Homework 2

Vincent Xie

Table of contents

Exercise 3

Monty Hall

Write a function to demonstrate the Monty Hall problem through simulation. The function takes two arguments `ndoors` and `ntrials`, representing the number of doors in the experiment and the number of trails in a simulation, respectively. The function should return the proportion of wins for both the switch and no-switch strategy. Apply your function with 3 doors and 5 doors, both with 1000 trials. Include sufficient text around the code to explain your them.

```python
import random
random.seed(3255)

def monty_hall(ndoors, ntrials):
    # Initialize a counter for wins
    switch_wins = 0
    no_switch_wins = 0

    for i in range(ntrials):
        # Depending on number of doors, one door will have a prize
        doors = ['goat'] * (ndoors-1) + ['car']

        # Shuffle the doors
        random.shuffle(doors)

        # Random choice based on number of doors
        choice = random.randint(0, ndoors-1)
```

```python
        # Opens door with goat behind it
        monty_choices = [i for i in range(ndoors)
                         if i != choice and doors[i] == 'goat']
        monty_open = random.choice(monty_choices)

        # Strategy of switch or no switch
        switch_choice = [i for i in range(ndoors)
                         if i != choice and i != monty_open]

        # Checks if player wins with switch strategy
        if doors[random.choice(switch_choice)] == 'car':
            switch_wins += 1

        # Checks if player wins with no switch strategy
        if doors[choice] == 'car':
            no_switch_wins += 1

    # Calculates percentages from win proportions
    switch_win_percentage = (switch_wins/ntrials) * 100
    no_switch_win_percentage = (no_switch_wins/ntrials) * 100

    return switch_win_percentage, no_switch_win_percentage

# Simulation with 3 doors and 5 doors for 1000 trials
switch_wins_3, no_switch_wins_3 = monty_hall(3, 1000)
switch_wins_5, no_switch_wins_5 = monty_hall(5, 1000)

print("Monty Hall Simulation Results")
print("3 Door - Switch Strategy Win Proportion: {:.1f}%".format(switch_wins_3))
print("3 Door - No Switch Strategy Win Proportion: {:.1f}%".format(no_switch_wins_3))
print("5 Door - Switch Strategy Win Proportion: {:.1f}%".format(switch_wins_5))
print("5 Door - No Switch Strategy Win Proportion: {:.1f}%".format(no_switch_wins_5))
```

```
Monty Hall Simulation Results
3 Door - Switch Strategy Win Proportion: 67.0%
3 Door - No Switch Strategy Win Proportion: 33.0%
5 Door - Switch Strategy Win Proportion: 27.4%
5 Door - No Switch Strategy Win Proportion: 19.3%
```

Exercise 6

Game 24

The math game 24 is one of the addictive games among number lovers. With four randomly selected cards form a deck of poker cards, use all four values and elementary arithmetic operations $(+ - \times /)$ to come up with 24. Let $\square$ be one of the four numbers. Let $\bigcirc$ represent one of the four operators. For example,

$$(\square \bigcirc \square) \bigcirc (\square \bigcirc \square)$$

is one way to group the the operations.

1. List all the possible ways to group the four numbers.

Linear

$$\square \bigcirc \square \bigcirc \square \bigcirc \square$$

One Bracket

$$(\square \bigcirc \square) \bigcirc \square \bigcirc \square$$

$$\square \bigcirc (\square \bigcirc \square) \bigcirc \square$$

$$\square \bigcirc \square \bigcirc (\square \bigcirc \square)$$

Two Brackets

$$(\square \bigcirc \square) \bigcirc (\square \bigcirc \square)$$

Nested Brackets

$$((\square \bigcirc \square) \bigcirc \square) \bigcirc \square$$

$$(\square \bigcirc (\square \bigcirc \square)) \bigcirc \square$$

$$\square \bigcirc (\square \bigcirc (\square \bigcirc \square))$$

$$\square \bigcirc ((\square \bigcirc \square) \bigcirc \square)$$

2. How many possibly ways are there to check for a solution?

Number Choice: There are 4 different numbers so this is a permutation of 4 numbers. So there are

$$4! = 4 * 3 * 2 * 1 = 24$$

ways to arrange the numbers.

Operation Choice: There are 3 different slots for the 4 basic arithmetic operations. Therefore, there are

$$4^3 = 64$$

possible combinations for the operations.

Grouping: From the previous question, we see that there are 9 different ways of grouping the shapes.

To find out how many ways there is to check for a solution, we use...

$$TotalPossibilities = NumberArrangements * OperationChoices * NumberOfGroupings$$

With this equation, we find out that there are a total of 13,824 possible ways to check for a solution in the game of 24.

3. Write a function to solve the problem in a brutal force way. The inputs of the function are four numbers. The function returns a list of solutions. Some of the solutions will be equivalent, but let us not worry about that for now.

```
from itertools import permutations, product

def game_of_24(numbers):
  solutions = []
  operators = ['+', '-', '*', '/']

  # Iterate through each permutation
  for perm in permutations(numbers):
    # Generate all possible arrangements of operators
    for ops in product(operators, repeat=3):
      # Evaluate expression
      expression = '({}{}{}){}({}{}{})'.format(perm[0], ops[0], perm[1],
                                                ops[1], perm[2], ops[2], perm[3])
      try:
        # If the expression is equal to 24, append to list of solutions
        if eval(expression) == 24:
          solutions.append(expression)
      # Divide by zero edge case
      except ZeroDivisionError:
          continue
  return solutions

# Example
numbers = [3, 6, 8, 3]
solutions = game_of_24(numbers)
for sol in solutions:
  print(sol)
```

```
(3+6)*(8/3)
(3+6)/(3/8)
(6+3)*(8/3)
(6+3)/(3/8)
(6+3)/(3/8)
(6+3)*(8/3)
```

```
(8/3)*(6+3)
(8/3)*(3+6)
(8/3)*(3+6)
(8/3)*(6+3)
(3+6)/(3/8)
(3+6)*(8/3)
```