Project Idea: Car Rental Contract Solver Using Python

Overview

This project addresses a **Constraint Satisfaction Problem (CSP)** for assigning car rental contracts to five customers while ensuring specific constraints are met (duration, location, and car brand). Using the python-constraint library, it ensures each customer is assigned a unique contract that satisfies the provided conditions.

Problem Description

You are tasked with assigning a rental car contract to each of the following customers:

- Freda
- Vicky
- Opal
- Sarah
- Penny

Each customer rents a car under a contract defined by:

- **Duration**: Number of days for the rental (2, 3, 4, 5, or 6 days)
- **Location**: The rental's city (Brownfield, Los Altos, Iowa Falls, Redding, Durham)
- **Car Brand**: The car brand (Dodge, Fiat, Hyundai, Jeep, Nissan)

Constraints

1. **Unique Locations and Brands**: No two customers can rent cars from the same location or drive the same car brand.

2. **Specific Hints**:
    - Vicky cannot rent in Los Altos or Durham, nor can she drive a Fiat.
    - No Jeep contract is in Iowa Falls, and no Jeep rental is for 6 days.
    - Vicky rents a Nissan in either Los Altos or Redding.
    - The contract in Iowa Falls is exactly for 5 days.
    - The contract in Durham is 3 days longer than Opal's contract.
    - Freda's contract is either a 2-day rental not in Redding or Vicky's Nissan rental in Redding.
    - Opal's contract is 1 day longer than the Hyundai contract.

How It Works

The program operates by:

1. **Defining Variables**: Each customer is assigned a set of possible rental contracts (combinations of duration, location, and car brand).

2. **Adding Constraints**: Constraints are added to ensure that all conditions, such as uniqueness of locations and brands and the specific hints provided, are met.
3. **Solving the Problem**: The program uses a backtracking algorithm to search through all possible combinations of contracts, filtering out those that violate the constraints.
4. **Validating Solutions**: A custom function checks each solution to see if it satisfies all the hints.

The output displays the total number of valid solutions and lists the top five solutions in detail, showing the contract assigned to each customer.

Algorithm Used

The program uses a **Backtracking Algorithm**, which is a depth-first search algorithm used for solving CSPs. The backtracking algorithm systematically tries out possible solutions, backtracking when a constraint is violated, and continues to explore other options until all possible solutions are found.

- **Domain filtering**: Each customer is assigned a list of possible rental contracts (combinations of trip duration, location, and car brand).
- **Constraint propagation**: Constraints are added to reduce the domain of possible values, preventing conflicts like two customers renting the same car brand.
- **Backtracking**: The algorithm attempts to assign values to customers while backtracking when a conflict arises, ensuring all possible valid solutions are explored.

Customization

You can customize this program by modifying:

1. **Customers, Locations, and Brands**:
    - Add more customers or change the available locations and car brands.
    - For example, add new locations or modify existing ones:

        customer_name = ['Freda', 'Vicky', 'Opal', 'Sarah', 'Penny', 'Alex'] # Adding new customer 'Alex'

        available_locations = ['Brownfield', 'Los Altos', 'Iowa Falls', 'Redding', 'Durham', 'Newport'] # Adding 'Newport'

2. **Modify Constraints**:
    - You can add or modify constraints based on new requirements. For example, to make Opal's rental contract exactly 4 days:

        problem.addConstraint(lambda o: o[0] == 4, ('Opal',))

3. **Display Options**:
    - Change how many solutions to display by modifying the number of top solutions printed.
    - Adjust the formatting to display additional information or fewer details.

How to Add New Constraints

To introduce new rules, simply add them using lambda functions. For example:

- If you want to make Sarah's rental for exactly 5 days:

problem.addConstraint(lambda s: s[0] == 5, ('Sarah',))

Solution Overview

The program calculates all valid solutions that meet the defined constraints. It first generates all possible contract combinations, filters out invalid ones, and then displays the top solutions.

Key Functionality

- **Total valid solutions**: The program computes the total number of valid solutions and displays this count.
- **Top 5 solutions**: If there are valid solutions, the top 5 are printed, showing each customer's assigned rental contract, including the number of days, location, and car brand.

Screenshots

```
Total number of valid solutions found for the problem: 40

Top Valid Solution 1:
Customer Name: Vicky ----> Duration: 6 days, Location: Redding, Car Brand: Nissan
Customer Name: Freda ----> Duration: 5 days, Location: Iowa Falls, Car Brand: Fiat
Customer Name: Opal ----> Duration: 3 days, Location: Los Altos, Car Brand: Jeep
Customer Name: Sarah ----> Duration: 6 days, Location: Durham, Car Brand: Dodge
Customer Name: Penny ----> Duration: 2 days, Location: Brownfield, Car Brand: Hyundai

Top Valid Solution 2:
Customer Name: Vicky ----> Duration: 6 days, Location: Redding, Car Brand: Nissan
Customer Name: Freda ----> Duration: 5 days, Location: Iowa Falls, Car Brand: Fiat
Customer Name: Opal ----> Duration: 3 days, Location: Brownfield, Car Brand: Jeep
Customer Name: Sarah ----> Duration: 6 days, Location: Durham, Car Brand: Dodge
Customer Name: Penny ----> Duration: 2 days, Location: Los Altos, Car Brand: Hyundai

Top Valid Solution 3:
Customer Name: Vicky ----> Duration: 6 days, Location: Redding, Car Brand: Nissan
Customer Name: Freda ----> Duration: 5 days, Location: Iowa Falls, Car Brand: Dodge
Customer Name: Opal ----> Duration: 3 days, Location: Los Altos, Car Brand: Jeep
Customer Name: Sarah ----> Duration: 6 days, Location: Durham, Car Brand: Fiat
Customer Name: Penny ----> Duration: 2 days, Location: Brownfield, Car Brand: Hyundai

Top Valid Solution 4:
Customer Name: Vicky ----> Duration: 6 days, Location: Redding, Car Brand: Nissan
Customer Name: Freda ----> Duration: 5 days, Location: Iowa Falls, Car Brand: Dodge
Customer Name: Opal ----> Duration: 3 days, Location: Brownfield, Car Brand: Jeep
Customer Name: Sarah ----> Duration: 6 days, Location: Durham, Car Brand: Fiat
Customer Name: Penny ----> Duration: 2 days, Location: Los Altos, Car Brand: Hyundai

Top Valid Solution 5:
Customer Name: Vicky ----> Duration: 6 days, Location: Redding, Car Brand: Nissan
Customer Name: Freda ----> Duration: 2 days, Location: Los Altos, Car Brand: Hyundai
Customer Name: Opal ----> Duration: 3 days, Location: Brownfield, Car Brand: Jeep
Customer Name: Sarah ----> Duration: 6 days, Location: Durham, Car Brand: Fiat
Customer Name: Penny ----> Duration: 5 days, Location: Iowa Falls, Car Brand: Dodge
```

The program found 40 valid solutions to the problem, and it is showing a few of the top valid solutions. Each solution specifies which customer is assigned a car for how long, at which location, and what brand they rented.

Each solution presents a different combination of valid assignments that satisfy the given constraints.