The Problem

Suppose you’re on a game show and you’re given the choice of three doors. Behind one door is a car, behind the others, goats. The car and the goats were placed randomly placed behind the doors before the show.

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.

The question is:

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

The Problem

Is it to your advantage to change your choice?

We will run the following two sets of simulations many times and check the results. In one set of simulations:

- the contestant sticks to the original choice, and, in the other:
- the contestant always chooses to switch to the last remaining door.

The Problem Output

- After carrying out the simulation, the results of the two simulations are written to the files, "MontyAlgorithmType1.txt" and "MontyAlgorithmType2.txt".

- How many simulations of each algorithm will be run?

- What results do we want?

- What format?

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of simulations</th>
<th>Number of cars won</th>
<th>Number of goats won</th>
<th>Percentage cars won</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contestant sticks to the original choice</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>A</td>
</tr>
<tr>
<td>Contestant always chooses other door</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>
What is the format for each single simulation in the numbered list?

The contents of the three doors – randomly placed goat, goat, car.

The contestant’s first choice (both the contents and the index number)

Monty Hall’s choice (both the contents – always a goat - and the index number)

The contestants final choice – depending on the algorithm used (both the contents and the index number)

What the contestant won

Multiple assignment

Returning multiple values from a function

Removing an element from a string/list object

Building up strings

Converting an integer into a string

The game doors contain the randomly positioned strings: ‘goat’, ‘goat’, ‘car’. The contestant’s choice and Monty's choice are an index position (0, 1 or 2).
The function which carries out a set of simulations

* This function runs the required number of simulations. Which helper functions are needed?

```python
def run_simulations(how_many, algorithm_type):
    pass
```

Helper functions

* Returns a list containing a random combination of the three strings 'goat', 'goat', 'car'.

```python
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
```

Helper functions

* Returns the a tuple (monty's door index and the other door index)

```python
def get_monty_and_other_door_indices(doors, indices_remaining):
    item_at_index_0 = doors[indices_remaining[0]]
    item_at_index_1 = doors[indices_remaining[1]]
```

```python
def run_one_simulation(algorithm_type):
    doors = get_random_combination()
    monty_door_index, other_door_index = get_monty_and_other_door_indices(doors, indices_remaining)
    ...
Helper functions

* Returns a tuple: a boolean indicating whether or not the car has been won, and a string with all the information about a single simulation (see example on slide 5).

```python
def get_results(doors, monty_index, contestant_old_index, contestant_final_index):
    result_info = 'Content of the doors: ' + doors[0] + " " +
                   doors[1] + " " +
                   doors[2] + "\n"
    contestant_has_won_car = doors[contestant_final_index] == 'car'
    return (contestant_has_won_car, result_info)
```

```python
contentant_has_won_car = doors[contentant_final_index] == 'car'
return (contentant_has_won_car, result_info)
```

```python
def run_one_simulation(algorithm_type):
    ... doors = get_random_combination()
    ... has_won_car, result_info = get_results(doors, monty_index, contestant_old_index, contestant_final_index)
```

* Writes all the information to the output file. See slide 4 for a better description of the format of the output file.

```python
def write_simulation_results_to_file(filename, algorithm_desc, cars_won, how_many, results_str):
    ...
```

```python
def main():
    ... write_simulation_results_to_file(filename, algorithm_desc, cars_won, how_many, results_str)
main()
```