Python Data Organization Project

Recipe Finder System

Project Description:

This project helps you practice your Python skills by creating a user-friendly Recipe Finder System. The system enables users to explore recipes based on their preferences, search for dishes, filter by specific criteria, and contribute their own recipes to the database.

Functionality:

Recipe Data:

Use a dictionary to store all recipe information. Each key is a unique recipe name (string), and its value is another dictionary containing attributes such as:

- Cuisine (string, e.g., 'Italian', 'Mexican')
- Ingredients (list of strings)
- Prep Time (integer, in minutes)
- Difficulty (string: 'Easy', 'Medium', 'Hard')
- Rating (float, 1 to 5)

Recommendations:

- Ask the user to enter their preferred cuisine.
- Search the recipe data for matching cuisine.
- If matches exist, list relevant recipes.
- Otherwise, display a helpful message suggesting another input.
- Example Output:
  Based on your love for [cuisine], try these recipes: [list of recipes].

Find a Recipe:

- Enable users to search for recipes by full or partial name match.
- If found, show detailed recipe information including ingredients, prep time, difficulty, and rating.
- Use a loop to allow multiple searches or option to exit.

Example Output:
Recipe: Spaghetti Carbonara
Cuisine: Italian
Ingredients: Pasta, Eggs, Cheese, Bacon
Prep Time: 20 mins
Difficulty: Medium
Rating: 4.5

Filter Recipes:

Let users filter recipes by:
- Difficulty (Easy/Medium/Hard)
- Prep Time (e.g., under 30 minutes)
- Rating (e.g., 4.0+)

Display results in a clean, formatted list.

Add a New Recipe:

- Enable users to input new recipe details.
- Use None for any missing fields (like rating).
- Add the new recipe to the existing dictionary.

Deliverables:

- Complete Python script implementing the recipe finder.
- A sample interactive session showing user engagement with all features.

Tips:

- Start with a manageable recipe dataset (5-10 entries).
- Use distinct functions for each system feature.
- Include clear comments and variable names.
- Account for edge cases like empty input or unknown cuisines.

Bonus (2 points):

- Bonus points if you create a simple user interface (UI) using a Python library such as Tkinter or PySimpleGUI.

Sample Recipe Data:

```
recipe_data = {
  'Spaghetti Carbonara': {
    'Cuisine': 'Italian',
    'Ingredients': ['Pasta', 'Eggs', 'Cheese', 'Bacon'],
    'Prep Time': 20,
    'Difficulty': 'Medium',
    'Rating': 4.5
  },
  'Chicken Tikka Masala': {
    'Cuisine': 'Indian',
    'Ingredients': ['Chicken', 'Yogurt', 'Spices', 'Tomato Sauce'],
    'Prep Time': 45,
    'Difficulty': 'Hard',
    'Rating': 4.8
  },
  'Avocado Toast': {
    'Cuisine': 'American',
```

```
        'Ingredients': ['Bread', 'Avocado', 'Salt', 'Pepper'],
        'Prep Time': 5,
        'Difficulty': 'Easy',
        'Rating': 3.7
    }
}
```

Example Output:

```
Welcome to the Recipe Finder!
Choose an option:
1. Get recipe recommendations
2. Search for a recipe
3. Filter recipes (by time/difficulty/rating)
4. Add a new recipe
5. Exit

> 1
Enter your preferred cuisine: Italian
Based on your love for Italian, try these recipes:
- Spaghetti Carbonara
```