1. Abstract: (200 words) What problem and major findings

The setup is a 2-member game and each member takes turns to play. Their actions are dictated by the roll of the die. Both of them start with the resource of a few coins and as the game progresses, the one who runs out of coins (when he does not have coins he is supposed to put in), the game ends and the person loses. We find that the average number of cycles taken for the game to end is 8.875. We plot a histogram to see the distribution of cycle length and conclude that it follows log normal distribution (Shapiro-Wilk Normality). Regarding the beginning advantage, we find no significant beginner's advantage. We conduct several experiments by modifying the game rules and observe the cycle length and beginner advantage. We also find out how good the random number generator is by generating a histogram and two-dimensional plot to ensure if the generator is biased and scatterplot.

2. Background and Description of the Problem:

2.1 Literature review

In game theory, our problem falls under the category of "Stochastic Game". At the beginning, there is an initial state and as each player takes action in sequence, the state changes *stoc*hastically. After each stochastic action, the game moves on to a different random state. Stochastic games have applications in economics, evolutionary biology and computer networks.

2.2 Detailed rules and definitions of the game

Initial conditions: A - 4 coins B - 4 coins Pot - 2 coins

If the player rolls a:

- 1 - then the player does nothing.
- 2 - then the player takes all the coins in the pot.
- 3 - then the player takes half of the coins in the pot (rounded down).
- 4,5,6: then the player puts a coin in the pot.

Ending criteria: A player loses (and the game is over) if they are unable to perform the task (i.e., if they have 0 coins and need to place one in the pot). We define a cycle as A and then B completing their turns. The exception is if a player goes out; that is the final cycle (but it still counts as the last cycle).

2.3 Organization of the report

The rest of the report is organized as follows:
The main findings of cycle length and distributions are elaborated in Chapter 3. Discussions on testing the random number generator, testing beginner's luck and experiments on changing rules of the game are in Chapter 4. We conclude in Chapter 5 with future work.

3. Main Findings

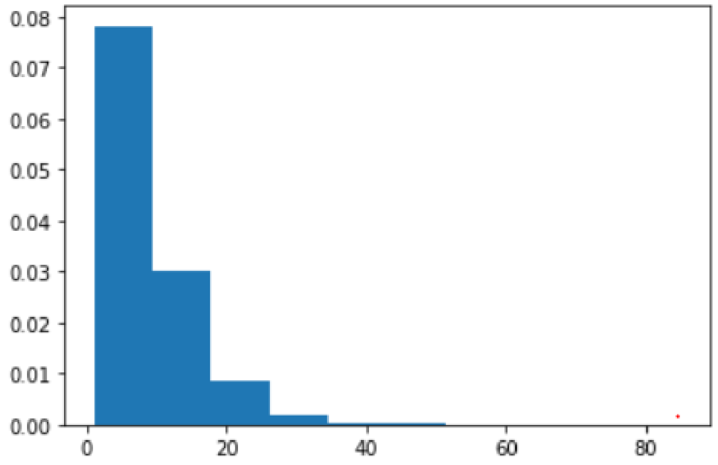
The game logic was coded using Python. Random dice rolls were generated using random.randint python module to generate pseudo random integers from 1 to 6. To find the average number of cycles the game would last, we are simulating 1000000 runs of the game.

- Expected number of cycles

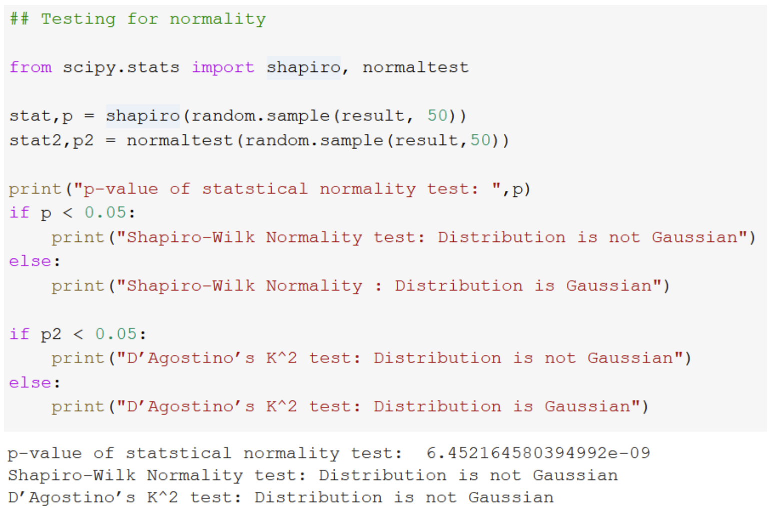
The average number of cycles after simulation comes to 8.875.

- Distribution of game cycles

To identify the distribution of the number of cycles, a histogram was plotted (fig. below).



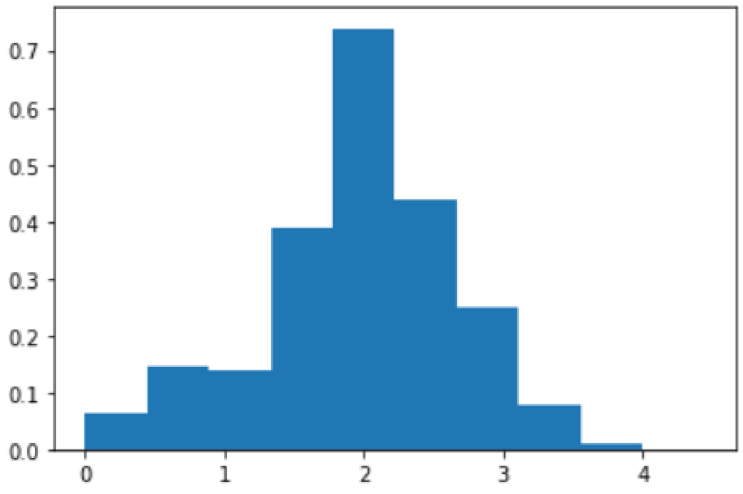
We used the “Shapiro-Wilk” normality test to test if this distribution was normal. The NULL Hypothesis here tests if data follows a normal/ gaussian distribution Vs the alternate hypothesis that data is not normally distributed. Resulting p-value for this test was not significant to prove normality of data. Below are the results.

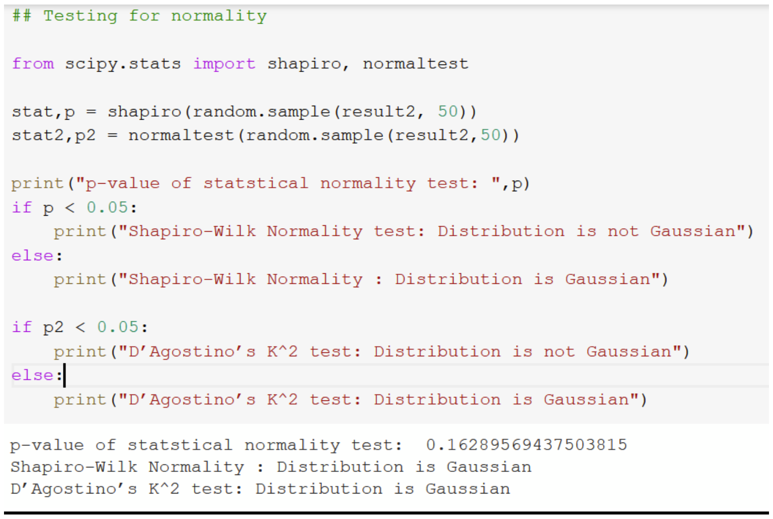


```
## Testing for normality

from scipy.stats import shapiro, normaltest

stat,p = shapiro(random.sample(result, 50))
stat2,p2 = normaltest(random.sample(result,50))

print("p-value of statstical normality test: ",p)
if p < 0.05:
    print("Shapiro-Wilk Normality test: Distribution is not Gaussian")
else:
    print("Shapiro-Wilk Normality : Distribution is Gaussian")

if p2 < 0.05:
    print("D'Agostino's K^2 test: Distribution is not Gaussian")
else:
    print("D'Agostino's K^2 test: Distribution is Gaussian")
```

```
p-value of statstical normality test:  6.452164580394992e-09
Shapiro-Wilk Normality test: Distribution is not Gaussian
D'Agostino's K^2 test: Distribution is not Gaussian
```

Next, we performed log-transformation and fit the transformed data to a histogram. Below is the histogram of log(cycles). Looking at the plot, the distribution looks normally distributed now. To verify this statistically, the “Shapiro-Wilk" normality test was again conducted on this transformed data. The resulting p-value from this test was significant to prove that the transformed data follows a normal distribution.





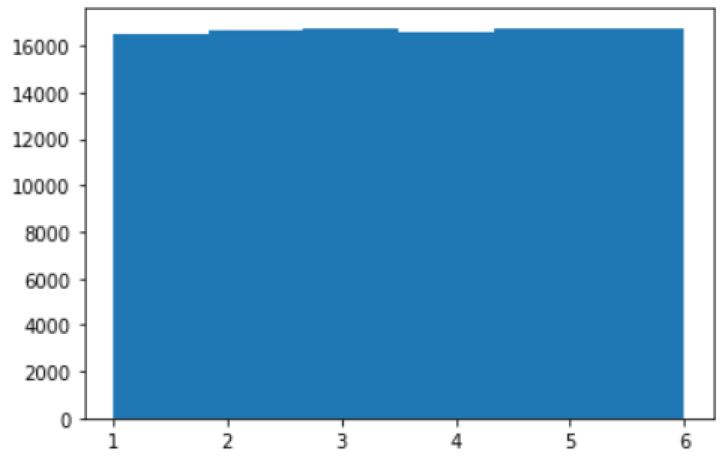
```
## Testing for normality

from scipy.stats import shapiro, normaltest

stat,p = shapiro(random.sample(result2, 50))
stat2,p2 = normaltest(random.sample(result2,50))

print("p-value of statstical normality test: ",p)
if p < 0.05:
    print("Shapiro-Wilk Normality test: Distribution is not Gaussian")
else:
    print("Shapiro-Wilk Normality : Distribution is Gaussian")

if p2 < 0.05:
    print("D'Agostino's K^2 test: Distribution is not Gaussian")
else:
    print("D'Agostino's K^2 test: Distribution is Gaussian")
```

```
p-value of statstical normality test:  0.16289569437503815
Shapiro-Wilk Normality : Distribution is Gaussian
D'Agostino's K^2 test: Distribution is Gaussian
```

4. Discussions

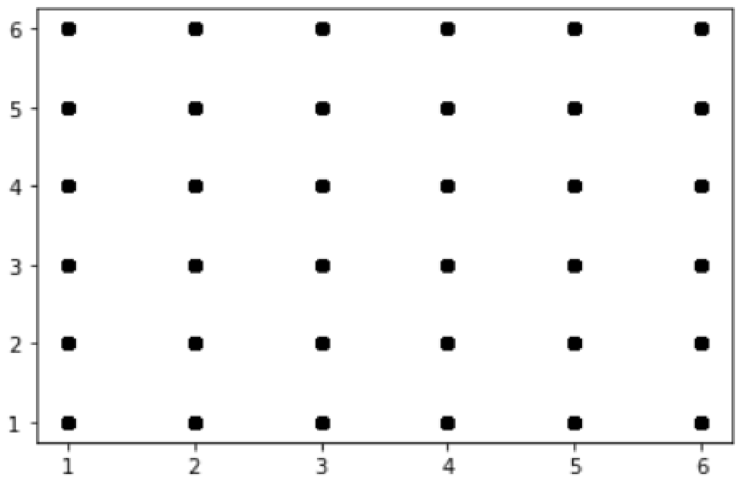
4.1 Testing the goodness of the random number generator

It is essential to have a random number generator that is uncorrelated and non-repetitive because contrary to what they appear to be, the random numbers generated are in fact pseudo-random. We use the numpy's random.randint function to generate the random numbers.



We plot the histogram of the numbers obtained for 100000 numbers and we see almost all numbers being equally distributed with the same frequency - it's unbiased.

To test for repeating patterns, we also plot the two arrays of numbers generated by the random number generator and we see equal distribution in all bins.



4.2 Beginner’s luck

In our simulation, we always let A start the game. To verify if that plays any role in the final victory, we simulate 100000 cases and observe the following numbers:

```
{‘A': 495609, 'B': 504391}
```

We don’t see much of a beginner’s advantage. It is definitely worth exploring this aspect by modifying the rules as done below.

4.3 Changing the rules of the game

Currently, we see
0.5 weightage given to putting 1 in the pot
0.166 for doing nothing
0.166 for taking all in the pot
0.166 take half in the pot

- **Variant 1:** Giving more weightage (0.5) to taking all in the pot and simulating it for 100000 iterations. We see that the new Average cycle length: `2.549699`
  Winning frequency: `Counter({'A': 435525, 'B': 564475})`

  Observation: We see that having higher weightage decreases the cycle length considerably because it increases the probability of running out of coins for either party. It greatly disadvantages the beginner

- **Variant 2**: Giving more weightage (0.5) to do nothing and simulating it for 100000 iterations. We see that the new
  Average cycle length: `8.987181`
  Winning frequency: `{'A': 495741, 'B': 504259}`

  Observation: We see that higher weightage for doing nothing lengthens the cycle length as expected because it puts the state of ‘not being able to pay’ away. We do not see a significant beginners advantage

- **Variant 3:** Giving more weightage (0.5) to taking half in the pot and simulating it for 100000 iterations. We see that the new
  Average cycle length:`9.073737`
  Winning frequency: `{'A': 493078, 'B': 506922})`
  Observation: We see that giving weightage to taking half in the pot increases the cycle since people will not reach the ‘not being able to pay’ state soon. It’s noteworthy that the number of coins they may take could be zero (depending on how much is left in the pot) but the game continues for long. There is no significant beginner’s advantage.

5. Conclusions:

We successfully simulated the main scenarios for the Stochastic game and were able to identify the cycle length, the distributions of the cycle length and test for it. We observe that log normal distribution is the best fit for the distribution of cycle lengths observed. We also test the random number generator for its biases and correlation. We find that numpy’s random.randint() is unbiased using a histogram and uncorrelated using a 2D scatter plot. In addition, we perform experiments by placing higher weightage to different scenarios and observe the cycle length change along with the beginner’s advantage. The results are in accordance with intuition. This was a fun project which let us brainstorm on a variety of possibilities and hypothetical ‘adversarial’ scenarios.

As a future work, it will be interesting to see how the dynamics change on introducing moves like ‘swap coins’ and increasing the number of players. A deeper analysis on how the distribution of cycle lengths vary for multi-player games and additional rules will be worth looking into.