Working with Probabilities: Mobile App for Lottery Addiction

In this project, I am going to contribute to the development of a mobile app by writing a couple of functions that are mostly focused on calculating probabilities. The app is aimed to both prevent and treat lottery addiction by helping people better estimate their chances of winning.

The app idea comes from a medical institute which is specialized in treating gambling addictions. The institute already has a team of engineers that will build the app, but they need to get the logical core of the app and calculate probabilities. For the first version of the app, they want me to focus on the 6/49 lottery and build functions that can answer the following questions for users of the app:

- What is the probability of winning the big prize with a single ticket?
- What is the probability of winning the big prize if we play 40 different tickets (or any other number)?
- What is the probability of having at least five (or four, or three) winning numbers on a single ticket?

Core Functions ¶

Below, I will write two functions that I will be using frequently:

factorial() — a function that calculates factorials combinations() — a function that calculates combinations

In [2]:
```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

def combinations(n, k):
    if k < 0 or k > n:
        return 0
    else:
        return factorial(n) // (factorial(k) * factorial(n - k))
```

One-ticket Probability

I need to build a function that calculates the probability of winning the big prize for any given ticket. For each drawing, six numbers are drawn from a set of 49, and a player wins the big prize if the six numbers on their tickets match all six numbers.

The engineer team told me that I need to be aware of the following details when writing the function:

- Inside the app, the user inputs six different numbers from 1 to 49.
- Under the hood, the six numbers will come as a Python list and serve as an input to our function.
- The engineering team wants the function to print the probability value in a friendly way — in a way that people without any probability training are able to understand.

Below, I write the one_ticket_probability() function, which takes in a list of six unique numbers and prints the probability of winning in a way that's easy to understand.

In [3]:
```
def one_ticket_probability(user_numbers):
    total_possible_outcomes = combinations(49, 6)
    successful_outcomes = 1  # There's only one winning combination

    probability = (successful_outcomes / total_possible_outcomes) * 100

    print("Your chosen combination: {}".format(user_numbers))
    print("Total possible outcomes: {:,}".format(total_possible_outcomes))
    print("Number of successful outcomes: {}".format(successful_outcomes))
    print("Probability of winning: {:.10f}%".format(probability))
```

In [4]:
```
## Testing the One-Ticket Probability function
user_numbers = [3, 12, 27, 42, 5, 19]
one_ticket_probability(user_numbers)
```

```
Your chosen combination: [3, 12, 27, 42, 5, 19]
Total possible outcomes: 13,983,816
Number of successful outcomes: 1
Probability of winning: 0.0000071511%
```

Historical Data Check for Canada Lottery

The institute also wants me to consider the data coming from the national 6/49 lottery game in Canada. The data set contains historical data for 3,665 drawings, dating from 1982 to 2018 (the data set can be downloaded from here: https://www.kaggle.com/datasets/datasciencegal/lottery-dataset).

In [7]:
```
import pandas as pd

# Load the CSV file into a pandas DataFrame
lottery_canada = pd.read_csv('649.csv')

# Print the number of rows and columns
num_rows, num_columns = lottery_canada.shape
print("Number of rows:", num_rows)
print("Number of columns:", num_columns)

# Print the first and last three rows

lottery_canada.head(3)
```

```
Number of rows: 3665
Number of columns: 11
```

Out[7]:

	PRODUCT	DRAW NUMBER	SEQUENCE NUMBER	DRAW DATE	NUMBER DRAWN 1	NUMBER DRAWN 2	NUMBER DRAWN 3	NUMBER DRAWN 4	NUMBER DRAWN 5	NUMBER DRAWN 6	BONUS NUMBER
0	649	1	0	6/12/1982	3	11	12	14	41	43	13
1	649	2	0	6/19/1982	8	33	36	37	39	41	9
2	649	3	0	6/26/1982	1	6	23	24	27	39	34

In [8]:
```
lottery_canada.tail(3)
```

Out[8]:

	PRODUCT	DRAW NUMBER	SEQUENCE NUMBER	DRAW DATE	NUMBER DRAWN 1	NUMBER DRAWN 2	NUMBER DRAWN 3	NUMBER DRAWN 4	NUMBER DRAWN 5	NUMBER DRAWN 6	BONUS NUMBER
3662	649	3589	0	6/13/2018	6	22	24	31	32	34	16
3663	649	3590	0	6/16/2018	2	15	21	31	38	49	8
3664	649	3591	0	6/20/2018	14	24	31	35	37	48	17

Function for Historical Data Check

The engineering team tells me that I need to write a function that can help users determine whether they would have ever won by now using a certain combination of six numbers. These are the details to be aware of:

- Inside the app, the user inputs six different numbers from 1 to 49.
- Under the hood, the six numbers will come as a Python list and serve as an input to the function.
- The engineering team wants me to write a function that prints: the number of times the combination selected occurred; and the probability of winning the big prize in the next drawing with that combination.

I will begin by extracting all the winning numbers from the lottery data set. The extract_numbers() function will go over each row of the dataframe and extract the six winning numbers as a Python set.

In [10]:
```
import pandas as pd

# Load the CSV file into a pandas DataFrame
df = pd.read_csv('649.csv')

def compare_ticket(ticket_numbers):
    # Convert the user's ticket numbers into a set for efficient comparison
    user_combination = set(ticket_numbers)

    # Check how many times the user's combination occurred in the historical data
    occurrences = df[df.apply(lambda row: set(row[4:10]) == user_combination, axis=1)].shape[0]

    # Calculate the probability of winning in the next drawing
    total_possible_outcomes = combinations(49, 6)
    probability = (occurrences / total_possible_outcomes) * 100

    print("Your selected combination: {}".format(ticket_numbers))
    print("Number of times this combination occurred in the historical data: {}".format(occurrences))
    print("Probability of winning in the next drawing: {:.10f}%".format(probability))

# Example usage:
user_ticket = [33, 36, 37, 39, 8, 41]
compare_ticket(user_ticket)
```

```
Your selected combination: [33, 36, 37, 39, 8, 41]
Number of times this combination occurred in the historical data: 1
Probability of winning in the next drawing: 0.0000071511%
```

Multi-Ticket Probability

For the first version of the app, users should also be able to find the probability of winning if they play multiple different tickets. For instance, someone might intend to play 15 different tickets and they want to know the probability of winning the big prize.

The engineering team wants me to be aware of the following details when writing the function:

- The user will input the number of different tickets they want to play (without inputting the specific combinations they intend to play).
- The function will see an integer between 1 and 13,983,816 (the maximum number of different tickets).
- The function should print information about the probability of winning the big prize depending on the number of different tickets played.
- The multi_ticket_probability() function below takes in the number of tickets and prints probability information depending on the input.

In [11]:
```
def calculate_chances(num_tickets):
    if num_tickets < 1 or num_tickets > 13983816:
        print("Number of tickets must be between 1 and 13,983,816.")
        return

    total_possible_outcomes = combinations(49, 6)
    probability = (num_tickets / total_possible_outcomes) * 100

    print("Number of tickets played: {}".format(num_tickets))
    print("Total possible outcomes: {:,}".format(total_possible_outcomes))
    print("Probability of winning with these tickets: {:.10f}%".format(probability))

# Example usage:
num_tickets = 1000
calculate_chances(num_tickets)
```

```
Number of tickets played: 1000
Total possible outcomes: 13,983,816
Probability of winning with these tickets: 0.0071511238%
```

Less Winning Numbers

In [12]:
```
def calculate_winning_probabilities(ticket_numbers, num_winning_numbers):
    if num_winning_numbers < 2 or num_winning_numbers > 5:
        print("Number of winning numbers must be between 2 and 5.")
        return

    total_possible_outcomes = combinations(49, 6)
    successful_outcomes = combinations(6, num_winning_numbers) * combinations(49 - 6, 6 - num_winning_numbers)

    probability = (successful_outcomes / total_possible_outcomes) * 100

    print("Your selected combination: {}".format(ticket_numbers))
    print("Number of winning numbers expected: {}".format(num_winning_numbers))
    print("Total possible outcomes: {:,}".format(total_possible_outcomes))
    print("Number of successful outcomes: {:,}".format(successful_outcomes))
    print("Probability of having {} winning numbers: {:.10f}%".format(num_winning_numbers, probability))

# Example usage:
user_ticket = [3, 12, 27, 42, 5, 19]
num_winning_numbers = 3
calculate_winning_probabilities(user_ticket, num_winning_numbers)
```

```
Your selected combination: [3, 12, 27, 42, 5, 19]
Number of winning numbers expected: 3
Total possible outcomes: 13,983,816
Number of successful outcomes: 246,820
Probability of having 3 winning numbers: 1.7650403867%
```

Summary

In this project, I developed a set of functions to assist users in understanding their chances of winning in a 6/49 lottery based on various scenarios. The main objectives of the project were:

1. **Factorial and Combinations Functions:** I initially created a `factorial()` function to calculate the factorial of a given number, and a `combinations()` function to calculate the number of combinations for selecting k objects from a group of n objects.
2. **One Ticket Probability Function:** I developed a function named `one_ticket_probability()` that allows users to input a list of six unique numbers, representing a lottery ticket. The function then calculates and prints the probability of winning the big prize with that specific ticket.
3. **Comparing User's Ticket to Historical Data Function:** Next, I created the `compare_ticket()` function, which enables users to compare their chosen combination against historical data of winning numbers. The function prints the number of times the combination occurred in the data and the probability of winning in the next drawing with that combination.
4. **Calculating Chances for Different Number of Tickets Function:** I developed the `calculate_chances()` function, which takes an input representing the number of different tickets the user wants to play. The function calculates and prints the probability of winning the big prize based on the number of tickets played.
5. **Calculating Winning Probabilities Function:** Lastly, I designed the `calculate_winning_probabilities()` function. This function allows users to input their chosen combination of numbers and the number of winning numbers they expect. The function then calculates and prints the probability of having the specified number of winning numbers in the lottery drawing.

These functions collectively provide users with a comprehensive toolset to explore and understand their odds of winning various prizes in a 6/49 lottery. Users can input different combinations, explore different numbers of tickets, and assess their chances of matching a certain number of winning numbers. The project serves as a helpful resource for individuals interested in lottery probabilities and outcomes.

In [ ]:
```
```