Homework 2

Due: October 24, 2019 **Points:** 100

The Monty Hall problem is a very famous problem in probability. It's based on an old TV show called "Let's Make a Deal". The stage of that show had 3 doors numbered "1", "2", and "3". Behind one of the doors was a valuable prize (like a new car); the other two contained gag gifts (like a goat or a can of cat food). The host, Monty Hall (hence the name of the problem), would choose someone from the audience and ask them to pick a door. The contestant would choose one, say door "2". Monty would then open one of the other doors that *always* had a gag gift behind it (say, door "1" for our example). He would then ask the contestant if he or she wanted to stay with door "2", or change their selection to door "3". The problem is to determine which action – keep or change – gives the contestant the greater probability of selecting the door with the real prize.

We're going to answer this question by simulation — the technique is called a *Monte Carlo* method. Basically, we play a large number of games on the computer, always switching doors (or never switching doors), and record whether we won. We then divide the number of times we won by the number of trials, giving a number between 0 and 1 (inclusive). This is the probability that the strategy will cause the contestant to win.

We're going to build this program in steps, because that will simplify writing it. It will also show you how a program is put together!

1. (*10 points*) The basis for this simulation is a function that generates numbers randomly. Write a function that uses the Python random number generator (see section 4.5) to generate one of the numbers 1, 2, and 3 ranadomly. Then write a small program that calls this function 50 times and prints each number generated. Print the numbers on the same line with no intervening spaces.
   The function, and the program, take no input.

   To turn in: Please turn in the program in the file *monty1.py*.

2. (*40 points*) Next, we will write functions to simulate playing one game.
   The basic approach is to generate a random number representing the door behind which the real prize sits, and another random number representing the door that the contestant initially selects. Monty opens the remaining door. Then, have the contestant switch doors (or not switch doors), and see if the contestant winds up with the door behind which the prize sits.
   Write two different functions to do this. The first, called `montyalways()`, has the contestant *always* changing doors after Monty opens the third door. The second, called `montynever()`, has the contestant *never* changing doors after Monty opens the third door. Both functions should return the Boolean value `True` if the contestant wins, and the Boolean `False` if she does not.
   Your function should not print anything. It should *only* return `True` or `False`.

   To turn in: Please turn in the program in the file *monty2.py*.

3. (*40 points*) Now we will simulate a large number of games.
   Write a program that asks the user for the number of games to be played. Then play one set of games for the contestant always changing the door and another set for the contestant never changing the door. Print the resulting (decimal) fraction of times that the contestant wins, and the number of games won.
   For input, your program asks the user for the number of games to be played. This must be a *positive* integer. Remember to handle invalid inputs gracefully, by printing an error message and exiting the program.
   Here is what a correct input should look like (the red text is what you type):

   ```
   Number of games to play: 100000
   ```

   If the input is invalid:

   ```
   Number of games to play: hello
   Please enter a positive integer
   ```

   and then the program exits.
   The output of your program must look like this:

   ```
   Out of 100000 games:
   Always switching wins:  0.6680600 (66806 games)
   Never switching wins:  0.3346300 (33463 games)
   ```

Important note: your numbers may be different.

To turn in: Please turn in the program in the file *monty3.py*.

Extra Credit

E1. (*20 points*) Write a function to determine whether a year, given as the argument, is a leap year. A year is a leap year if it is evenly divisible by 4, unless it is evenly divisible by 100 and not 400. So 2000 was a leap year, but 2100 and 2200 will not be. Then write a program that asks the user to enter a year and uses the function you wrote to determine whether the year is a leap year. The program then prints the result.

To turn in: Please turn in the program in the file *leap.py*.