r/ThingsYouWillNeed • 17 days ago
TheStocksGuy

Random Meal Generator with Smart Logging in Python

Random Meal Generator with Smart Logging in Python

I recently developed a Python script that generates random meal combinations while logging the choices made to avoid repeating sides and pairings on the same day. This ensures a fresh dining experience every time!

Key Features:

- **Random Meal Selection**: Choose from a list of main foods, sides, and additional pairings.
- **Smart Logging**: Uses a JSON file to log meals selected, avoiding repeated sides and pairings within the same day.
- **Flexible Customization**: Easily add more choices to the main foods, sides, and pairings.

How It Works:

1. **Random Selection**: The script randomly selects a main food, side, and pairing.
2. **Logging**: It logs the selected meal to a JSON file with a timestamp.
3. **Avoid Repeats**: When generating a meal on the same day, the script checks the log to avoid repeating sides and pairings.

Adding More Choices:

To add more choices, simply update the lists and dictionaries in the script:

- **Main Foods**: Add more items to the `main_foods` list.
- **Sides and Pairings**: Update the `sides` and `additional_pairings` dictionaries with new options.

JSON Logging:

The script uses a JSON file to store the log:

- **Load Log**: Reads the log file at the start.
- **Save Log**: Writes the updated log to the file after generating a meal.
- **Avoid Repeats**: Uses the log to check previous choices for the same day and avoid repeating them.

Here's a snippet of the main logic (simplified for readability):

```
import random
import json
from datetime import datetime
```

```
# Sides
sides = {
    "Burger": ["Fries", "Onion Rings", "Salad"],
    "Chicken": ["Mashed Potatoes", "Coleslaw", "Biscuit"],
    "Steak": ["Baked Potato", "Steamed Vegetables", "Garlic Bread"],
    "Sushi": ["Miso Soup", "Edamame", "Seaweed Salad"],
    "Chow Mein": ["Spring Rolls", "Dumplings", "Wonton Soup"],
    "Fried Rice": ["Egg Roll", "Sweet and Sour Pork", "Hot and Sour Soup"]
}

# Additional pairings
additional_pairings = {
    "Burger": ["Soda", "Milkshake"],
    "Chicken": ["Lemonade", "Iced Tea"],
    "Steak": ["Red Wine", "Beer"],
    "Sushi": ["Green Tea", "Sake"],
    "Chow Mein": ["Hot Tea", "Plum Wine"],
    "Fried Rice": ["Bubble Tea", "Soft Drink"]
}

log_file = "meal_log.json"

def load_log():
    try:
        with open(log_file, 'r') as file:
            return json.load(file)
    except FileNotFoundError:
        return {}

def save_log(log):
    with open(log_file, 'w') as file:
        json.dump(log, file, indent=4)

def create_log_file_if_not_exists():
    try:
        with open(log_file, 'x') as file:
            json.dump({}, file)
    except FileExistsError:
        pass

def get_random_meal():
    create_log_file_if_not_exists()

    log = load_log()
    today = datetime.today().strftime('%Y-%m-%d')
```

```
        log[today] = []

    previous_sides = [entry["side"] for entry in log[today]]
    previous_pairings = [entry["pairing"] for entry in log[today]]

    main_food = random.choice(main_foods)
    available_sides = [side for side in sides[main_food] if side not in previous_sides]
    available_pairings = [pairing for pairing in additional_pairings[main_food] if pairing not in

    if not available_sides or not available_pairings:
        return None, None, None

    side = random.choice(available_sides)
    pairing = random.choice(available_pairings)

    log[today].append({"main_food": main_food, "side": side, "pairing": pairing, "time": time_stam
    save_log(log)

    return main_food, side, pairing

def display_log():
    log = load_log()
    for date, entries in log.items():
        print(f"Date: {date}")
        for entry in entries:
            print(f"  Time: {entry['time']}, Main Food: {entry['main_food']}, Side: {entry['side']

# Generate and print a random meal, ensuring no repeat sides or pairings on the same day
main_food, side, pairing = get_random_meal()
if main_food and side and pairing:
    print(f"Main Food: {main_food}")
    print(f"Side: {side}")
    print(f"Pairing: {pairing}")
else:
    print("No available combinations that avoid repeats found for today.")

# Display the logged results with timestamps
display_log()
```

Console Log - JSON File Storage (Print results Example:

> Main Food: Chow Mein Side: Wonton Soup Pairing: Hot Tea Date: 2024-12-30 Time: 16:49:26, Main Food: Sushi, Side: Seaweed Salad, Pairing: Green Tea Time: 16:49:32, Main Food: Fried Rice, Side: Sweet and Sour