<재료 소진 경고>

def predict\_ingredient\_depletion(self):

        # 현재 주문된 메뉴들의 재료 소진을 예측하는 함수

        # DP 배열 초기화

        dp = [float('inf')] \* (len(self.ingredients) + 1)

        dp[0] = 0

        for order in self.total\_orders:

            menu\_ingredients = self.menu[order[1]]['재료']

            for ingredient in menu\_ingredients:

                dp[len(menu\_ingredients)] = min(dp[len(menu\_ingredients)], self.ingredients[ingredient])

        # 예측 결과 출력

        min\_ingredient\_count = min(dp[1:])  # 0번째는 무시하고 최소값 찾기

        depleted\_ingredients = [ingredient for ingredient, count in self.ingredients.items() if count == min\_ingredient\_count]

        prediction\_text = f"!!재료소진경고!!\n {', '.join(depleted\_ingredients)} 재고를 확인해주세요"

        return prediction\_text

// dp알고리즘을 사용하는 것이 아닌 재료들 중 가장 작은 값(min)을 반환해서 재료 소진 경고를

메시지를 나타내는 코드입니다(오해한 부분)

<재료 리필 권고>

def check\_depleted\_ingredient\_warning(self):

        # 각 손님까지의 주문에서 사용된 단일 재료의 카운트를 기록하는 리스트 초기화

        ingredient\_counts = {ingredient: 0 for ingredient in self.ingredients}

        # 각 주문에서 사용된 재료의 카운트 기록

        for order in self.total\_orders:

            \_, pizza, \_ = order

            for ingredient in self.menu[pizza]['재료']:

                ingredient\_counts[ingredient] += 1

        # 현재까지의 주문에서 사용된 재료 카운트 출력

        print("현재까지의 주문에서 사용된 재료 카운트:")

        for ingredient, count in ingredient\_counts.items():

            print(f"{ingredient}: {count}")

        # 최대로 소진된 재료 선택

        most\_depleted\_ingredient = max(ingredient\_counts, key=ingredient\_counts.get)

        count\_of\_most\_depleted\_ingredient = ingredient\_counts[most\_depleted\_ingredient]

        def fib(n, memo={}):

            if n <= 1:

                return n

            if n not in memo:

                memo[n] = fib(n - 1, memo) + fib(n - 2, memo)

            return memo[n]

        # 피보나치 수열 값 계산

        fibonacci\_values = [0, 1]

        n = 2

        while True:

            value = fib(n)

            if value > count\_of\_most\_depleted\_ingredient:

                break

            fibonacci\_values.append(value)

            n += 1

        # 최대로 소진된 재료의 사용 횟수가 피보나치 수열 값(n-1)보다 큰 경우 경고 출력

        warning\_message = ""

        if count\_of\_most\_depleted\_ingredient > fibonacci\_values[n - 1]:

            warning\_message = f"{most\_depleted\_ingredient}(이)가 자주 사용되어 자주 리필해놓으셔야 될 것 같습니다."

        # 피보나치 수열 값보다 작은 경우 리필하면 안될 것 같다는 메시지 출력

        if not warning\_message:

            warning\_message = f"재료들을 자주 리필해놓으실 필요는 없을 것 같습니다."

        return warning\_message

//DP알고리즘이 사용된 재료 소진 권고 메시지 나타내 주는 코드 가장 많이 사용된 재료를 카운트 해서 그 카운트 된 단일 재료의 숫자가 Fibonacci(n-1)보다 클 경우 (여기서 n은 주문한 사람 수 n임) 재료가 그만큼 자주 사용된다는 것이기 때문에 재료들을 자주 리필해놓으셔야 한다는 권고 메시지가 콘솔창에 뜨게 됩니다 => 여기서 왜 재료 소진 경고 메시지는 프로그램 창에 띄우고 재료 리필 권고 메시지는 콘솔창에 띄우게 되냐면 재료 소진은 그 자리에서 바로 채워야 하는 더 중요한 메시지 이기 때문에 프로그램에 띄우게 설정해뒀고, 리필 권고는 말 그대로 권고이기 때문에 콘솔창에 띄워도 될 것 같다고 생각해 봤어요, DP알고리즘이 적용된 부분은 Fibonacci 수열을 계산하는 부분입니다 => 단순히 n으로 비교하는 것이 아닌 Fibonacci(n-1)이랑 비교한 이유는 주문한 사람의 숫자가 늘어날수록 더 fib값처럼 늘어나는 숫자가 더 커질 것이라 생각했고 약간의 조정이 필요하다 생각되어 fib(n-1)의 값을 집어넣게 되었습니다.