Baseball World Series Simulation

On average, how many games (out of 7) does it take for a team to World Series?

- An American League (AL) baseball team is considered to have a 60% chance of beating the National League (NL) team in any given World Series game. A team wins the World Series by being the first to win four individual games.

Individual game

- Generate a random integer between 1 and 100
- #'s 1-60 will represent an American League (AL) team win
  - 60% chance to win
- #'s 61-100 will represent a National League (NL) team win
  - 40% chance to win

In [64]:
```
# Import the numpy package
import numpy

# Generate random number between 1-100
single_game = numpy.random.randint(1,101)

# Test the percentage ranges
if single_game <= 60:
    print("AL")
else:
    print("NL")
```

```
AL
```

Copy and paste code into a function

- Return a String of the winning division
- Abstraction!

In [65]:
```
# Create function that represents one game being played
def single_game_trial():
    # Generate random number between 1-100
    single_game = numpy.random.randint(1,101)

    # Test the percentage ranges
    if single_game <= 60:
        return("AL")
    else:
        return("NL")
```

In [66]:
```
# Verify that your function is working correctly!
print(single_game_trial())
```

```
AL
```

Individual World Series

- Create two variables for AL and NL win counts
- While both counts are below 4 wins, play a single game
  - Update variables appropriately
- Return a tuple in the form (winning division,total games played)

In [67]:
```
# Import the numpy package
import numpy

# Create variables
AL_wins = 0
NL_wins = 0

# While loop
while AL_wins < 4 and NL_wins < 4:
    result = single_game_trial()
    if result == "AL":
        AL_wins += 1
    else:
        NL_wins +=1

# Total games played before a division reached four wins
total_games_played = AL_wins + NL_wins

# Print winner and games
if AL_wins == 4:
    print("AL WINS!", total_games_played)
else:
    print("NL WINS!", total_games_played)
```

```
NL WINS! 6
```

Defining an individual trial

Copy and paste code into a function

- Return a tuple in the format (winning division, total number of games played)
- Abstraction!

In [68]:
```
# Create function to represent one trial of a World Series
def world_series_trial():
    # Create variables
    AL_wins = 0
    NL_wins = 0

    # While loop
    while AL_wins < 4 and NL_wins < 4:
        result = single_game_trial()
        if result == "AL":
            AL_wins += 1
        else:
            NL_wins +=1

    # Total games played before a division reached four wins
    total_games_played = AL_wins + NL_wins

    # Print winner and games
    if AL_wins == 4:
        return("AL!", total_games_played)
    else:
        return("NL!", total_games_played)
```

In [69]:
```
# Verify that your function is working correctly!
print(world_series_trial())
```

```
('AL!', 7)
```

Finding the average number of total games played to win a World Series

Simulate a large number of individual trials of a World Series, analyze the results

- Create an array to hold your results
- Create a loop to run 10,000 times
  - Add results of a single World Series to the array

In [70]:
```
# Create variables
ws_results = []

# Create a loop that repeats a large number of trials (World Series), in this c
for i in range(10000):
    game_result = world_series_trial()
    ws_results.append(game_result)

# Print the results
print(ws_results)
```

```
L!', 6), ('NL!', 5), ('NL!', 6), ('AL!', 7), ('AL!', 5), ('NL!', 7), ('A
L!', 5), ('AL!', 5), ('AL!', 5), ('NL!', 4), ('AL!', 6), ('AL!', 5), ('N
L!', 5), ('AL!', 4), ('AL!', 7), ('AL!', 4), ('AL!', 6), ('AL!', 5), ('A
L!', 4), ('AL!', 6), ('NL!', 6), ('NL!', 7), ('NL!', 4), ('AL!', 6), ('A
L!', 5), ('NL!', 7), ('AL!', 7), ('AL!', 5), ('AL!', 5), ('AL!', 7), ('N
L!', 5), ('NL!', 5), ('AL!', 7), ('AL!', 5), ('AL!', 7), ('AL!', 6), ('A
L!', 7), ('AL!', 7), ('AL!', 6), ('NL!', 6), ('AL!', 6), ('AL!', 5), ('A
L!', 4), ('AL!', 7), ('NL!', 7), ('AL!', 7), ('AL!', 6), ('AL!', 4), ('A
L!', 4), ('AL!', 6), ('NL!', 7), ('AL!', 6), ('AL!', 7), ('AL!', 5), ('A
L!', 7), ('AL!', 6), ('NL!', 5), ('AL!', 7), ('AL!', 6), ('AL!', 4), ('N
L!', 7), ('AL!', 7), ('AL!', 4), ('AL!', 7), ('NL!', 6), ('AL!', 5), ('A
L!', 6), ('AL!', 4), ('AL!', 5), ('AL!', 6), ('AL!', 5), ('AL!', 6), ('A
L!', 7), ('NL!', 6), ('AL!', 6), ('NL!', 7), ('NL!', 7), ('AL!', 6), ('N
L!', 5), ('AL!', 7), ('NL!', 7), ('NL!', 5), ('AL!', 6), ('NL!', 5), ('N
L!', 7), ('AL!', 6), ('AL!', 6), ('AL!', 6), ('NL!', 6), ('AL!', 6), ('A
L!', 7), ('AL!', 5), ('AL!', 5), ('AL!', 5), ('AL!', 5), ('NL!', 5), ('A
L!', 6), ('AL!', 4), ('AL!', 6), ('AL!', 5), ('NL!', 4), ('NL!', 7), ('N
L!', 7), ('NL!', 5), ('NL!', 7), ('AL!', 5), ('AL!', 6), ('NL!', 7), ('A
L!', 4), ('AL!', 5), ('AL!', 5), ('NL!', 5), ('AL!', 4), ('AL!', 7), ('A
L!', 5), ('AL!', 6), ('AL!', 7), ('AL!', 6), ('AL!', 7), ('AL!', 6), ('A
```

Analyze the results

Find the counts of each number of total games (How many times did it take 4, 5, 6, 7 games to win?)

- Create an array to hold the results
- Loop through each tuple, read second value (total games played)
- Add the value to array

In [71]:
```
# Create variables
total_games_array = []

# Loop through each individual trial within the ws_results array
    # Store second value of the tuple at index 1 (total number of games)
    # Add game total to the total_games_array
for tuple in ws_results:
    num_games = tuple[1]
    total_games_array.append(num_games)

# Print results
print(total_games_array)
```

```
4, 4, 6, 4, 6, 4, 5, 7, 6, 7, 5, 7, 6, 6, 6, 7, 4, 5, 5, 6, 6, 5, 5, 7, 5,
4, 6, 7, 6, 6, 6, 6, 4, 7, 5, 6, 7, 6, 5, 5, 5, 5, 7, 7, 6, 6, 6, 6, 7, 6,
5, 7, 6, 6, 7, 4, 6, 6, 6, 5, 4, 6, 7, 6, 5, 7, 6, 6, 4, 6, 7, 4, 7, 7, 5,
6, 5, 5, 5, 7, 7, 7, 6, 7, 5, 4, 4, 5, 6, 5, 7, 7, 6, 7, 5, 4, 7, 6, 6, 6,
5, 5, 6, 5, 7, 4, 7, 5, 6, 7, 5, 7, 6, 4, 4, 6, 5, 7, 5, 6, 7, 7, 6, 7, 5,
6, 5, 4, 7, 7, 4, 7, 7, 4, 5, 4, 7, 5, 6, 6, 4, 6, 5, 6, 4, 6, 4, 7, 7, 5,
5, 7, 5, 6, 5, 6, 6, 4, 5, 5, 6, 7, 6, 7, 7, 6, 4, 6, 4, 5, 5, 6, 7, 4, 6,
7, 4, 6, 7, 7, 7, 7, 6, 4, 7, 5, 4, 5, 7, 7, 7, 6, 7, 6, 6, 4, 7, 6, 5, 6,
7, 5, 4, 7, 7, 7, 5, 5, 5, 5, 7, 4, 6, 5, 7, 6, 7, 6, 6, 6, 7, 5, 6, 5, 6,
4, 6, 6, 7, 5, 4, 5, 6, 6, 7, 7, 7, 5, 7, 7, 5, 7, 7, 4, 7, 6, 7, 4, 6, 5,
7, 7, 7, 7, 7, 4, 6, 6, 5, 7, 6, 5, 4, 6, 7, 5, 7, 5, 7, 5, 5, 6, 6, 7, 6,
6, 5, 4, 4, 5, 7, 7, 6, 5, 6, 4, 6, 5, 5, 5, 7, 5, 5, 5, 7, 7, 6, 6, 6, 5,
6, 4, 4, 7, 6, 7, 7, 5, 6, 6, 5, 6, 7, 6, 7, 7, 6, 6, 6, 7, 5, 7, 7, 5, 7,
4, 6, 6, 7, 7, 5, 6, 5, 6, 6, 6, 6, 6, 5, 5, 6, 6, 7, 7, 5, 6, 4, 4, 7, 5,
5, 7, 7, 7, 4, 6, 6, 4, 4, 7, 7, 4, 4, 6, 5, 7, 5, 4, 7, 7, 7, 6, 7, 6, 6,
4, 6, 5, 5, 5, 7, 7, 6, 7, 7, 4, 5, 5, 5, 7, 7, 7, 6, 6, 6, 4, 5, 4, 6, 6,
5, 5, 6, 7, 6, 6, 7, 5, 7, 6, 7, 5, 5, 5, 6, 6, 4, 6, 6, 4, 4, 6, 4, 4, 4,
5, 7, 5, 5, 5, 5, 5, 7, 5, 6, 6, 7, 4, 7, 4, 7, 7, 6, 5, 6, 5, 5, 4, 5, 4,
6, 7, 7, 7, 5, 7, 6, 6, 6, 5, 4, 5, 5, 7, 4, 5, 4, 7, 7, 6, 6, 6, 5, 4, 6,
5, 6, 5, 6, 7, 7, 5, 5, 6, 6, 6, 4, 7, 5, 7, 6, 5, 5, 4, 7, 7, 6, 7, 7, 4,
```

Create bins for each total

In [72]:
```
# Create the bin
game_bin = numpy.bincount(total_games_array)

# Print the results
print(game_bin)
```

```
[   0    0    0    0 1553 2727 2963 2757]
```

In [73]:
```
# Make sure this adds to 10,000 trials!
```

Calculate the average number of games played in each trial

- This will represent the average number of games played to win the World Series after 10,000 trials

In [74]:
```
# Use np.average(array)
average = numpy.average(total_games_array)
print(average)
```

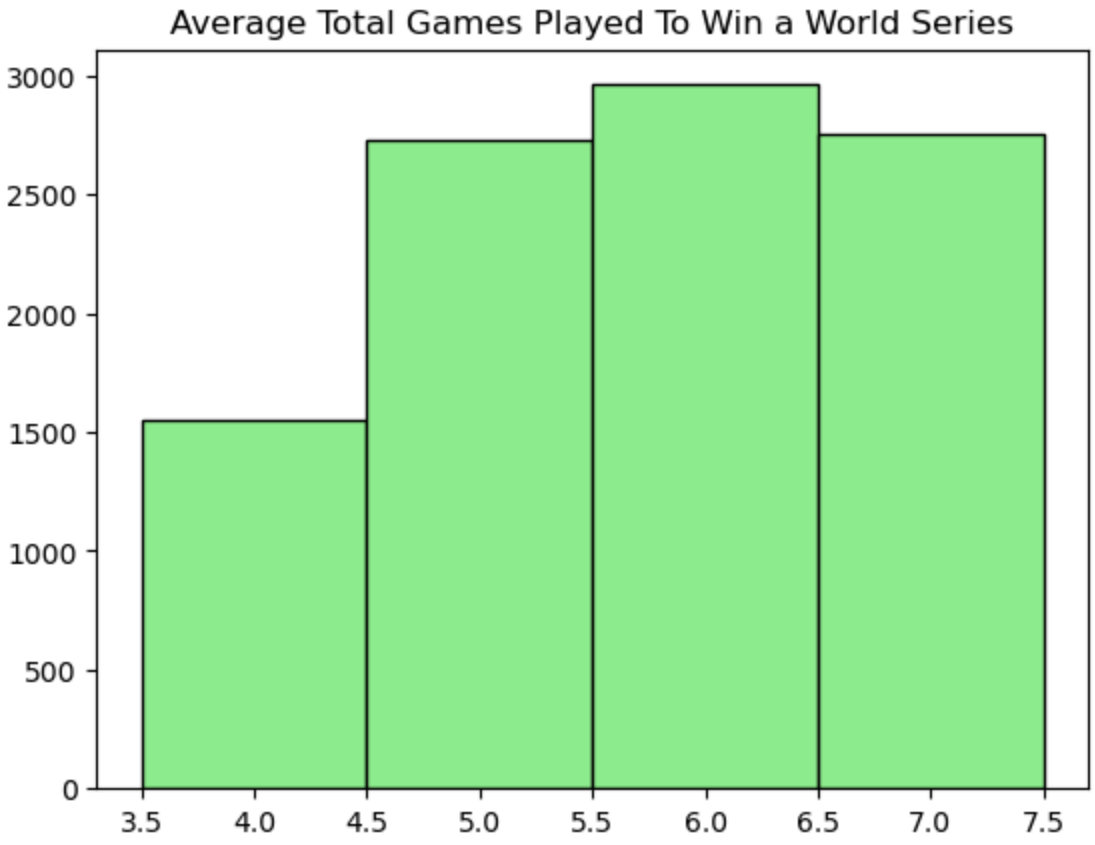
```
5.6924
```

Graph the results

Histogram

- Frequency chart

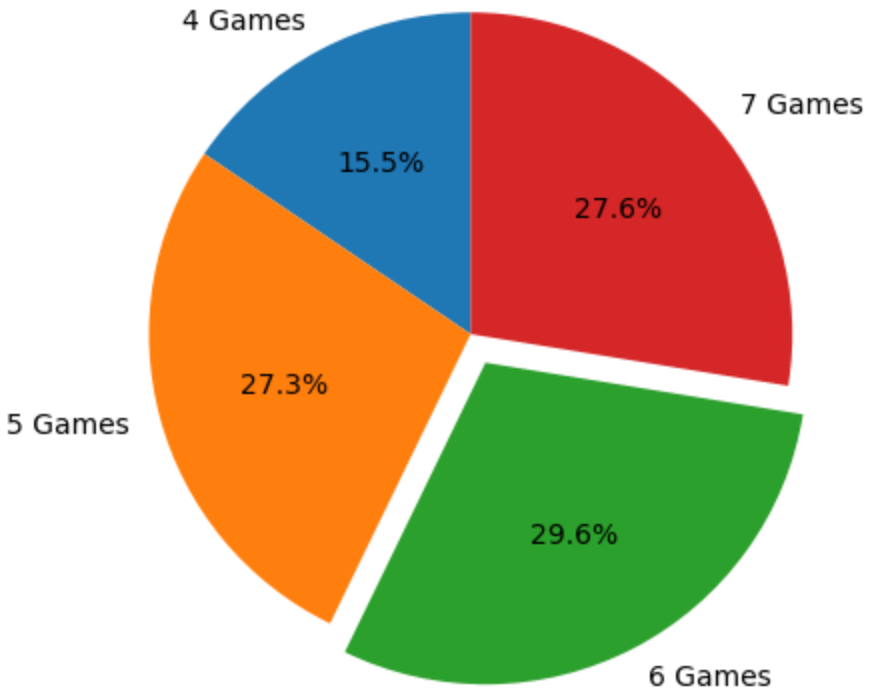
In [75]:
```
# Import the library
import matplotlib.pyplot as plot
%matplotlib inline
# Magic to allow the graph to display directly in this notebook

# Create bins/dividers for your data
bins = [4, 5, 6, 7, 8]

# plot.hist(array, bins, alignment, graph color, border color)
plot.hist(total_games_array, bins, align='left', color='lightgreen' ,edgecolor=
plot.title("Average Total Games Played To Win a World Series")
plot.show()
```



Pie Chart

- Percentage chart

In [76]:
```
# source: http://matplotlib.org/examples/pie_and_polar_charts/pie_demo_features

# Import the library
import matplotlib.pyplot as plot
%matplotlib inline
# Magic to allow the graph to display directly in this notebook

# Create an array of labels
labels = ["4 Games", "5 Games", "6 Games", "7 Games"]

# Crop the game_bin array to exclude the 0's
    # Only need indices 4 to the end
new_game_bin = game_bin[4:]

# Explode option
    # 'Slices' appear distanced from the center
        # Larger numbers = further explosion
    # Explode array should be same size as labels and
explode = (0, 0, 0.1, 0)

# Use matplotlib module subplots() to get data for various charts
    # Returns a tuple in the form (figure,axes)
fig1,ax1 = plot.subplots()

# Use axes to create a pie chart
    # ax1.pie(data array, explode array, labels array, starting angle)
ax1.pie(new_game_bin, explode, labels, autopct='%1.1f%%', startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plot.show()
```



Don't forget to answer the original question!

- Answer should be in the form of a complete sentence

Based on my simulation, on average it takes a team 6 game to win the World Series

Type *Markdown* and LaTeX: α^2

Additional data points

- How many times did an American League team win the World Series compared to the National League?

In [77]:
```
# Create variables
winning_division_array = []

# Loop through each individual World Series within the ws_results array
    # Store first value of the tuple at index 0 (winning division)
    # Add winning division to the winning_division_array
for world_series in ws_results:
    division = world_series[0]
    winning_division_array.append(division)

# Print results
print(winning_division_array)
```

```
L!', 'AL!', 'AL!', 'AL!', 'AL!', 'NL!', 'NL!', 'AL!', 'AL!', 'AL!', 'AL!',
'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'N
L!', 'AL!', 'AL!', 'NL!', 'NL!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'NL!',
'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'N
L!', 'NL!', 'AL!', 'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'NL!',
'AL!', 'NL!', 'AL!', 'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'NL!', 'NL!', 'A
L!', 'NL!', 'AL!', 'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'NL!',
'NL!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'NL!', 'AL!', 'NL!', 'AL!', 'A
L!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'NL!', 'NL!',
'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'AL!', 'A
L!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'NL!',
'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'NL!', 'AL!', 'NL!', 'NL!', 'NL!', 'A
L!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'NL!', 'AL!', 'AL!',
'AL!', 'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'NL!', 'AL!', 'NL!', 'AL!', 'A
L!', 'AL!', 'AL!', 'NL!', 'AL!', 'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'AL!',
'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'AL!', 'NL!', 'AL!', 'A
L!', 'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!', 'AL!',
'NL!', 'AL!', 'AL!', 'AL!', 'AL!', 'NL!', 'AL!', 'AL!', 'NL!', 'AL!', 'A
L!', 'AL!', 'AL!', 'AL!', 'AL!', 'NL!', 'AL!', 'NL!', 'AL!', 'NL!', 'AL!',
```

Count the totals of NL and AL

- Loop through each winning division in the array

- Update a count for each division accordingly

In [79]:
```
# Create variables
total_al = 0
total_nl = 0

# Loop through each division in the winning_division_array, update wins
for division in winning_division_array:
    if division == "AL!":
        total_al = total_al + 1
    else:
        total_nl = total_nl + 1

# Display results
print("Number NL wins: " + str(total_nl))
print("Number AL wins: " + str(total_al))
```

```
Number NL wins: 2890
Number AL wins: 7110
```

Don't forget to answer the original question!

- Answer should be in the form of a complete sentence

Based on my simulation we can conclude that the American League wins 7110 times compared to the National League wins 2890 times

Type *Markdown* and LaTeX: α^2

What else can you learn from this data? In the markdown cell below, list at least 2 questions that you would be able to answer by further analyzing the data. (Note: you do not need to actually answer the questions).

How many of the National League wins were sweeps?

What percent of the games did the American League win?

In [ ]: