Python programming

Savvy plate

Final Report

Date : 2023/12/24
Name : 안수빈
ID :214309

1. Introduction

This system is a Python program that recommends recipes based on the ingredients users have and helps manage groceries. When a user inputs the ingredients they currently have, it recommends possible recipes and creates a shopping list to assist with purchasing the necessary ingredients for the recipes. This system can contribute to reducing food waste and economical meal planning.

1) Background

In our busy lives, it is common to forget about purchased ingredients or let them go to waste after they pass their expiration date. Moreover, facing a variety of ingredients in the refrigerator without deciding what to make can cause significant stress for users. To solve this, we have recognized the need to develop a program that not only encourages users to effectively use the ingredients they already have but also optimizes the purchase and consumption of groceries.

This program will suggest recipes that utilize forgotten ingredients and features the ability to track the expiration dates of groceries, alerting users to use them within the timeframe to prevent waste. By managing a list of necessary groceries during shopping, it will promote an economical and environmentally sustainable diet.

2) Project goal

The goal is to develop a program that recommends personalized recipes based on the inventory of ingredients users currently have. This program aims to facilitate efficient management of ingredients and ease the shopping process, enabling users to effectively utilize their groceries, reduce waste, and make informed decisions regarding their ingredients.

3) Differences from existing programs

Most recipe recommendation services require users to actively search for specific recipes, demanding a passive approach to finding recipes that suit their dietary habits. "Savvy Plate" differentiates itself in the following ways:

- Automatic recipe suggestions: Based on the ingredients users have, "Savvy Plate" automatically suggests recipes. This saves time and provides new cooking inspiration to users.

- Ingredient substitution suggestions: If a user doesn't have a specific ingredient, "Savvy Plate" recommends other ingredients that can be used as substitutes.

2. Functional Requirement

1) Function 1 (Ingredient-based recipe recommendation feature.)

- It analyzes the ingredients entered by the user to recommend possible recipes.

(1) Detailed function 1 (User ingredient input analysis.)

- When a user enters their ingredients, the program analyzes the list of entered ingredients to search a recipe database that can use those ingredients.

(2) Detailed function 2 (Alternative ingredient suggestions.)

If some of the ingredients the user has do not exactly match those needed for a recipe, the program will suggest possible alternatives. This feature enhances the flexibility of recipes and helps users to cook easily.

2) Function 2 (Ingredient Management and Shopping List Creation)

-Through the recommended recipes, users can identify additional ingredients needed and manage them as a shopping list.

(1) Detailed function 1 (Automatic Identification of Required Ingredients)

Based on the recommended recipes, the program automatically identifies ingredients that the user does not currently have. It compares the list of entered ingredients with the ingredients list of the recommended recipe to find missing items.

3. Implementation

(1) Automatic Recipe Recommendation and Shopping List Generation

Input/Output:

Input: Users input their available ingredients, separated by commas.

Ouput: The program displays a list of possible recipes and provides a list of missing ingredients and a shopping list for the selected recipe.

Description: This system takes an input list of ingredients from the user and searches the 'recipes_db' database to recommend recipes that can be made with those ingredients. If the user selects a specific recipe, the system identifies any ingredients that the user does not have and suggests possible substitutes if necessary. Additionally, it generates a shopping list based on this information, and offers the functionality to save this list if the user desires.

Applied Learning:

1. Loops: The **for** loop is utilized to iterate through all recipes in the and compare them with the user's ingredients.

2. Conditional Statements:

The first **if not input_ingredients.strip()** checks if the user has entered at least one ingredient. If not, it prompts the user to enter at least one ingredient.

The second **if available_recipes** checks if there are recipes available based on the user's input ingredients. If not, it informs the user.

3. Functions: The code is well-structured with functions such as **find_recipes, suggest_alternatives, create_shopping_list,** and **print_recipe_instructions.** These functions help modularize your code, making it more readable and maintainable.

-find_recipes: Takes user ingredients and the recipes database, returning a list of recommended recipes based on user ingredients.

-suggest_alternatives: Provides alternative ingredients based on a given ingredient.

-create_shopping_list: Generates a shopping list for the selected recipe based on user ingredients.

-print_recipe_instructions: Prints instructions for a selected recipe.

4. Modules: The import statements at the beginning of the **main.py** file bring in functions and modules from external files(function.py, shopping_list_manager.py,recipes_module.py). This demonstrates the use of modular programming for better organization and reusability.

5. Dictionaries: The **recipes_db** and **alternatives_db** use dictionaries to manage the list of recipes and the list of ingredient substitutes, respectively.

6. Lists: Ingredients input by the user are stored in a list, and the recipe ingredients as well as the shopping list are managed using list data structures.

7. Exception Handling: The try-except blocks handle potential errors such as invalid input for the selected recipe index and ensure a graceful response to user mistakes.

Code screenshot

[main.py]

```python
from function import find_recipes, suggest_alternatives, create_shopping_list, print_recipe_instructions
from shopping_list_manager import save_shopping_list
import recipes_module


while True:
    # 사용자로부터 재료를 입력받기
    input_ingredients = input("가지고 있는 재료를 쉼표로 구분하여 입력해주세요: ")
    user_ingredients = [ingredient.strip() for ingredient in input_ingredients.split(',')]

    if not input_ingredients.strip():
        print("적어도 하나의 재료를 입력해주세요.")
        continue

    # 가진 재료로 만들 수 있는 레시피 추천
    available_recipes = find_recipes(user_ingredients, recipes_module.recipes_db)
    if available_recipes:
        print("\n가지고 있는 재료로 만들 수 있는 레시피:")
        for idx, recipe in enumerate(available_recipes, 1):
            print(f" - {idx}. {recipe['name']}")
    else:
        print("\n가진 재료로 만들 수 있는 레시피가 없습니다.")
        continue
```

User Input for Ingredients:

The user is prompted to input a list of ingredients, separated by commas. The input is then processed into a list of stripped ingredients. If the input is empty, the user is prompted to enter at least one ingredient.

Recipe Recommendation:

The find_recipes function is called to recommend recipes based on the user's ingredients. If there are available recipes, they are displayed to the user; otherwise, a message is printed, and the loop continues.

```python
    while True:
        # 사용자가 만들고 싶은 레시피를 선택할 수 있도록 안내
        selected_recipe_idx = input("\n만들고 싶은 레시피 번호를 입력해주세요: ")

        try:
            selected_recipe_idx = int(selected_recipe_idx)
            if 1 <= selected_recipe_idx <= len(available_recipes):
                selected_recipe = available_recipes[selected_recipe_idx - 1]

                # 선택한 레시피가 있다면, 필요한 재료를 체크
                print(f"\n{selected_recipe['name']} 레시피에 필요한 재료:")
                missing_ingredients = []
                for ingredient in selected_recipe['ingredients']:
                    # 재료가 사용자가 가지고 있는 재료인지 확인
                    if ingredient in user_ingredients:
                        print(f" - {ingredient} (가지고 있음)")
                    else:
                        missing_ingredients.append(ingredient)
                        alternatives = suggest_alternatives(ingredient)
                        alternative_text = f" (대체재료: {', '.join(alternatives)})" if alternatives else ""
                        print(f" - {ingredient} (필요함){alternative_text}")

```

Recipe Selection and Ingredients Check:

The user is prompted to input the index of the recipe they want to make.

The input is validated, and if it falls within the valid range, the selected recipe and its required ingredients are displayed.

For each ingredient, it checks whether the user has it or not. If not, it suggests alternative ingredients.

```
                # 쇼핑 리스트를 생성
                shopping_list = create_shopping_list(selected_recipe, user_ingredients)
                if shopping_list:
                    print("\n쇼핑 리스트:")
                    for item in shopping_list:
                        print(f" - {item}")
                else:
                    print("\n추가로 구매할 재료가 없습니다.")

                # 사용자에게 쇼핑 리스트를 저장할지 질문
                save_option = input("쇼핑 리스트를 저장하시겠습니까? (yes/no): ")

                if save_option.lower() == 'yes':
                    filename = save_shopping_list(shopping_list)
                    print(f"쇼핑 리스트가 {filename}에 저장되었습니다.")

                # 레시피 지침을 출력하는 코드를 추가
                print_recipe_instructions(selected_recipe['name'], recipes_module.recipes_db)
                break
            else:
                print("올바른 범위의 숫자를 입력해주세요.")
        except ValueError:
            print("올바른 숫자를 입력해주세요.")
        except IndexError:
            print("선택한 레시피를 찾을 수 없습니다.")
    break
```

Shopping List Generation:

A shopping list is created using the create_shopping_listfunction, containing the ingredients the user is missing for the selected recipe.

Shopping List Display and Saving:

The shopping list is displayed to the user. If there are no additional items to purchase, a message is printed.The user is given the option to save the shopping list. If the user chooses to save, the save_shopping_listfunction is called.

Recipe Instructions:

The print_recipe_instructionsfunction is called to display instructions for the selected recipe.

Error Handling:

The code includes **try-except** blocks to handle errors such as invalid input for the selected recipe index, ensuring a smooth user experience.

[function.py]

```python
import recipes_module
# 사용자가 가진 재료를 기반으로 레시피를 찾는 함수
def find_recipes(user_ingredients, recipes_db):
    # 사용자가 가진 재료를 기반으로 레시피를 찾아 리스트로 반환합니다.
    recommended_recipes = []
    for recipe in recipes_db:
        if any(ingredient in user_ingredients for ingredient in recipe['ingredients']):
            recommended_recipes.append(recipe)  # 레시피 전체를 추가
    return recommended_recipes


# 대체 재료를 제안하는 함수
def suggest_alternatives(ingredient):
    return recipes_module.alternatives_db.get(ingredient, [])


# 쇼핑 리스트 만드는 함수
def create_shopping_list(selected_recipe, user_ingredients):
    shopping_list = [ingredient for ingredient in selected_recipe['ingredients']
                     if ingredient not in user_ingredients]
    return shopping_list

# 레시피 지침을 출력하는 함수
def print_recipe_instructions(recipe_name, recipes_db):
    recipe = next((r for r in recipes_db if r['name'] == recipe_name), None)
    if not recipe:
        print("레시피를 찾을 수 없습니다.")
        return

    print(f"\n{recipe['name']} 레시피 지침:")
    for instruction in recipe.get('instructions', []):
        print(instruction)
```

find_recipes Function:

It iterates through each recipe in the database (recipes_db) and checks if any ingredient in the recipe is present in the user's ingredients (user_ingredients).

If at least one ingredient matches, the entire recipe is added to the **recommended_recipes** list. The function returns a list of recommended recipes that the user can make based on their ingredients.

suggest_alternatives Function:

This function takes an **ingredient** as a parameter and suggests alternative ingredients from the **alternatives_db** in the **recipes_module.**

It uses the **get** method on **alternatives_db** to retrieve a list of alternative ingredients for the given ingredient. If no alternatives are found, it returns an empty list.

create_shopping_list Function:

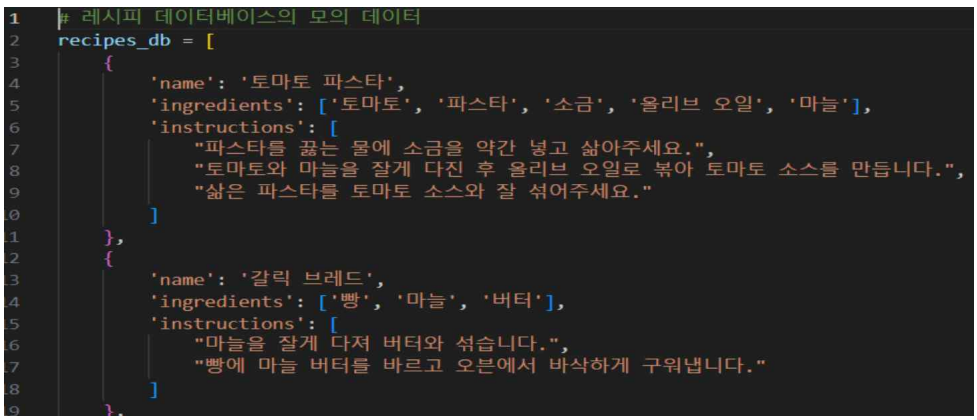
It creates a **shopping_list** by including ingredients from the selected recipe that are not already in the user's ingredients.

print_recipe_instructions Function:

It searches for the recipe with the specified name in the **recipes_db.**

If the recipe is found, it prints the recipe name and its instructions. If not, it prints a message indicating that the recipe cannot be found.

[recipes_module.py]



```
# 레시피 데이터베이스의 모의 데이터
recipes_db = [
    {
        'name': '토마토 파스타',
        'ingredients': ['토마토', '파스타', '소금', '올리브 오일', '마늘'],
        'instructions': [
            "파스타를 끓는 물에 소금을 약간 넣고 삶아주세요.",
            "토마토와 마늘을 잘게 다진 후 올리브 오일로 볶아 토마토 소스를 만듭니다.",
            "삶은 파스타를 토마토 소스와 잘 섞어주세요."
        ]
    },
    {
        'name': '갈릭 브레드',
        'ingredients': ['빵', '마늘', '버터'],
        'instructions': [
            "마늘을 잘게 다져 버터와 섞습니다.",
            "빵에 마늘 버터를 바르고 오븐에서 바삭하게 구워냅니다."
        ]
    },
```

```python
# 대체 재료 데이터베이스의 모의 데이터
alternatives_db = {
    '토마토': ['캔 토마토', '토마토 페이스트'],
    '파스타': ['스파게티', '페투치니'],
    '올리브 오일': ['카놀라 오일', '해바라기 오일'],
    '마늘': ['마늘 분말', '마늘 페이스트'],
    '빵': ['크루통', '바게트'],
    '버터': ['마가린', '식물성 버터'],
    '계란': ['두부'],
    '간장': ['타마리', '코코넛 아미노스', '우스터 소스', '소야 소스'],
    '참기름': ['들기름', '호두 오일'],
    '김': ['노리', '해초 샐러드'],
    '통깨': ['깨소금', '참깨'],
    '크림': ['코코넛 밀크', '캐시넛 크림'],
    '파마산 치즈': ['베지테리언 파마산'],
    '후추': ['흰 후추', '카옌 페퍼'],
    '돼지고기': ['닭고기', '쇠고기'],
    '두부': ['치즈', '유부'],
```

The **recipes_db** contains mock data representing various recipes. Each recipe is a dictionary with keys such as 'name', 'ingredients', and 'instructions'. It includes information about the recipe name, required ingredients, and cooking instructions.

The **alternatives_db** contains mock data for suggesting alternative ingredients. It associates various ingredients with alternative options.

[shopping_list_manager.py]

```python
# shopping_list_manager.py
def save_shopping_list(shopping_list, filename="shopping_list.txt"):
    with open(filename, "w") as file:
        for item in shopping_list:
            file.write(f"{item}\n")
    return filename
```

This function takes a list of items (shopping_list) and saves it to a text file.

4. Test Result

(1)Ingredient-Based Recipe Recommendation Function

Description:

This function takes the input of ingredients from the user and recommends possiblerecipes from the **recipe_db** based on the entered ingredients.

Alternative Ingredient Suggestions: If some ingredients do not exactly match the requirements of the recipe, the system proposes alternative ingredients.

(2)Shopping List Generation-Manages the additional required ingredients as a

shopping list

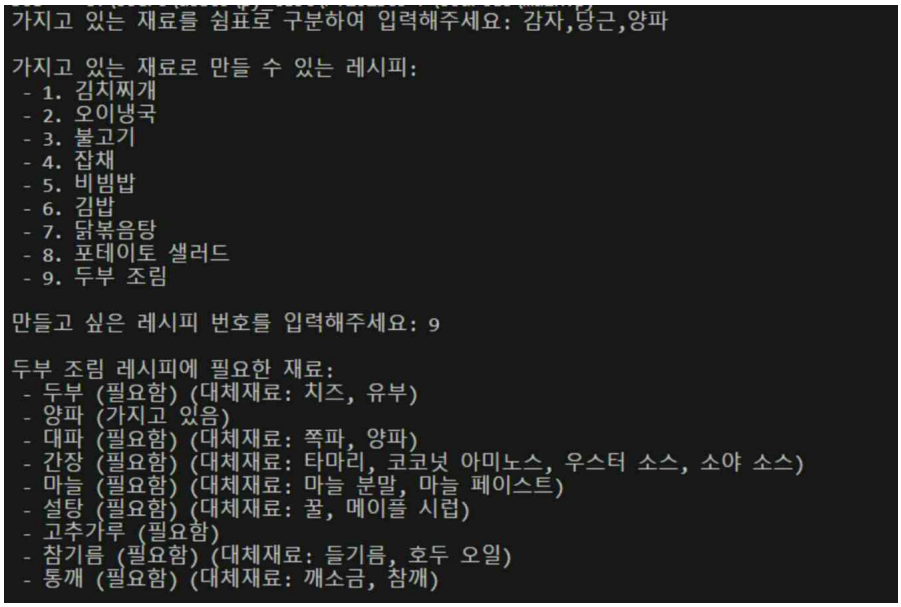
Description:

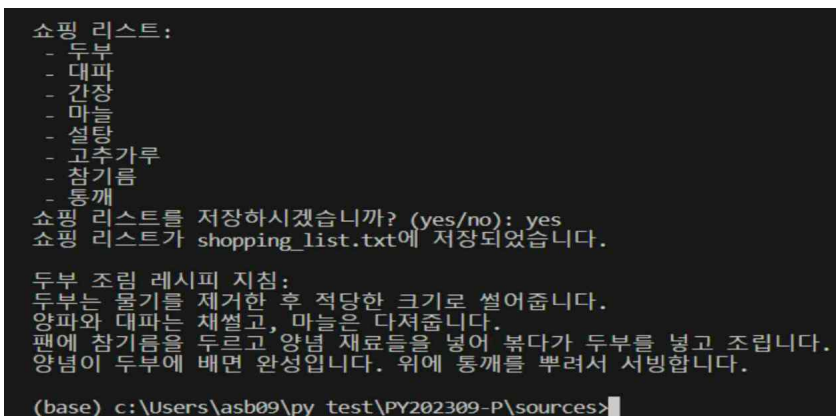
It identifies ingredients that the user does not have and asks whether to save the missing items as a shopping list.

+)

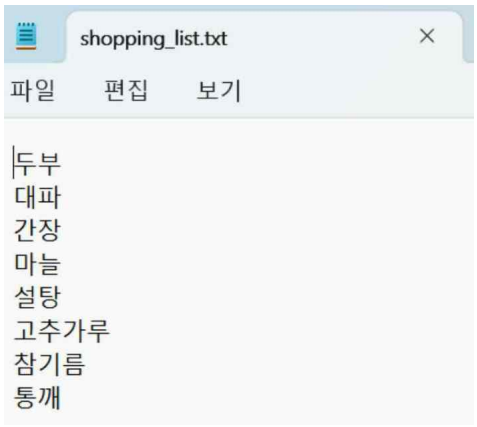
The recipe recommendation process has been simplified by assigning numbers to each recipe, allowing users to make selections using numerical input rather than entering names

[Test results screenshot]

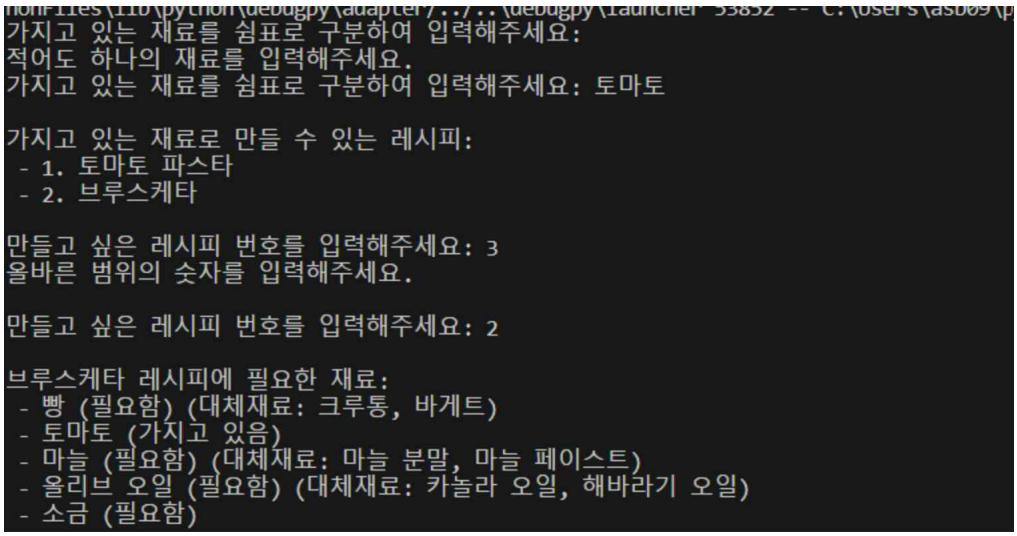




When users input the ingredients they have, the system presents recipes that can be made with those ingredients. Upon entering the desired recipe number, the system informs the user of the necessary ingredients and compiles a shopping list for any missing items, then inquires if the user wishes to save this list. If saved, it is stored in the shopping list.



[To enhance the overall robustness of the program, I identified and strengthened areas prone to errors.]



-When inputting material, if it is empty, output it to input at least one

-If the recipe number is entered incorrectly, it will be re-printed again.

5. Changes in Comparison to the Plan

1) Recommendations for Soon-to-Expire Foods

Before:

Ingredient Optimization: The program assists in the optimal utilization of refrigerator contents. By suggesting recipes that prioritize ingredients nearing their expiration, it helps reduce food waste and maintain the freshness of groceries.

After:

I removed that functionality as it was not fully implemented.

Reason:

In reconsideration of the program's scope and goals, I've opted to shift my focus away from the complex task of assigning expiration dates to grocery items. This decision stems from a desire to deliver a more streamlined and user-friendly program within the available timeframe. While the original idea was intriguing, practical constraints and the need for a more feasible project led me to reprioritize and emphasize refining the existing features for a more polished user experience.

Additionally, I aim to enhance the central functionality that was initially envisioned, striving to elevate the overall completion of the main features. This adjustment allows for a more comprehensive and effective utilization of the program's core capabilities, ensuring a higher level of user satisfaction.

6. Lessons Learned & Feedback

During the class, it was great to have the opportunity to practice the concepts demonstrated by the professor through hands-on exercises. It was beneficial not only for Python but also for getting a glimpse into machine learning and data-related topics, sparking my interest in those areas. However, the course seemed a bit challenging for a first-year student.