CHS CS0008
University of Pittsburgh
Project 1

Overview

The Monty Hall problem is stated as follows:

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. You pick a door, say No. 1, and the host, who knows what's behind the doors, opens another door, say No. 3, which has a goat. He then says to you, "Do you want to pick door No. 2?" Is it to your advantage to switch your choice?

With this first project, we will write a simulator to see if it is more advantageous to always switch or to always stay with your original choice.

Rather than write the program all at once, you will program one task at a time, building up to the fully working end result. If you have trouble programming the control structure that connects all of these parts, you may wish to draw a workflow chart.

Background

One thing that our simulation will need to do is repeatedly pick a random door to hide the car behind. Generating random numbers can be tricky, but fortunately someone has already written Python code for us to use.

With `print()` and `input()` you have already been using code written by someone else within your programs. The functionality that they provide is used so often that they are *built-in* to the Python language. Since the code to generate random numbers is not built-in, we will need to explicitly tell Python that we would like to use it via the `import` statement:

```
from random import randint
```

We can now use `randint()` to generate random numbers! For example, the following line of code will store an integer between 1 and 3 (i.e., either 1, 2, or 3) in the variable named x:

```
x = randint(1, 3)
```

With `randint()`, we can set to building our simulator!

Note that `randint()` is one of the many, many things already programmed for you in the Python *standard library.* We will be covering other parts of the standard library later in the course.

1 A basic game (without switching)

For the first part of this project, you will use input, `randint()`, and output to play a single round of the game.

1. Create a new Python program.
2. Use the `import` statement to tell Python we will be using `randint()`.
3. Use `randint()` to select a door for the car to be hidden behind (1, 2, or 3), and assign the result to a variable named `car`.
4. Prompt the user for a guess as to which door hides the car, convert their entry to an integer using `int()`, and assign the result to a variable named `guess`.
5. Print to the screen which door was actually hiding the car (the value of the `car` variable).
6. Print to the screen which door the user had picked (the value of the `guess` variable).

Here is the output from several example runs of the desired program:

```
Which door would you like to pick:  3
The car was behind Door #2!
You picked Door #3.

Which door would you like to pick:  1
The car was behind Door #1!
You picked Door #1.

Which door would you like to pick:  1
The car was behind Door #3!
You picked Door #1.
```

2 Revealing a goat and allowing the user to switch

For the second part of the project, you will add decision structures to your program. This will allow the program to select a door hiding a goat to reveal to the user, and hence, the user can be given the option to switch their selection.

If the user's selection is not the door hiding a car, there is only one choice for your program to reveal (e.g, if the car is behind Door 1, and the user picked Door 2, your program must reveal the goat behind Door 3). If the user's initial choice does hide the car (e.g., the car is behind Door 1 and the user picked Door 1), your program should reveal the next available door. I.e.:

- If the car is behind Door 1 and the user picked Door 1
  - Reveal the goat behind Door 2
- If the car is behind Door 2 and the user picked Door 2
  - Reveal the goat behind Door 3
- If the car is behind Door 3 and the user picked Door 3
  - Reveal the goat behind Door 1

After a goat is revealed to the user, give them the option to change their selection. Once they have made a decision, tell the user whether they have won or lost.

Here is the output from several example runs of the desired program:

```
Which door would you like to pick:  1

There is a goat behind Door #2

Would you like to change your pick? no

The car was behind Door #3!
You lost.
```

```
Which door would you like to pick:  2

There is a goat behind Door #1

Would you like to change your pick? no

The car was behind Door #2!
You won!
```

```
Which door would you like to pick:  2

There is a goat behind Door #3

Would you like to change your pick? yes

The car was behind Door #2!
You lost.
```

3 Building the simulation

For the third part of this project, you will build a simulator to run the game a large number of times automatically.

You should first change your program to, at the start, ask the user to pick a number of rounds of the game to simulate (between 10 and 10,000). Next, ask the user if the simulator should always *switch* or always *stay*.

After this, your program should simulate the specified number of rounds. In each round, it should pick a random door to hide the car. Next, it should pick a random door for the "player". If the user stated that the simulator should always stay, your program should record whether the "player" won or lost and begin the next round. If the user stated that the simulator should always switch, your program should determine which door would be revealed to the "player" (using the rules in the previous step), and then record whether or not the car is behind the door that the "player" would switch to.

After the specified number of rounds are have been completed, your program should print out the number of rounds that were won, as well as the percentage of rounds that were won.

By running your simulator multiple times, you can determine once and for all what the best strategy is! (If there is one...)

Here is the output from an example run of the desired program:

```
How many rounds of the game should be simulated?  5
Must enter a number between 10 and 10000
Please try again:  2000000000
Must enter a number between 10 and 10000
Please try again:  10

Should the player switch or stay?  foo
Must enter either "switch" or "stay"
Please try again:  bar
Must enter either "switch" or "stay"
Please try again:  baz
Must enter either "switch" or "stay"
Please try again:  stay

The player won 4/10 games (40.00%)!
```