Instructions to run and compile:
Extract all .py files in the same folder. Then run “python main.py”
It runs for all 4 cases : jobs puzzle with and without mrv & houses puzzle with and without mrv.
When a result is found, it asks if you want to search the rest of the domain for more results . Entering 'y' would continue searching while anything else as input would end that particular part and print its result.

=> The houses puzzle takes some time to find a result (15-20 seconds)

Jobs puzzle:

4 states are defined i.e. for 4 people and each of them is tried to assign the possible combination of jobs, which is the domain of assignment (all possible sets of different jobs of length 2). Then from domain we try to assign a set of jobs to person1 then if valid job is assigned we move to person 2 and similarly for other and follow backtracking if we do not get a suitable combination moving forward

persons = [["roberta","female"],["thelma","female"],["steve","male"],["pete","male"]]
jobs = ["chef","guard","nurse","clerk","police","teacher","actor","boxer"]
Each person is assigned an attribute as gender and education(which is defined < 9 for pete and None for others) . The job list is used to generate all possible combinations in the domain.

To implement MRV feature, before moving on to next state for assigning the values from domain we calculate all possible moves and the size of domain possible for each and then choose the one with minimum possible domain size. To calculate such state first the non assigned states are calculated then for each of them possible valid domain size is calculated by calling consistency check function.
The consistency function is defined such that it covers all the constraints mentioned in the problem statement by using a bunch of if conditions.

Houses puzzle:
5 states are defined i.e. the color of the house and each of them is assigned. Then to each state we try to assign the possible values from the domain. The domain in this case is a set of country of citizenship, food, drink, pet and house number i.e. all possible combinations of all possible values of these sets of values. To each color of house a combination from domain is tried to assigned and then we move to the next house color and try to assign it a possible combination of values from the domain.

house_colr = ["green","ivory","red","blue","yellow"]
citizen = ["norwegian","japanese","english","spaniard","ukranian"]
food = ["kitkat","milky ways","snickers","hersheys","smarties"]
drink = ["water","milk","orange juice","coffee","tea"]
pets = ["zebra","horse","fox","dog","snails"]
These are initial variables, the house color is used as a state to which all possible combinations of rest of variables along with house number (1 to 5) are assigned.

To implement MRV feature, before moving on to next state for assigning the values from domain we calculate all possible moves and the size of domain possible for each and then choose the one with minimum possible domain size. To calculate such state first the non assigned states are calculated then for each of them possible valid domain size is calculated by calling consistency check function.
The consistency function is defined such that it covers all the constraints mentioned in the problem statement by using a bunch of if conditions.

Note : The number of states searched depends on initial assignment of values and how close the initial assignment is to the target value.

Results:

Jobs puzzle:

States searched : 138 with MRV and 2910 without MRV

pete [('clerk', 'actor')]
steve [('nurse', 'police')]
thelma [('chef', 'boxer')]
roberta [('guard', 'teacher')]

Houses puzzle:

States searched : 179690 with MRV and 8320315 without MRV

yellow [('norwegian', 'kitkat', 'water', 'fox', 1)]
blue [('ukranian', 'hersheys', 'tea', 'horse', 2)]
red [('english', 'smarties', 'milk', 'snails', 3)]
ivory [('spaniard', 'snickers', 'orange juice', 'dog', 4)]
green [('japanese', 'milky ways', 'coffee', 'zebra', 5)]