

Python programming and practice

# ***Lunch menu recommendation***

**Progress Report : 1**

Date : 2023.11.23

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)

Remember the user's past history and recommend food menus.

## 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)

```

# 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)

```
# 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 -1을 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(restaurants)

```
# 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)

```
# 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: `write_review()`

```
# 사용자 리뷰 입력 함수
def write_review():
    review = input("식당에 대한 리뷰를 작성하시겠습니까? (Y/N): ").lower() # 사용자에게 리뷰를 작성할건지에 대해 물어본다.
    if review == "y": # 만약 작성한다면(Y) 리뷰를 작성하고 review_text 변수에 저장한다.
        review_text = input("리뷰를 작성해주세요: ")
        return review_text
    else: # 작성 하지 않으면(N) 그냥 return문을 반환한다.
        return "사용자가 리뷰를 작성하지 않았습니다."
```

- I/O: When a user writes a review, it returns a review; otherwise, it returns "You did not write a review."

- Description: Ask the user whether to write a review in the `write_review()` function, and if they say 'y', let them write a review. However, if you answer 'n', you return the return statement.

- Applied Learning: 1) I created a function called `write_review()`. 2) The conditional statement asks if the user wants to write a review.

## (8) Function:



```
restaurant_data = read_restaurants_from_file("list_restaurants.txt") # "list_restaurants.txt" 파일을 read_restaurants_from_file 함수를 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.

## (9) Function:

```
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.

## (10) Function:

```
selected_menu = select_random_menu(selected_restaurant) # 반환된 식당 정보에서 'menu' 안에 해당하는 식사 메뉴 중 하나를 무작위로 반환
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.

## (11) 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.

## (12) Function:

```
review = write_review() # review 받기
print("리뷰: " + review)
```

- I/O: Outputs user-written reviews.

- Description: Ask the user whether to write a review through the write\_review() function, save the written review to the review, and then output it.

- Applied Learning: 1) I saved the function in the variable.

## 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

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)

식당에 대한 리뷰를 작성하시겠습니까? (Y/N): n

리뷰: 사용자가 리뷰를 작성하지 않았습니다.

### 2) When a user writes a review

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: 10000

예산 맞춤 추천 메뉴: 짜장면 (가격: 8000원, 칼로리: 500kcal)

식당에 대한 리뷰를 작성하시겠습니까? (Y/N): y

리뷰를 작성해주세요: 음식이 맛있고 빨리 나와서 좋아요!

리뷰: 음식이 맛있고 빨리 나와서 좋아요!

## 4. Changes in Comparison to the Plan

1) It's going well as planned at the moment. So there are no changes yet.

## 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.				