The Problem

The rules of the game show are as follows:

After you have chosen a door, the door remains closed for the time being. The game show host, Monty Hall, who knows what is behind the doors, now has to open one of the two remaining doors, and the door he opens must have a goat behind it. If both remaining doors have goats behind them, he chooses one randomly.

After Monty Hall opens a door with a goat, he will ask you to decide whether you want to stay with your first choice or to switch to the last remaining door.

Do you stick with your first choice or switch to the last unopened door?

The Output

The function which carries out how_many simulations

```python
import random

def main():
    algorithm_type = 1
    how_many = 1000
    filename = 'MontyAlgorithmType1.txt'
    algorithm_desc = 'Algorithm 1: contestant always chooses other door.'
    cars_won, results_str = run_simulations(how_many, algorithm_type)
    write_simulation_results_to_file(filename, algorithm_desc, cars_won, how_many, results_str)

    algorithm_type = 2
    filename = 'MontyAlgorithmType2.txt'
    algorithm_desc = 'Algorithm 2: contestant always sticks to the original choice.'
    cars_won, results_str = run_simulations(how_many, algorithm_type)
    write_simulation_results_to_file(filename, algorithm_desc, cars_won, how_many, results_str)
```

The function which carries out how_many simulations

```python
def run_simulations(how_many, algorithm_type):
    result_str = ''
    number_of_car_wins = 0
    for count in range(1, how_many + 1):
        has_won_car, result_info = run_one_simulation(algorithm_type)
        result_str = result_str + str(count) + '.
        result_str = result_str + result_info
        if has_won_car:
            number_of_car_wins = number_of_car_wins + 1
    return (number_of_car_wins, result_str)
```
def get_random_combination():
    possible_prize = ['goat', 'goat', 'car']
    possible_indices = ['012'
        combination = []
        while len(possible_indices) > 0:
            random_pos = random.randrange(len(possible_indices))
            door_index = int(possible_indices[random_pos])
            combination.append(possible_prize[door_index])
            possible_indices = possible_indices[:random_pos] + possible_indices[random_pos + 1:]
        return combination

def get_results(doors, monty_index, contestant_old_index, contestant_final_index):
    result_info += ('Contestant original choice: ' + doors[contestant_old_index] + '\n'
        monty_index = random.randrange(len(indices_remaining))
    result_info += ('Monty choice: ' + doors[monty_index] + '\n'
        contestant_final_index = contestant_final_index + 1:
    contestant_final_index = other_door_index
    contestant_old_index = contestant_index
    if algorithm_type == 1:
        contestant_final_index = other_door_index
    else:
        contestant_final_index = contestant_old_index
    has_won_car, result_info = get_results(doors, monty_door_index, contestant_old_index, contestant_final_index)
    return (has_won_car, result_info)

def write_simulation_results_to_file(filename, algorithm_desc, cars_won, how_many, results_str):
    file_out = open(filename, "w")
    results_str += '%
    file_out.write("Carried out " + str(how_many) + " simulations\n")
    file_out.write("Number of cars won: " + str(cars_won) + "\n")
    file_out.write("Number of goats won: " + str(how_many - cars_won) + "\n")
    file_out.write("Percentage of wins: " + str(percentage_car_wins) + "%\n")
    file_out.write("%\n")
    file_out.write(results_str)
    file_out.close()