Python programming

Savvy plate

Progress Report : 1

Date : 2023/11/25
Name : 안수빈
ID :214309

1. Introduction (16 pt)

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 (14 pt)

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.

- Ingredient optimization: The program helps to utilize the ingredients in the refrigerator optimally. By suggesting recipes that prioritize ingredients nearing their expiration date, it reduces food waste and helps keep ingredients fresh.

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. Progress

1) Implementation of features

(1) Implemented Feature Name: Ingredient-Based Recipe Recommendation and Shopping List Management

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:

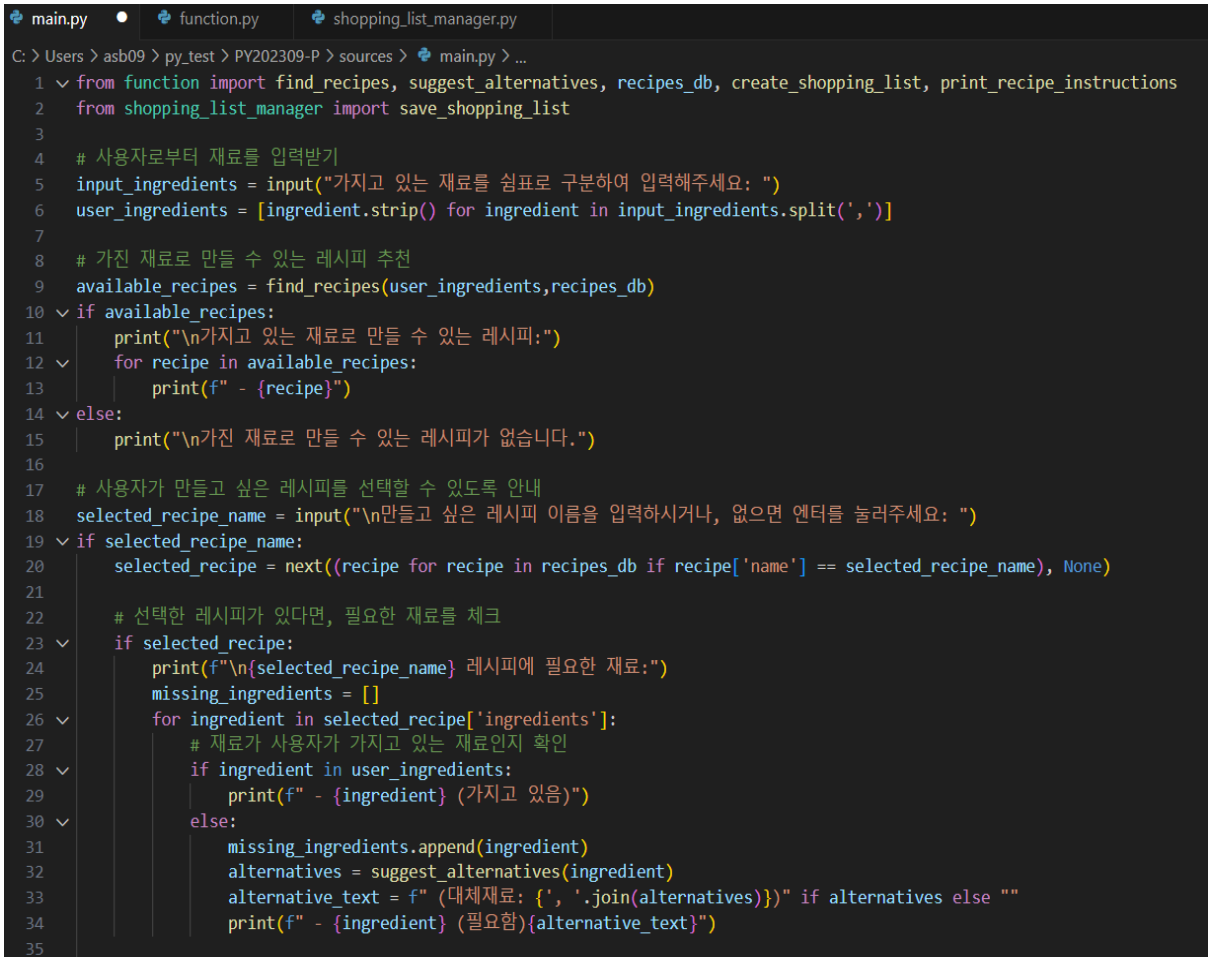
- **Loops**: The `for` loop is utilized to iterate through all recipes in the `recipes_db` and compare them with the user's ingredients.
- **Conditional Statements:** The `if` statement is employed to check if the user's ingredients match the ingredients of the recipe and to provide appropriate messages.
- **Functions:** Functions such as `find_recipes`, `suggest_alternatives`, `create_shopping_list`, and `print_recipe_instructions` have been defined to modularize the code and enhance reusability.
- **Modules:** The code has been organized into separate files like `main.py`, `function.py`, and `shopping_list_manager.py`, modularizing the code by

functionality.

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

Code screenshot

[main.py]



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

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

# 가진 재료로 만들 수 있는 레시피 추천
available_recipes = find_recipes(user_ingredients,recipes_db)
if available_recipes:
    print("\n가지고 있는 재료로 만들 수 있는 레시피:")
    for recipe in available_recipes:
        print(f" - {recipe}")
else:
    print("\n가진 재료로 만들 수 있는 레시피가 없습니다.")

# 사용자가 만들고 싶은 레시피를 선택할 수 있도록 안내
selected_recipe_name = input("\n만들고 싶은 레시피 이름을 입력하시거나, 없으면 엔터를 눌러주세요: ")
if selected_recipe_name:
    selected_recipe = next((recipe for recipe in recipes_db if recipe['name'] == selected_recipe_name), None)

    # 선택한 레시피가 있다면, 필요한 재료를 체크
    if selected_recipe:
        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}")
```

```python
    # 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
    if shopping_list:
        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_db)
else:
    print("선택한 레시피를 찾을 수 없습니다.")

```

[function.py]

main.py ● | function.py ✕ | shopping_list_manager.py

C: > Users > asb09 > py_test > PY202309-P > sources > function.py > ...

```python
# 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."
        ]
    },
    {
        'name': '브루스케타',
        'ingredients': ['빵', '토마토', '마늘', '올리브 오일', '소금', '바질', '페퍼'],
        'instructions': [
            "토마토는 주사위 모양으로 잘게 썰고, 마늘은 다져 준비합니다.",
            "빵을 오븐에서 바삭하게 구운 후 마늘을 문지릅니다.",
            "구운 빵 위에 토마토를 올리고 올리브 오일, 소금, 바질, 후추로 양념합니다."
        ]
    },
    {
```

```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'],
    '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'],
    # ... 더 많은 대체 재료들
}

```

```python

# 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['name'])
    return recommended_recipes

# 대체 재료를 제안하는 함수
def suggest_alternatives(ingredient):
    return 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)
```

[shopping_list_manager.py]

main.py ● function.py shopping_list_manager.py ×

C: > Users > asb09 > py_test > PY202309-P > sources > 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

```

2) Test results

(1) Ingredient-Based Recipe Recommendation Function

Description:

This function takes the input of ingredients from the user and recommends possible

recipes from the **`recipe_db`** based on the entered ingredients.

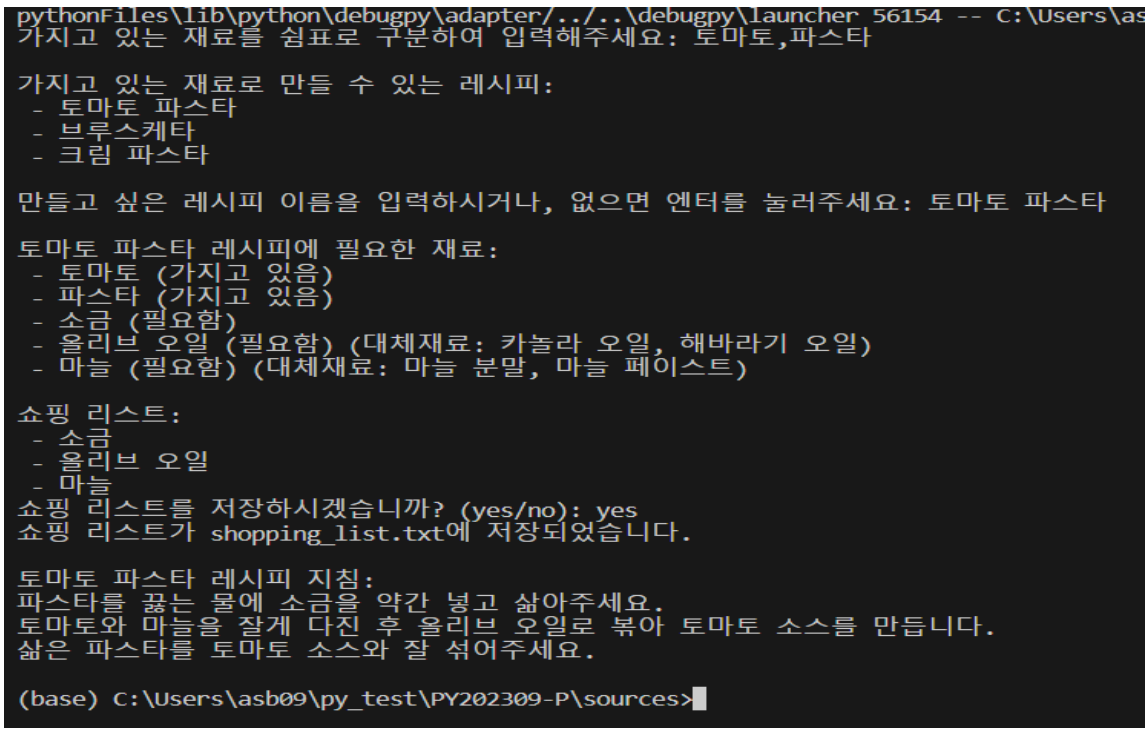
Alternative Ingredient Suggestions: If some ingredients do not exactly match the requirements of the recipe, the system proposes alternative ingredients.

(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.

[Test results screenshot]



4. Changes in Comparison to the Plan

1) Change title

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:

Even if not by the exact date of manufacture, assign an order to indicate which items should be consumed soon.

Reason:

I initially wanted to differentiate my program by including an ingredient optimization feature that would assign expiration dates to each grocery item and suggest recipes based on items nearing expiry. However, my current skill level is not sufficient to develop such a program. Instead, I'd like to focus on refining the other features to ensure they are programmed with higher quality. I want to address potential issues that may arise with these other functionalities. Additionally, I aim to review what I've learned in class and use that knowledge as a foundation to develop a more cohesive and complete codebase.

5.Schedule