CS 171 - Programming Assignment 4

Jeffrey Popyack and Mark Boady - Drexel University

February 5, 2018

1 Contents

Contents

2 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

3 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.

4 Programming Project

Create a Python 3 program in the file **monty.py**. The program will simulate multiple games of "Let's Make a Deal." *Hint: Start by simulating a single game, then try to simulate multiple games.*

4.1 Preparation

This program uses a random number generator, but it also being designed for a class. To make grading of the program easier, the program will ask the user for the Random Number generator seed. This is not something you should ever to in practice. The goal of a random number generator is to be random. This is only being done to allow for automatic testing.

Before your program begins simulating "Let's Make a Deal," ask the user for an integer and set it as the seed. If the user gives you input that is not a number, the program should exit immediately.

4.2 Repeated Games

After setting the seed, you program should welcome the user to the Monty Hall Analysis program. Ask the user how many tests they want to run. If the user enters "exit" as the number then the program should exit. If they enter a number, the program should run that many games and print the results. If the user enters something else (not a number and not exit) ask again.

Once the simulation is run, ask for how many test to run again. Repeat this process until the user tells you to exit the program. After each set of games, give a summary of how many times the player should have switched and how many times they should have stayed.

If the number of games is less than or equal to 10, print out the results of each game. If the result is greater than 10, just print the summary.

4.3 Simulating a Single Game

Simulating a single game follows the below pattern.

1. Create a List with the car behind door number 1. Shuffle a list of doors to randomize the car location.
2. Have the player pick a door between 1 and 3.
3. Have Monty Pick a door. Monty must pick a door with the goat and cannot pick a door the player picked.
4. Determine if the player should switch or not.

4.4 Example

You are not required to perfectly match the below outputs. Even using a planned seed, the results could be slightly different. The seed is important because it allows the graders to repeat the same test multiple times on your code. Depending on your implementation, even with the same seed you may not exactly match the below examples.

4.4.1 Example 1

```
Enter Random Seed:
cats
Seed is not a number!
```

4.5 Example 2

```
Enter Random Seed:
10
Welcome to Monty Hall Analysis
Enter 'exit' to quit.
How many tests should we run?
Robots
Please enter a number:
Ghost
Please enter a number:
1
Game 1
Doors: ['G', 'C', 'G']
Player Selects Door 2
Monty Selects Door 1
Player should stay to win.
Stay Won 100.0% of the time.
Switch Won 0.0% of the time.
How many tests should we run?
7
Game 1
Doors: ['G', 'C', 'G']
Player Selects Door 1
Monty Selects Door 3
Player should switch to win.
Game 2
Doors: ['C', 'G', 'G']
Player Selects Door 2
Monty Selects Door 3
Player should switch to win.
Game 3
Doors: ['G', 'C', 'G']
Player Selects Door 1
Monty Selects Door 3
Player should switch to win.
Game 4
Doors: ['C', 'G', 'G']
Player Selects Door 2
Monty Selects Door 3
```

```
Player should switch to win.
Game 5
Doors: ['G', 'G', 'C']
Player Selects Door 3
Monty Selects Door 1
Player should stay to win.
Game 6
Doors: ['G', 'C', 'G']
Player Selects Door 2
Monty Selects Door 1
Player should stay to win.
Game 7
Doors: ['G', 'G', 'C']
Player Selects Door 2
Monty Selects Door 1
Player should switch to win.
Stay Won 28.6% of the time.
Switch Won 71.4% of the time.
How many tests should we run?
10
Game 1
Doors: ['C', 'G', 'G']
Player Selects Door 3
Monty Selects Door 2
Player should switch to win.
Game 2
Doors: ['C', 'G', 'G']
Player Selects Door 1
Monty Selects Door 2
Player should stay to win.
Game 3
Doors: ['C', 'G', 'G']
Player Selects Door 3
Monty Selects Door 2
Player should switch to win.
Game 4
Doors: ['G', 'C', 'G']
Player Selects Door 2
Monty Selects Door 1
Player should stay to win.
Game 5
Doors: ['G', 'G', 'C']
Player Selects Door 3
Monty Selects Door 1
Player should stay to win.
Game 6
```

```
Doors: ['G', 'C', 'G']
Player Selects Door 3
Monty Selects Door 1
Player should switch to win.
Game 7
Doors: ['G', 'G', 'C']
Player Selects Door 1
Monty Selects Door 2
Player should switch to win.
Game 8
Doors: ['G', 'G', 'C']
Player Selects Door 3
Monty Selects Door 1
Player should stay to win.
Game 9
Doors: ['G', 'C', 'G']
Player Selects Door 2
Monty Selects Door 1
Player should stay to win.
Game 10
Doors: ['C', 'G', 'G']
Player Selects Door 2
Monty Selects Door 3
Player should switch to win.
Stay Won 50.0% of the time.
Switch Won 50.0% of the time.
How many tests should we run?
25
Stay Won 12.0% of the time.
Switch Won 88.0% of the time.
How many tests should we run?
50
Stay Won 24.0% of the time.
Switch Won 76.0% of the time.
How many tests should we run?
100
Stay Won 37.0% of the time.
Switch Won 63.0% 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 each number of tests using different random

seeds. 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
Seed 1 (% Switch Wins)							
Seed 1 (% Stay Wins)							
Seed 2 (% Switch Wins)							
Seed 2 (% Stay Wins)							
Seed 3 (% Switch Wins)							
Seed 3 (% Stay Wins)							
Seed 4 (% Switch Wins)							
Seed 4 (% Stay Wins)							
Seed 5 (% Switch Wins)							
Seed 5 (% Stay Wins)							

6 Grading

You are not required to perfectly match the example output, but your content should be correct. Your output should also be easy to read and understand. 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
- Sets a Random Seed from User Input. (7 points)
- Program Exits gracefully on non-integer seed. (3 points)
- Program Exits gracefully if "exit" is entered as number of tests. (7 points)
- Program asks repeatedly if number of test in non-integer. (7 points)
- Program runs correct number of experiments. (7 points)
- Program prints summary results after each round of tests. (7 points)
- If number of tests is less than or equal to 10, prints results of each test. (7 points)
- Door are shuffled randomly. (7 points)
- Player selects random door. (7 points)
- Monty Selects a valid door. (7 points)
- Determines if Player should switch or stay. (7 points)
- Accurately commutes percentages for summary results. (7 points)

- File is well commented (2 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 point deduction will be taken. Only portions of the code that execute without errors will be graded. The graders will not test with bad inputs unless specified in the grading scheme.

7 Resources

Additional Resources

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