Lecture 21 - Problem Solving – Monty Hall
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?**
Is it to your advantage to change your choice?

... 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. ...

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.
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?

Either 'contestant always chooses other door.' Or 'contestant always sticks to the original choice.'
The Problem Output

What is the format for each single simulation in the numbered list?

A section of the example output file

1. Content of the doors: goat car goat
   Contestant original choice: car (1)
   Monty choice: goat (0)
   Contestant final choice: goat (2)
   Contestant won a goat

2. Content of the doors: goat car goat
   Contestant original choice: car (1)
   Monty choice: goat (0)
   Contestant final choice: goat (2)
   Contestant won a goat

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
The Problem - Initialising

What data structures will I need to store the information for this simulation?

How will I initialise the data structure I choose to use?

What tasks need to be done to solve this problem? What functions will I need?
- 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 main() function for our program

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)
main()
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):

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)
```
Helper functions

- This function runs one single simulation and returns a tuple:
  - a boolean indicating if the contestant has won a car, and
  - a string with the single simulation information.

```
def run_one_simulation(algorithm_type):
    return (has_won_car, result_info)

def run_simulations(how_many, algorithm_type):
    ...
    has_won_car, result_info = run_one_simulation(algorithm_type)
    ...

'Content of the doors: goat goat car
Contestant original choice: goat (1)
Monty choice: goat (0)
Contestant final choice: car (2)'
```
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

def run_one_simulation(algorithm_type):
    ...
    doors = get_random_combination()
    ...
```
Helper functions

- Returns the a tuple (monty's door index and the other door index)
- `indices_remaining` is a list containing the two remaining indices. `monty_index` is the index chosen by Monty Hall (i.e., the index of a box containing a goat).

```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):
contestant_has_won_car = doors[contestant_final_index] == 'car'
...
return (contestant_has_won_car, result_info)
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)
```

Example string returned by ONE simulation

'Content of the boxes: goat goat car
Contestant original choice: goat (1)
    Monty choice: goat (0)
    Contestant final choice: car (2)
Contestant has won a car'
The Program Output

- 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):
    # Function to write simulation results to a file.

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