Mobile App for Lottery Addiction

In this project, we are 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 us to create the logical core of the app and calculate probabilities. For the first version of the app, they want us to focus on the 6/49 lottery and build functions that can answer users the following questions:

- 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?

The scenario we're following throughout this project is fictional — the main purpose is to practice applying probability and combinatorics (permutations and combinations) concepts in a setting that simulates a real-world scenario.

Main functions

Below we are going to write two main function which it 'll be used frequency

1. Facturial

2. Combinations

In [104…

```
def factoiral(n):
    final_product=1
    for i in range(n,0,-1):
        final_product*=i
    return final_product

def combinations(n,k):
    numerator = factoiral(n)
    denominator = factoiral(k) * factoiral(n-k)
    return numerator/ denominator
```

One-ticket Probability

We 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 us that we need to be aware of the following details when we write 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, we 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 [105…

```python
def one_ticket_probability(user):
    n_combinations = combinations(49,6)
    probability_one_ticket = 1/ n_combinations
    percentage_winner = probability_one_ticket*100
    print('''Your chances to win the big prize with the numbers {} are {:.7f}%.
In other words, you have a 1 in {:,} chances to win.'''.format(user,
                    percentage_winner, int(n_combinations)))
```

We now test a bit the function on two different outputs.

In [106…

```python
test_input_1 = [2, 43, 22, 23, 11, 5]
one_ticket_probability(test_input_1)
```

```
Your chances to win the big prize with the numbers [2, 43, 22, 23, 11, 5] are 0.0000072%.
In other words, you have a 1 in 13,983,816 chances to win.
```

In [107…

```python
test_input_2 = [9, 23, 44, 17, 10, 8]
one_ticket_probability(test_input_2)
```

```
Your chances to win the big prize with the numbers [9, 23, 44, 17, 10, 8] are 0.0000072%.
In other words, you have a 1 in 13,983,816 chances to win.
```

Historical Data Check for Canada Lottery

The institute also wants us 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.

In [108…
```python
import pandas as pd

lottery_canada = pd.read_csv('649.csv')
lottery_canada.shape
```

Out[108…
```
(3665, 11)
```

The data set contains historical data for 3,665 drawings (each row shows data for a single drawing), dating from 1982 to 2018. For each drawing, we can find the six numbers drawn in the following six columns:

- NUMBER DRAWN 1
- NUMBER DRAWN 2
- NUMBER DRAWN 3
- NUMBER DRAWN 4
- NUMBER DRAWN 5
- NUMBER DRAWN 6

In [109…
```python
lottery_canada.head(3)
```

Out[109…

	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 [110…
```python
lottery_canada.tail(3)
```

Out[110…

	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 us that we 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 we'll need 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 our function.
- The engineering team wants us 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.

We're going to 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 [111…

```
lottery_canada.iloc[:,4:11]
```

Out[111…

	NUMBER DRAWN 1	NUMBER DRAWN 2	NUMBER DRAWN 3	NUMBER DRAWN 4	NUMBER DRAWN 5	NUMBER DRAWN 6	BONUS NUMBER
0	3	11	12	14	41	43	13
1	8	33	36	37	39	41	9
2	1	6	23	24	27	39	34
3	3	9	10	13	20	43	34
4	5	14	21	31	34	47	45
...	...	...	...	...	...	...	...
3660	10	15	23	38	40	41	35

	NUMBER DRAWN 1	NUMBER DRAWN 2	NUMBER DRAWN 3	NUMBER DRAWN 4	NUMBER DRAWN 5	NUMBER DRAWN 6	BONUS NUMBER
3661	19	25	31	36	46	47	26
3662	6	22	24	31	32	34	16
3663	2	15	21	31	38	49	8
3664	14	24	31	35	37	48	17

3665 rows × 7 columns

In [113…

```
def extract_numbers(row):
    row = row[4:10]
    row = set(row.values)
    return row
winning_numbers = lottery_canada.apply(extract_numbers,axis=1)
winning_numbers
```

Out[113…

```
0         {3, 41, 11, 12, 43, 14}
1         {33, 36, 37, 39, 8, 41}
2          {1, 6, 39, 23, 24, 27}
3          {3, 9, 10, 43, 13, 20}
4         {34, 5, 14, 47, 21, 31}
                   ...
3660    {38, 40, 41, 10, 15, 23}
3661    {36, 46, 47, 19, 25, 31}
3662     {32, 34, 6, 22, 24, 31}
3663     {2, 38, 15, 49, 21, 31}
3664    {35, 37, 14, 48, 24, 31}
Length: 3665, dtype: object
```

Below, we write the check_historical_occurrence() function that takes in the user numbers and the historical numbers and prints information with respect to the number of occurrences and the probability of winning in the next drawing.

In [114…

```
def check_historical_occurence(user_numbers, historical_numbers):
    '''
    user_numbers: a Python list
    historical numbers: a pandas Series
    '''

    user_numbers_set = set(user_numbers)
    check_occurrence = historical_numbers == user_numbers_set
    n_occurrences = check_occurrence.sum()

    if n_occurrences == 0:
        print('''The combination {} has never occured.
This doesn't mean it's more likely to occur now. Your chances to win the big prize in the next drawing using the comb
In other words, you have a 1 in 13,983,816 chances to win.'''.format(user_numbers, user_numbers))

    else:
        print('''The number of times combination {} has occured in the past is {}.
Your chances to win the big prize in the next drawing using the combination {} are 0.0000072%.
In other words, you have a 1 in 13,983,816 chances to win.'''.format(user_numbers, n_occurrences,
                                                                     user_numbers))
```

In [115…

```
test_input_3 = [39, 26, 31, 39, 8, 45]
check_historical_occurence(test_input_3, winning_numbers)
```

```
The combination [39, 26, 31, 39, 8, 45] has never occured.
This doesn't mean it's more likely to occur now. Your chances to win the big prize in the next drawing using the comb
ination [39, 26, 31, 39, 8, 45] are 0.0000072%.
In other words, you have a 1 in 13,983,816 chances to win.
```

In [116…

```
test_input_4 = [23, 40, 41, 15, 10, 38]
check_historical_occurence(test_input_4, winning_numbers)
```

```
The number of times combination [23, 40, 41, 15, 10, 38] has occured in the past is 1.
Your chances to win the big prize in the next drawing using the combination [23, 40, 41, 15, 10, 38] are 0.0000072%.
In other words, you have a 1 in 13,983,816 chances to win.
```

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 us to be aware of the following details when we're 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).

- Our 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 [117…

```
def multi_ticket_probability(n_tickets):

    n_combinations = combinations(49, 6)

    probability = n_tickets / n_combinations
    percentage_form = probability * 100

    if n_tickets == 1:
        print('''Your chances to win the big prize with one ticket are {:.6f}%.
In other words, you have a 1 in {:,} chances to win.'''.format(percentage_form, int(n_combinations)))

    else:
        combinations_simplified = round(n_combinations / n_tickets)
        print('''Your chances to win the big prize with {:,} different tickets are {:.6f}%.
In other words, you have a 1 in {:,} chances to win.'''.format(n_tickets, percentage_form,
                                                               combinations_simplified))
```

Below we're going to test function with number of tickets.

In [118…

```
test_inputs = [1, 10, 100, 10000, 1000000, 6991908, 13983816]

for test_input in test_inputs:
    multi_ticket_probability(test_input)
    print('________________________')
```

```
Your chances to win the big prize with one ticket are 0.000007%.
In other words, you have a 1 in 13,983,816 chances to win.
________________________
Your chances to win the big prize with 10 different tickets are 0.000072%.
In other words, you have a 1 in 1,398,382 chances to win.
________________________
Your chances to win the big prize with 100 different tickets are 0.000715%.
In other words, you have a 1 in 139,838 chances to win.
________________________
Your chances to win the big prize with 10,000 different tickets are 0.071511%.
In other words, you have a 1 in 1,398 chances to win.
```

```
Your chances to win the big prize with 1,000,000 different tickets are 7.151124%.
In other words, you have a 1 in 14 chances to win.
________________________
Your chances to win the big prize with 6,991,908 different tickets are 50.000000%.
In other words, you have a 1 in 2 chances to win.
________________________
Your chances to win the big prize with 13,983,816 different tickets are 100.000000%.
In other words, you have a 1 in 1 chances to win.
________________________
```

Less Winning Numbers — Function

In most 6/49 lotteries, there are smaller prizes if a player's ticket match two, three, four, or five of the six numbers drawn. This means that players might be interested in finding out the probability of having two, three, four, or five winning numbers — for the first version of the app, users should be able to find those probabilities.

These are the details we need to be aware of when we write a function to make the calculations of those probabilities possible:

- Inside the app, the user inputs:
  - six different numbers from 1 to 49; and
  - an integer between 2 and 5 that represents the number of winning numbers expected
- Our function prints information about the probability of having a certain number of winning numbers

To calculate the probabilities, we tell the engineering team that the specific combination on the ticket is irrelevant and we only need the integer between 2 and 5 representing the number of winning numbers expected. Consequently, we will write a function named probability_less_6() which takes in an integer and prints information about the chances of winning depending on the value of that integer.

The function below calculates the probability that a player's ticket matches exactly the given number of winning numbers. If the player wants to find out the probability of having five winning numbers, the function will return the probability of having five winning numbers exactly (no more and no less). The function will not return the probability of having at least five winning numbers.

In [119…

```
def probability_less_6(n_winning_numbers):

    n_combinations_ticket = combinations(6, n_winning_numbers)
    n_combinations_remaining = combinations(43, 6 - n_winning_numbers)
    successful_outcomes = n_combinations_ticket * n_combinations_remaining
    n_combinations_total = combinations(49, 6)
    probability = successful_outcomes / n_combinations_total

    probability_percentage = probability * 100
    combinations_simplified = round(n_combinations_total/successful_outcomes)
    print('''Your chances of having {} winning numbers with this ticket are {:.6f}%.
In other words, you have a 1 in {:,} chances to win.'''.format(n_winning_numbers, probability_percentage,
                                                               int(combinations_simplified)))
```

Now, let's test the function on all the three possible inputs.

In [120…

```
for test_input in [2, 3, 4, 5]:
    probability_less_6(test_input)
    print('--------------------------') # output delimiter
```

```
Your chances of having 2 winning numbers with this ticket are 13.237803%.
In other words, you have a 1 in 8 chances to win.
--------------------------
Your chances of having 3 winning numbers with this ticket are 1.765040%.
In other words, you have a 1 in 57 chances to win.
--------------------------
Your chances of having 4 winning numbers with this ticket are 0.096862%.
In other words, you have a 1 in 1,032 chances to win.
--------------------------
Your chances of having 5 winning numbers with this ticket are 0.001845%.
In other words, you have a 1 in 54,201 chances to win.
--------------------------
```

Next steps

For the first version of the app, we coded four main functions:

- one_ticket_probability() — calculates the probability of winning the big prize with a single ticket
- check_historical_occurrence() — checks whether a certain combination has occurred in the Canada lottery data set
- multi_ticket_probability() — calculates the probability for any number of of tickets between 1 and 13,983,816
- probability_less_6() — calculates the probability of having two, three, four or five winning numbers exactly

Possible features for a second version of the app include:

- Making the outputs even easier to understand by adding fun analogies (for example, we can find probabilities for strange events and compare with the chances of winning in lottery; for instance, we can output something along the lines "You are 100 times more likely to be the victim of a shark attack than winning the lottery")
- Combining the one_ticket_probability() and check_historical_occurrence() to output information on probability and historical occurrence at the same time
- Create a function similar to probability_less_6() which calculates the probability of having at least two, three, four or five winning numbers. Hint: the number of successful outcomes for having at least four winning numbers is the sum of these three numbers:
  - The number of successful outcomes for having four winning numbers exactly
  - The number of successful outcomes for having five winning numbers exactly
  - The number of successful outcomes for having six winning numbers exactly

In [ ]:

```

```