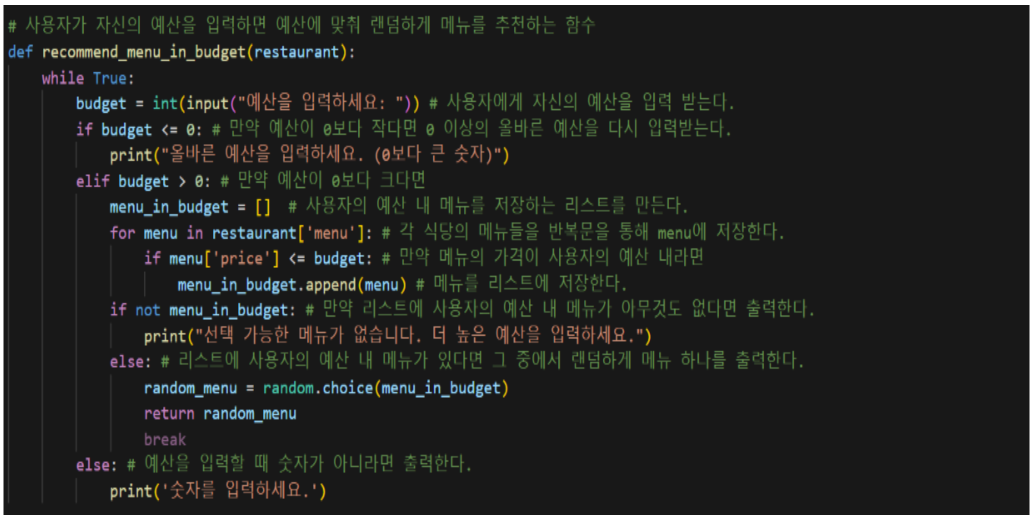
- I/O: Store one of the restaurant's menus randomly in random_choice and return random_choice.

- Description: The restaurant stores the information of 'menu' in a variable. Randomly pick one of the saved 'menu' meal menus. And return a randomly selected meal menu.

- Applied Learning: 1) I created a function called select_random_menu(). 2)

Random modules were used.

(6) Function: recommend_menu_in_budget(restaurant)

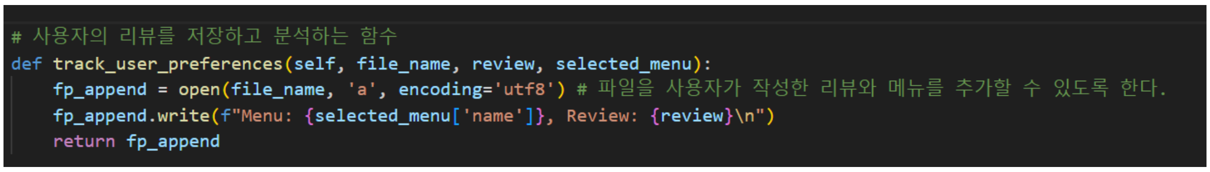


```python
# 사용자가 자신의 예산을 입력하면 예산에 맞춰 랜덤하게 메뉴를 추천하는 함수
def recommend_menu_in_budget(restaurant):
    while True:
        budget = int(input("예산을 입력하세요: ")) # 사용자에게 자신의 예산을 입력 받는다.
        if budget <= 0: # 만약 예산이 0보다 작다면 0 이상의 올바른 예산을 다시 입력받는다.
            print("올바른 예산을 입력하세요. (0보다 큰 숫자)")
        elif budget > 0: # 만약 예산이 0보다 크다면
            menu_in_budget = []  # 사용자의 예산 내 메뉴를 저장하는 리스트를 만든다.
            for menu in restaurant['menu']: # 각 식당의 메뉴들을 반복문을 통해 menu에 저장한다.
                if menu['price'] <= budget: # 만약 메뉴의 가격이 사용자의 예산 내라면
                    menu_in_budget.append(menu) # 메뉴를 리스트에 저장한다.
            if not menu_in_budget: # 만약 리스트에 사용자의 예산 내 메뉴가 아무것도 없다면 출력한다.
                print("선택 가능한 메뉴가 없습니다. 더 높은 예산을 입력하세요.")
            else: # 리스트에 사용자의 예산 내 메뉴가 있다면 그 중에서 랜덤하게 메뉴 하나를 출력한다.
                random_menu = random.choice(menu_in_budget)
                return random_menu
                break
        else: # 예산을 입력할 때 숫자가 아니라면 출력한다.
            print('숫자를 입력하세요.')
```

- Description: First, you get your budget input from the user. And if the budget received is less than 0, the correct budget is re-entered above zero, and if the budget is better than zero, create a list that stores the menu within the user's budget. Save each restaurant's menu in menu through an iteration. If the price of the menu is within the user's budget, save the menu to the list. If there is nothing on the list within the user's budget, print out that there is nothing on the list. And if there is a menu within the user's budget in the list, one of them is randomly output.

- Applied Learning: 1) I created a function called recommend_menu_in_budget(). 2) The user was asked to enter the budget through the repetition statement. 3) The conditional statement determined whether the budget was greater than or less than zero. 4) Random modules were used.

(7) Function: track_user_preferences(self, file_name, review, selected_menu)

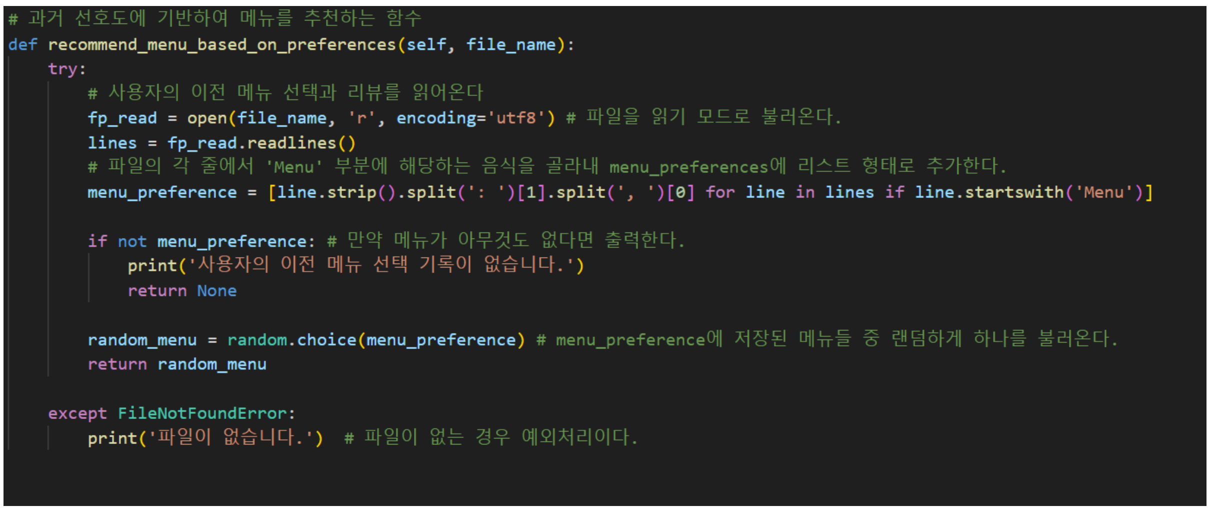


```python
# 사용자의 리뷰를 저장하고 분석하는 함수
def track_user_preferences(self, file_name, review, selected_menu):
    fp_append = open(file_name, 'a', encoding='utf8') # 파일을 사용자가 작성한 리뷰와 메뉴를 추가할 수 있도록 한다.
    fp_append.write(f"Menu: {selected_menu['name']}, Review: {review}\n")
    return fp_append
```

- I/O: Add user-eaten food and user-written reviews to the file and then return the file.
- Description: Adds the food the user ate and the user-written review to the file (file_name received as a parameter).

- Applied Learning: 1) Functionalize, 2) Read and write files

(8) Function: recommend_menu_based_on_preferences(self, file_name)



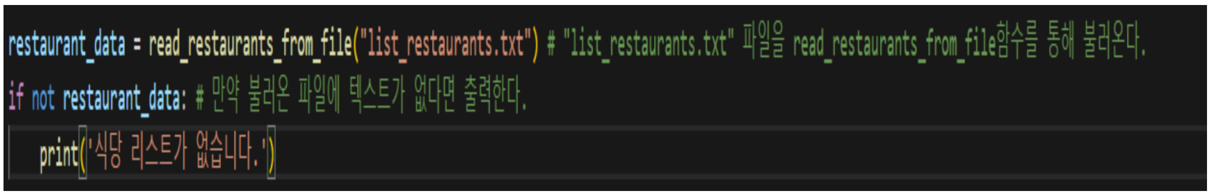
```python
# 과거 선호도에 기반하여 메뉴를 추천하는 함수
def recommend_menu_based_on_preferences(self, file_name):
    try:
        # 사용자의 이전 메뉴 선택과 리뷰를 읽어온다
        fp_read = open(file_name, 'r', encoding='utf8') # 파일을 읽기 모드로 불러온다.
        lines = fp_read.readlines()
        # 파일의 각 줄에서 'Menu' 부분에 해당하는 음식을 골라내 menu_preferences에 리스트 형태로 추가한다.
        menu_preference = [line.strip().split(': ')[1].split(', ')[0] for line in lines if line.startswith('Menu')]

        if not menu_preference: # 만약 메뉴가 아무것도 없다면 출력한다.
            print('사용자의 이전 메뉴 선택 기록이 없습니다.')
            return None

        random_menu = random.choice(menu_preference) # menu_preference에 저장된 메뉴들 중 랜덤하게 하나를 불러온다.
        return random_menu

    except FileNotFoundError:
        print('파일이 없습니다.')  # 파일이 없는 경우 예외처리이다.
```

- Description: After reading and fetching a file (file_name received as a parameter) in which a review written by the user is saved, the food corresponding to Menu is stored in the variable. Randomly fetching and returning one of the stored foods.

- Applied Learning: 1) Functionalization of functions 2) Various Python built-in functions 3) Repeated statements 4) File reading and writing 5)

Conditional statements 6) Random functions 7) Exception handling

(9) Function:



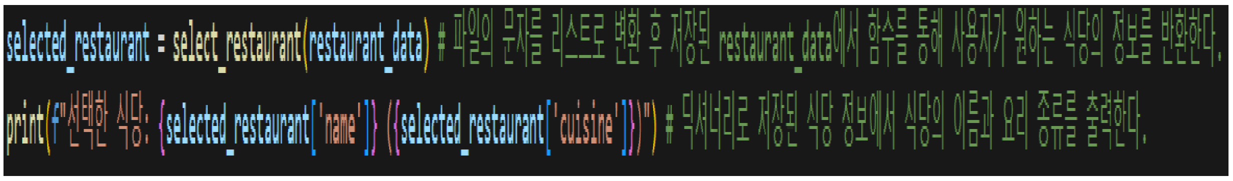
```python
restaurant_data = read_restaurants_from_file("list_restaurants.txt") # "list_restaurants.txt" 파일을 read_restaurants_from_file함수를 통해 불러온다.
if not restaurant_data: # 만약 불러온 파일에 텍스트가 없다면 출력한다.
    print('식당 리스트가 없습니다.')
```

- I/O: Gets restaurant information from file and outputs 'No restaurant list' if not.

- Description: The read_restaurants_from_file() function fetches and reads the information about the restaurants 'list_restaurants_txt'. And store it in the restaurant_data variable. Through conditional statements, if there is no text in the imported file, the output statement is printed.

- Applied Learning: 1) I saved the function in the variable. 2) Through conditional statements, if there is no text in the imported file, the output statement is printed.

(10) Function:



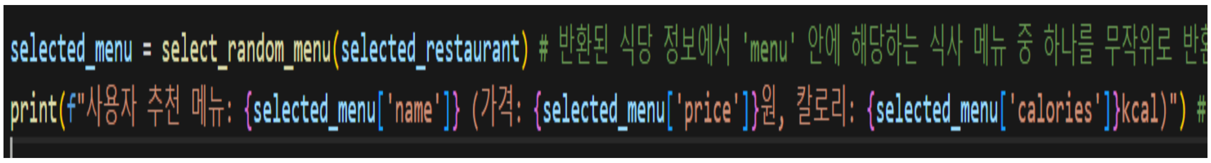
```python
selected_restaurant = select_restaurant(restaurant_data) # 파일의 문자를 리스트로 변환 후 저장된 restaurant_data에서 함수를 통해 사용자가 원하는 식당의 정보를 반환한다.
print(f"선택한 식당: {selected_restaurant['name']} ({selected_restaurant['cuisine']})") # 딕셔너리로 저장된 식당 정보에서 식당의 이름과 요리 종류를 출력한다.
```

- I/O: The user inputs the type of food he/she wants to eat and outputs the type of restaurant and dish.

- Description: After converting the characters in the file into a list, the stored restaurant_data returns the information of the restaurant that the user wants through the function. And the name of the restaurant and the type of dish are output from the restaurant information stored in the dictionary.

- Applied Learning: 1) I saved the function in the variable. 2) It shows the value values of the dictionary.

(11) Function:



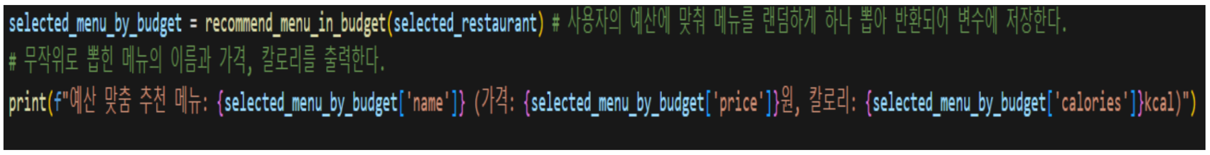
```
selected_menu = select_random_menu(selected_restaurant) # 반환된 식당 정보에서 'menu' 안에 해당하는 식사 메뉴 중 하나를 무작위로 반
print(f"사용자 추천 메뉴: {selected_menu['name']} (가격: {selected_menu['price']}원, 칼로리: {selected_menu['calories']}kcal)") #
```

- I/O: Prints the name, price, and calories of a randomly selected menu.

- Description: In the returned restaurant information, one of the meal menus corresponding to the 'menu' is randomly returned and stored in the variable. It also prints the name, price, and calories of the randomly selected menu.

- Applied Learning: 1) I saved the function in the variable. 2) It shows the value values of the dictionary.

(12) Function:



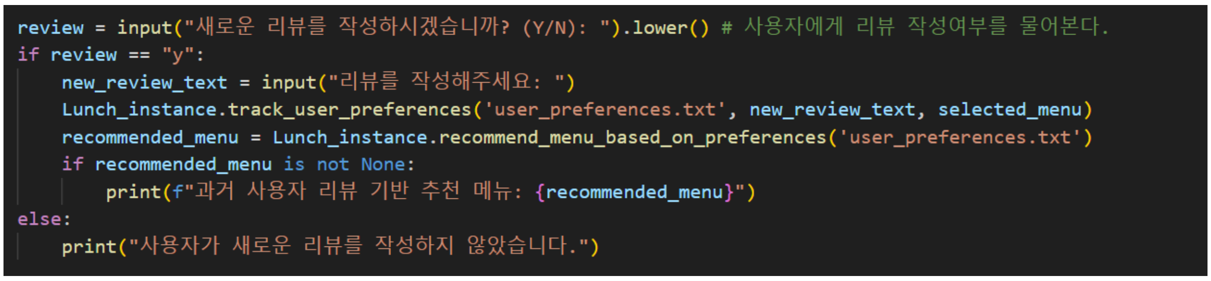
```
selected_menu_by_budget = recommend_menu_in_budget(selected_restaurant) # 사용자의 예산에 맞춰 메뉴를 랜덤하게 하나 뽑아 반환되어 변수에 저장한다.
# 무작위로 뽑힌 메뉴의 이름과 가격, 칼로리를 출력한다.
print(f"예산 맞춤 추천 메뉴: {selected_menu_by_budget['name']} (가격: {selected_menu_by_budget['price']}원, 칼로리: {selected_menu_by_budget['calories']}kcal)")
```

- I/O: When a user enters a budget through the referral_menu_in_budget() function, the menu is randomly selected according to the budget and the name, price, and calories of the menu are output.

- Description: Through the referral_menu_in_budget() function, a menu is randomly selected according to the user's budget, returned, and stored in a variable. It also prints the name, price, and calories of the randomly selected menu.

- Applied Learning: 1) I saved the function in the variable. 2) It shows the value values of the dictionary.

(13) Function:



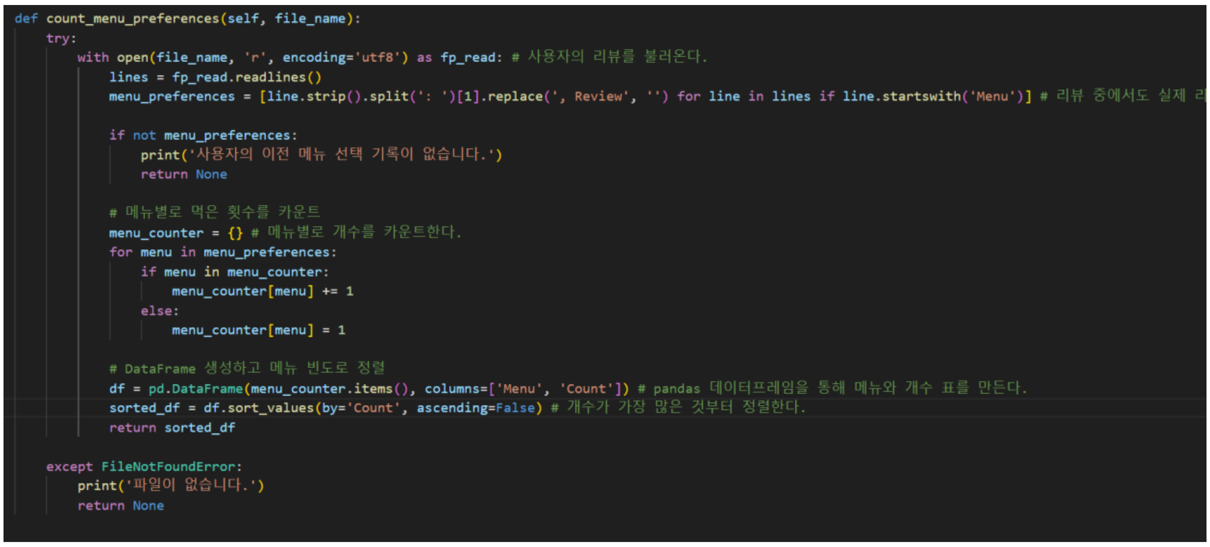
```python
review = input("새로운 리뷰를 작성하시겠습니까? (Y/N): ").lower() # 사용자에게 리뷰 작성여부를 물어본다.
if review == "y":
    new_review_text = input("리뷰를 작성해주세요: ")
    Lunch_instance.track_user_preferences('user_preferences.txt', new_review_text, selected_menu)
    recommended_menu = Lunch_instance.recommend_menu_based_on_preferences('user_preferences.txt')
    if recommended_menu is not None:
        print(f"과거 사용자 리뷰 기반 추천 메뉴: {recommended_menu}")
else:
    print("사용자가 새로운 리뷰를 작성하지 않았습니다.")
```

- I/O: The user writes a review and then randomly returns one of the menus corresponding to each review.

- Description: After asking the user whether to write a review, if they do, save the menu and review in 'user_preferences.txt' through the track_menu_based_on_preferences function. And through the referral_menu_based_on_preferences function, one of the menus stored in 'user_preferences.txt' is randomly returned and output.

- Applied Learning: 1) Conditional Statement, 2) User Input

(14) Function: count_menu_preferences(self, file_name)

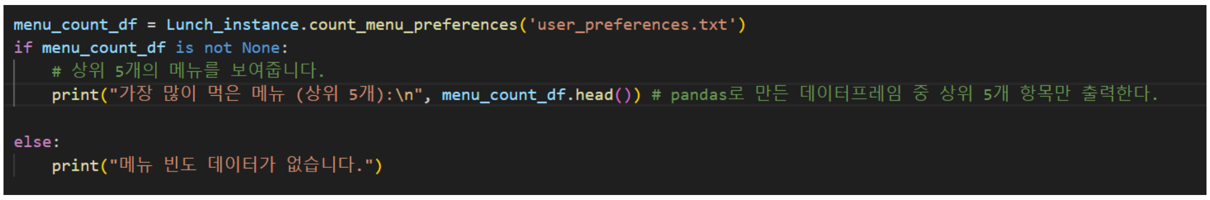


```python
def count_menu_preferences(self, file_name):
    try:
        with open(file_name, 'r', encoding='utf8') as fp_read: # 사용자의 리뷰를 불러온다.
            lines = fp_read.readlines()
            menu_preferences = [line.strip().split(': ')[1].replace(', Review', '') for line in lines if line.startswith('Menu')] # 리뷰 중에서도 실제 리

            if not menu_preferences:
                print('사용자의 이전 메뉴 선택 기록이 없습니다.')
                return None

            # 메뉴별로 먹은 횟수를 카운트
            menu_counter = {} # 메뉴별로 개수를 카운트한다.
            for menu in menu_preferences:
                if menu in menu_counter:
                    menu_counter[menu] += 1
                else:
                    menu_counter[menu] = 1

            # DataFrame 생성하고 메뉴 빈도로 정렬
            df = pd.DataFrame(menu_counter.items(), columns=['Menu', 'Count']) # pandas 데이터프레임을 통해 메뉴와 개수 표를 만든다.
            sorted_df = df.sort_values(by='Count', ascending=False) # 개수가 가장 많은 것부터 정렬한다.
            return sorted_df

    except FileNotFoundError:
        print('파일이 없습니다.')
        return None
```

- Description: After fetching the user's review, only the food names corresponding to Menu are saved as a list through the repetition. Then, make a blank dictionary, count the food names there, and save it. Then, based on the dictionary, create a data frame through pandas and sort the most counted ones first. Then return the sorted data frame.

- Applied Learning: 1) pandas 2) exception handling 3) functions 4) file read and write 5) conditional statements

(15) Function:



```python
menu_count_df = Lunch_instance.count_menu_preferences('user_preferences.txt')
if menu_count_df is not None:
    # 상위 5개의 메뉴를 보여줍니다.
    print("가장 많이 먹은 메뉴 (상위 5개):\n", menu_count_df.head()) # pandas로 만든 데이터프레임 중 상위 5개 항목만 출력한다.

else:
    print("메뉴 빈도 데이터가 없습니다.")
```

- Description: The number of menus in the review written by the user is made into a data frame, showing the top five menus that the user ate the most.

- Applied Learning: 1) Conditional Statement 2) Pandas

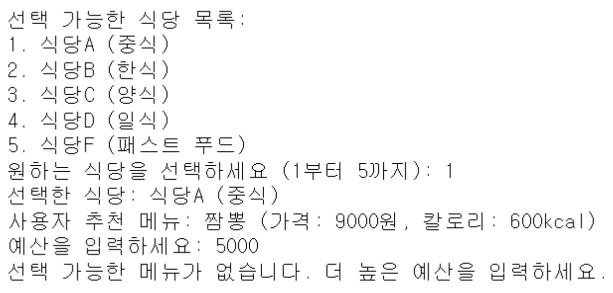
2) Testing

- (1) When the user selects the desired restaurant and the menu is randomly recommended at that restaurant

```
선택 가능한 식당 목록:
1. 식당A (중식)
2. 식당B (한식)
3. 식당C (양식)
4. 식당D (일식)
5. 식당F (패스트 푸드)
원하는 식당을 선택하세요 (1부터 5까지): 1
선택한 식당: 식당A (중식)
사용자 추천 메뉴: 짬뽕 (가격: 9000원, 칼로리: 600kcal)
```

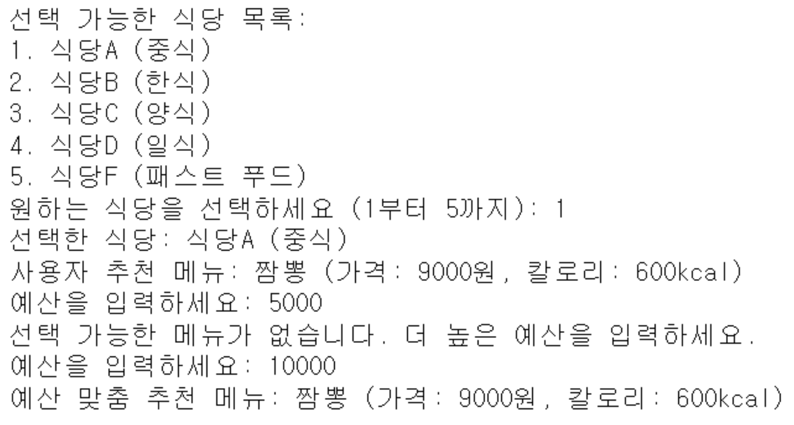
- (2) When a user writes down his or her budget and the menu is randomly recommended within his or her budget

1) When there is no menu available within the budget



```
선택 가능한 식당 목록:
1. 식당A (중식)
2. 식당B (한식)
3. 식당C (양식)
4. 식당D (일식)
5. 식당F (패스트 푸드)
원하는 식당을 선택하세요 (1부터 5까지): 1
선택한 식당: 식당A (중식)
사용자 추천 메뉴: 짬뽕 (가격: 9000원, 칼로리: 600kcal)
예산을 입력하세요: 5000
선택 가능한 메뉴가 없습니다. 더 높은 예산을 입력하세요.
```

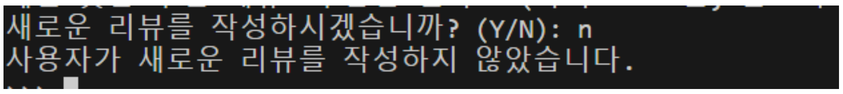
2) There's a menu that you can choose within your budget, so when you recommend a random menu



```
선택 가능한 식당 목록:
1. 식당A (중식)
2. 식당B (한식)
3. 식당C (양식)
4. 식당D (일식)
5. 식당F (패스트 푸드)
원하는 식당을 선택하세요 (1부터 5까지): 1
선택한 식당: 식당A (중식)
사용자 추천 메뉴: 짬뽕 (가격: 9000원, 칼로리: 600kcal)
예산을 입력하세요: 5000
선택 가능한 메뉴가 없습니다. 더 높은 예산을 입력하세요.
예산을 입력하세요: 10000
예산 맞춤 추천 메뉴: 짬뽕 (가격: 9000원, 칼로리: 600kcal)
```

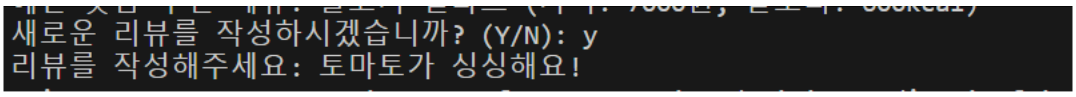
- **(3)** When users finish eating and write a review of the restaurant

1) When a user is not writing a review



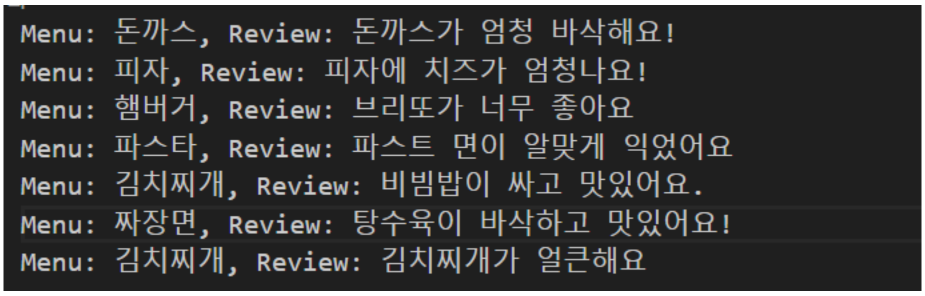
```
새로운 리뷰를 작성하시겠습니까? (Y/N): n
사용자가 새로운 리뷰를 작성하지 않았습니다.
```

2) When a user writes a review



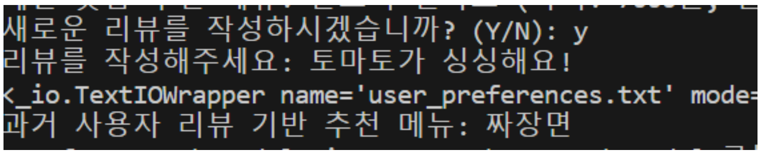
```
새로운 리뷰를 작성하시겠습니까? (Y/N): y
리뷰를 작성해주세요: 토마토가 싱싱해요!
```

- When a user writes a review, the menu and review that the user ate are saved



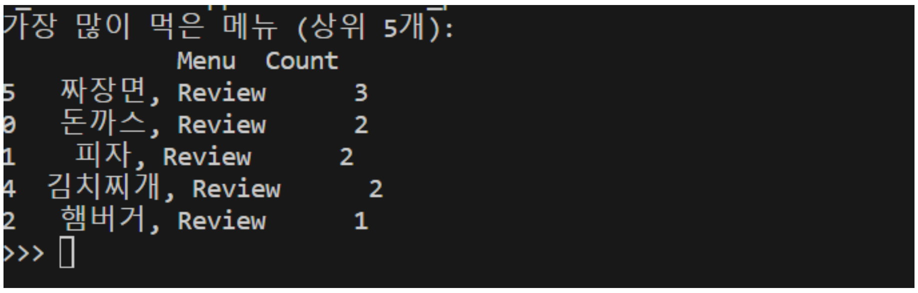
```
Menu: 돈까스, Review: 돈까스가 엄청 바삭해요!
Menu: 피자, Review: 피자에 치즈가 엄청나요!
Menu: 햄버거, Review: 브리또가 너무 좋아요
Menu: 파스타, Review: 파스트 면이 알맞게 익었어요
Menu: 김치찌개, Review: 비빔밥이 싸고 맛있어요.
Menu: 짜장면, Review: 탕수육이 바삭하고 맛있어요!
Menu: 김치찌개, Review: 김치찌개가 얼큰해요
```

3) Based on the user's review, one menu is randomly recommended.



```
새로운 리뷰를 작성하시겠습니까? (Y/N): y
리뷰를 작성해주세요: 토마토가 싱싱해요!
<_io.TextIOWrapper name='user_preferences.txt' mode=
과거 사용자 리뷰 기반 추천 메뉴: 짜장면
```

- (4) Print out the top five items among the menus the user ate.



```
가장 많이 먹은 메뉴 (상위 5개):
           Menu  Count
5   짜장면, Review     3
0   돈까스, Review     2
1    피자, Review     2
4  김치찌개, Review     2
2   햄버거, Review     1
>>> 
```

4. Changes in Comparison to the Plan

1) In retrieving restaurant information, I wanted to update restaurants based on the actual user's location and recommend menus appropriate to the user, but this level was not implemented because there was still a limit to learning.

5. Schedule

- Progress Indication

업무		11/3	11/10	11/17	…….
제안서 작성		완료			
기능1	세부기능1		완료		
기능2	세부기능1			완료	
기능3	세부기능 1				완료
기능4	세부기능 1				Recommendation by reflecting the user's past eating history code
기능 +a	Add code if it has better features, and modify any parts that are not executed or need to be modified from time to time.				