The Two Envelopes problem is an interesting decision theory problem.

> Consider that you are given two indistinguishable envelopes, each containing money, one contains twice as much as the other. You may pick one envelope and keep the money it contains. Having chosen an envelope at will, but before inspecting it, you are given the chance to switch envelopes. Should you switch?

> The game is this: \_stick\_ or \_switch\_; It seems obvious that there is no point in switching envelopes as the situation is symmetric. However, because you stand to gain twice as much money if you switch while risking only a loss of half of what you currently have, it is possible to argue that it is more beneficial to switch.The problem is to show what is wrong with this argument.

\*\*Two Envelopes problem\*\*: Implement a function, called `simulateProblem()`, that does the game simulation for the two envelopes problem. Run the simulation 1000 times to figure out the empirical (observed) probability of gaining more money when switching and gaining more money when sticking to the original choice. Each simulation operates as follows:

1. First, randomly pick an envelopes configuration out of the two possible configurations, (A,2A) or (2A,A). In the first configuration, the second envelope has twice the money and in the second configuration, the first envelope has twice the money.

2. Next, randomly pick one of the two envelopes.

3. Finally, randomly choose to either stick or switch. The program checks if you won (the envelope that picked has more money) or not (the envelope that picked has less money). In case of winning, record if the winning was because of \_sticking\_ or \_switching\_.

You can perform the \_random\_ choice as follows, using the `np.random.randint()` method.

```python

import numpy as np

print(np.random.randint(2))

```

The `simulateProblem()` function takes no arguments and returns two values, first is a boolean output which is `True` if you win and `False` if you lose. In case of a win, the second output is `True` if the win was due to \_sticking\_ and `False` if the win was due to \_switching\_.

Once the method `simulateProblem()` that does the above steps and returns \_sticking\_,or \_switching\_, depending on the win/loss scenario, run the method 1000 times and count the number of times the win was due to \_sticking\_ to the pick in Step 2, and number of times the win was due to \_switching\_ the envelope.