CS 171 - Programming Assignment 4

Jeffrey Popyack and Mark Boady - Drexel University

November 14, 2017

1 Overview

There used to be a TV game show called “Let’s Make a Deal” which ran for many years with host Monty Hall (1921-2017).

`www.letsmakeadeal.com`

A typical situation in the game show was this: the stage is set up with three large doors labeled #1, #2, and #3. Behind two of the doors is a goat. Behind the other door is a new car.

You don’t know which door has the car, but Monty Hall wants you to pick a door. Suppose you pick door #1. Monty Hall then opens one of the other doors (say, door #3) and shows you that there’s a goat behind it. He then says: ”I’ll give you a chance to change your mind. Do you want to change your mind and pick door number 2?” After you make a decision, they open the door you finally picked and you find out whether you’ve won a new car or a new goat. To clarify: the locations of the car and the goats are fixed before your game starts. Monty does not get a chance to switch things around in the middle of the game. He just shows you that one of the choices you didn’t make had a goat.

https://bueschermath.wordpress.com/2013/09/04/3/

There was considerable debate among fans of the show (as well as statisticians and mathematicians) about the following question: When Monty shows you that there's a goat behind one of the doors that you didn't pick, should you switch your choice? Or should you stay with your original decision?

You will write a program to simulate this game and analyze this situation

2 Random Numbers in Python

Python 3 has a Library for randomizing values called **random**.

The shuffle command will take a list and shuffle the values.

```
>>> import random
>>> L =[1,2,3]
>>> random.shuffle(L)
>>> print(L)
>>> [2, 3, 1]

>>> random.shuffle(L)
>>> print(L)
>>> [1, 3, 2]
```

We can also generate random numbers.

```
>>> random.randint(0,10)
>>> 6
random.randint(10,20)
>>> 20
```

Testing code that uses random numbers can be hard. The seed command can be used to control the sequence of random numbers generated. This will fix the sequence of execution for your program.

The following code will always print the same output.

```
import random
random.seed(10)
print(random.randint(0,10))
```

The below code will be random.

```
import random
random.seed()
print(random.randint(0,10))
```

We recommend you use a seed when developing your code, when switch to random values once you have tested it.

You should only call seed once for your entire program. Put it near the top of your file.

3 Programming Project

3.1 Set up Doors

Create a function `def setupDoors()` that returns a list with three strings. The list will have three values. Two values will be "G" for goat and one will be "C" for car. This function returns a list of strings.

For example,

```
print(setupDoors())
```

Might print ['G', 'G', 'C']. This says the car is behind door #3 and goats are behind doors #1 and #2.

Your function should use randomization to change which door is selected with each run.

3.2 Player Door

Create a function `def playerDoor()` that randomly selects a door between 1 and 3. This is the door the player selected. This function returns an integer between 1 and 3.

3.3 Monty Door

Create a function `def montyDoor(Doors, Player)` that takes the list of doors and the door the player selected. Return the door Monty selected. Monty always selects a door such that

1. The door has a goat.
2. The door is not selected by the player.

This function returns an integer between 1 and 3.

3.4 Play Round

Create a function `def playRound()` that plays one game of Let's Make a Deal. The function returns either 0 or 1.

- 0 - The player wins if they stay with their original selection.

- 1 - The player wins if they switch to a new door after Monty open's his.

You **must** use the functions you write in previous sections to complete this function.

3.5 Main Program Body

You python file should be named monty.py.

Ask the user how many experiments to run. Print out what percent of the random experiments would be won by staying with the original choice and how many would be won by switching to a new door. You **must** use the functions you have written above. If you user gives a bad number, ask again.

Continue running tests until the user enters "exit".

4 Example Execution Trace

You are not required to exactly match the below layout, but your content and results must be the same.

Since this program uses a random number generator, you will not get the same answers I do. I used the random.seed(10) to do the below experiments.

4.1 Example

```
Welcome to Monty Hall Analysis
Enter 'exit' to quit.
How many tests should we run?
Bad Input Test
Please enter a number:
2
Stay Won 50.0% of the time.
Switch Won 50.0% of the time.
How many tests should we run?
4
Stay Won 25.0% of the time.
Switch Won 75.0% of the time.
How many tests should we run?
8
Stay Won 50.0% of the time.
Switch Won 50.0% of the time.
How many tests should we run?
16
Stay Won 12.5% of the time.
Switch Won 87.5% of the time.
How many tests should we run?
32
```

```
Stay Won 25.0% of the time.
Switch Won 75.0% of the time.
How many tests should we run?
exit
Thank you for using this program.
```

5 Analysis

Create a Document in the Word Processor of your choosing and use your program to fill out the following table. Run 5 tests with each number of tests. Once you have completed the table, write a paragraph uses the data to justify if it is better to switch or stay when playing Let's Make a Deal.

Num Tests	10	50	100	500	1000	5000	10000
Test 1 (% Switch Wins)							
Test 1 (% Stay Wins)							
Test 2 (% Switch Wins)							
Test 2 (% Stay Wins)							
Test 3 (% Switch Wins)							
Test 3 (% Stay Wins)							
Test 4 (% Switch Wins)							
Test 4 (% Stay Wins)							
Test 5 (% Switch Wins)							
Test 5 (% Stay Wins)							

6 Grading

There are no strict guidelines for how to write your code or develop your user interface. You will be graded on the quality of your design and execution.

- Analysis (15 points)
  1. 1 point per row of data.
  2. 5 points for a reasoned analysis of the data
- Set Up Doors Function Correct (10 points)
- Player Door Function Correct (10 points)
- Monty Door Function Correct (10 points)
- Play Round Function Correct (15 points)
- Main Problem (30 points)
  - Computes Test Results Correctly (20 points)
  - Handles Bad Input Numbers (3 points)
  - Exits on Correct Input (2 points)

    - Uses Functions to Compute Results (5 points)

- User Interface easy to read/understand (4 points)
- File is well commented (3 points)
- Name in Comments (1 point)
- Section Number in Comments (1 point)
- File named correctly (1 point)

If you code has any runtime errors, a 50% deduction will be taken. Only portions of the code that execute without errors will be graded.

7 Resources

Additional Resources
`https://betterexplained.com/articles/understanding-the-monty-hall-problem/`