RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ÉCONOMIE ET MANAGEMENT
ÉCOLE UNIVERSITAIRE DE RECHERCHE

UNIVERSITÉ CÔTE D'AZUR

UNIVERSITE COTE D'AZUR
EUR ELMI
MASTER 1 EXPERTISE ECONOMIQUE

Course: Introduction to Python

PROJECT

Analysis of the impact of the COVID-19 pandemic on the French airline industry using Python

Submitted by :

Mariia PROTASOVA
James KENNEDY

Supervised by:

Olivier CROCE,
Research Engineer - CNRS

Nice, 2022

TABLE OF CONTENTS

INTRODUCTION

Studying the impact of the COVID-19 pandemic on the world economy is an important challenge. The coronavirus pandemic and the measures introduced to prevent the spread of the virus have particularly affected the tourism and passenger transport industries. Using Python, we want to analyze flight data from France in July 2019, 2021, 2022 and draw conclusions about the impact of the pandemic on this industry.

1. BRIEF DESCRIPTION OF THE TASK

In general, it was required to write a code that:

Ex. № 1:

1) compares the number of flights of each day in July 2021 and in July 2019;
2) compares the total number of flights in July 2022 and in July 2019;
3) shows the number of flights on the 14th of July of each year and compares it with other days;
4) compares the number of flights for July 1 and 31 for each year;

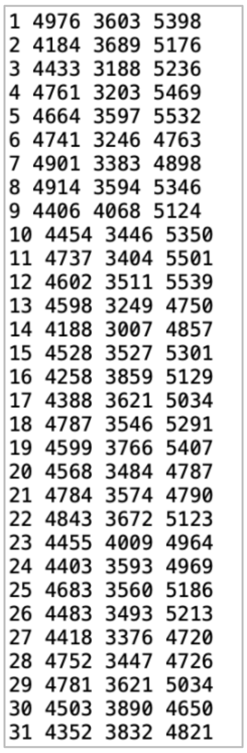
Ex № 2:

5) calculates the minimum, the maximum and the average number of flights for each year;
6) compares the min, the max and the mean;

7) BONUS: to draw a plot to demonstrate the data.

2. INITIAL DATA

The initial data (day, 2022, 2021, 2019) is presented in the box on the right.



```
1 4976 3603 5398
2 4184 3689 5176
3 4433 3188 5236
4 4761 3203 5469
5 4664 3597 5532
6 4741 3246 4763
7 4901 3383 4898
8 4914 3594 5346
9 4406 4068 5124
10 4454 3446 5350
11 4737 3404 5501
12 4602 3511 5539
13 4598 3249 4750
14 4188 3007 4857
15 4528 3527 5301
16 4258 3859 5129
17 4388 3621 5034
18 4787 3546 5291
19 4599 3766 5407
20 4568 3484 4787
21 4784 3574 4790
22 4843 3672 5123
23 4455 4009 4964
24 4403 3593 4969
25 4683 3560 5186
26 4483 3493 5213
27 4418 3376 4720
28 4752 3447 4726
29 4781 3621 5034
30 4503 3890 4650
31 4352 3832 4821
```

3. OUR PLAN

In order to complete these tasks, we created a plan:

- Read the task carefully
- Open the attached data and explore it
- Exercise 1
  - Reorganize the data to be able to work with it
  - Create reusable functions for comparing daily flights and the sum of flights between years

    - Return an output of a function for comparing the number of flights between given days
- Exercise 2
    - Reorganize the initial data again to be able to complete the task.
    - Find the mean, min and max using functions *mean(), max(), min()* *(*import *pandas*).
    - Use the *The if-elif-else Chain* to compare the results and find the smallest (largest) mean.
    - Use the *The if-elif-else Chain* to find the year with the highest and the smallest number of flights.
- Bonus: Plot
    - Decide on the shape(s) of the graphic representation
    - Build a line chart using *plt.plot*, adjust the plot (colors, size and labels) (import *matplotlib)*
    - Build a bar chart using *plt.bar*, adjust the plot (colors, size and labels)
- #Comment all lines of the code
- Write a report commenting the results

TECHNICAL PART

4. BEFORE USING THE CODE

Before using our code, you need to:

- make sure that the library Pandas is installed (otherwise enter "*pip install pandas*" on the terminal)
- make sure that the library Matplotlib is installed (otherwise enter "*pip install matplotlib*" on the terminal)
- make sure that the data file is placed in the same folder as the code.

5. THE CODE

Dear User, our program has been specifically written to answer the questions about COVID impact on the flight industry. However, it may be reutilized to analyze any data that has the similar structure. Please, refer to the section Initial Data above. Using our program doesn't require you to have any technical skills or knowledge of programming in order to be able to get the answers you're looking for. By default, once you run the program, you'll automatically get the answers to the questions that we've outlined above. However, if you want to explore the other possibilities or analyze a different dataset with a similar structure, then you will need to install the proper software (for example, PyCharm) to be able to change the parameters in the functions and get the results that you're looking for.

Please read carefully all the comments we made along the code. They'll provide you with precious information about its functions, without you needing to know how to program. Here we comment again some of the functions we made.

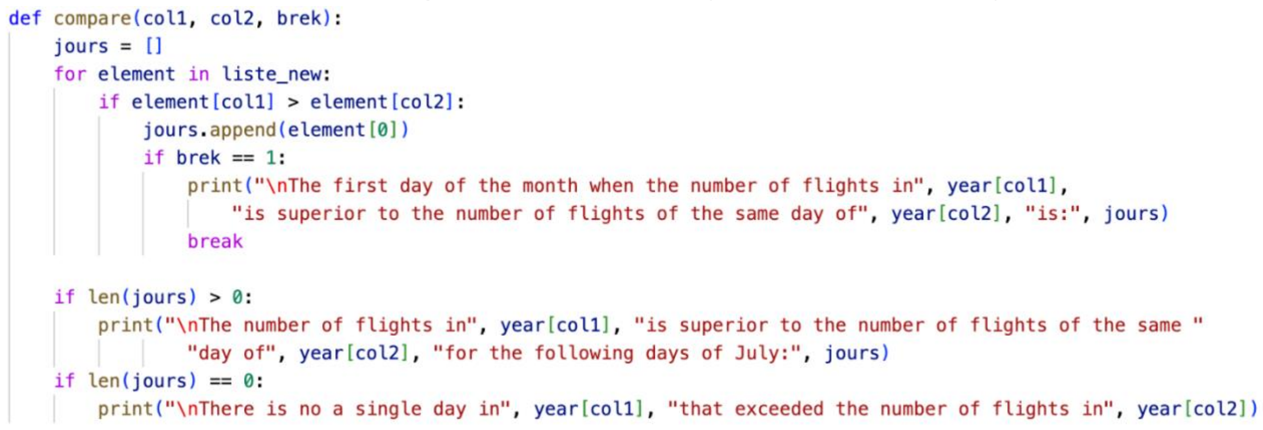
First of all, we import our data, read it and assign it to the variable "data". You need to make sure that your dataset is in the same folder as your Python program and specify the file's name between the brackets.

After reading the dataset, we put each line containing values for three years in a list by removing any unnecessary gaps between lines. We also make sure that our list doesn't have any empty elements by removing everything with a zero length. As a final step, one by one, we split the strings inside the old list, assigning the elements of a particular string to the element_new, which is going to be one of the lists inside a clean dataset. As a result, we get a list of 31 lists, where each list corresponds to a day of July and contains 4 elements : [day, flights of a corresponding day in 2022, 2021, 2019].

Because the Initial Dataset doesn't have any headers, we create a list "year" to specify the years for analysis. The 0 element of the list may take any value, it's there for technical reasons, the elements from 1 to 3 should correspond to the years of the dataset. By default, year has the following structure [1, 2022, 2021, 2019].

If you want to use the program on a different dataset, make sure to change the values in this list to the years you want to analyze. This list is used for automatic referencing.

Finally, we can analyze the data. For this, we create a function that allows us to compare the number of flights of the same day for two different years.



```python
def compare(col1, col2, brek):
    jours = []
    for element in liste_new:
        if element[col1] > element[col2]:
            jours.append(element[0])
            if brek == 1:
                print("\nThe first day of the month when the number of flights in", year[col1],
                    "is superior to the number of flights of the same day of", year[col2], "is:", jours)
                break

    if len(jours) > 0:
        print("\nThe number of flights in", year[col1], "is superior to the number of flights of the same "
                "day of", year[col2], "for the following days of July:", jours)
    if len(jours) == 0:
        print("\nThere is no a single day in", year[col1], "that exceeded the number of flights in", year[col2])
```

col1 and *col2* take values between 1 and 3 which correspond to years from 2022 to 2019 respectively (2022 = 1, 2021 = 2, 2019 = 3).

brek takes values 0 and 1. Introducing this parameter allows us to reuse the same function to answer two different questions.

• brek=1 means "display only the first day of the month where the number of flights for the year [col1] was higher than the number of flights for the same day in the year [col2]".

• brek=0 means "display all days of the month where the number of flights for the year [col1] was higher than the number of flights for the same day in the year [col2]".

For example, to show what was *the first day* in 2021 when the number of flights was higher than in 2019, you just specify the parameters:

```
compare(2, 3, 1)
```
,

where col1 = 2 (2021), col2 = 3 (2019), brek=1.

To make our code easier for the next tasks, we reorganize the data once again, making sure that all the days of July are in one list, while the number of flights for every day of July in a given year is in a separate list. That way, we get a list of four lists with a following structure : [days of July, daily flights in 2022, 2021, 2019].

Thank to this data representation, we create a function that can easily calculate the sum of flights in July in a given year just by adding all the values of a list corresponding to that year and compare it to the sum of flights in a different year. The function has two parameters : *col1* and *col2* take values between 1 and 3 which correspond to years from 2022 to 2019 respectively (2022 = 1, 2021 = 2, 2019 = 3).

For example, to compare the sum of flights in July 2022 and 2019, you simply call the function by inserting the corresponding parameters :

```
compare_sum(1,3)
```

The last part of the first exercise concerns building a function that returns a number of flights on a given day of July in a chosen year and reutilizing it later on for confirming different hypothesis.

The function vols_jours() has two parameters :

year goes from 1 to 3 (where 1 = 2022, 2 = 2021, 3 = 2019) in that order;

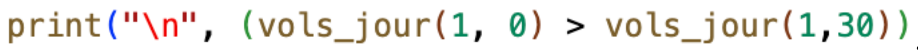
day goes from 0 to 30 which is equivalent to a range from the 1st to 31st July of a given year.

To see if the number of flights on the 14th of July which is the Bastille Day (The French National Day) is usually lower than on the other days, we print the sum of the times when the number of flights was lower than on the 14th of July each year. It is not a function, so it prints a response only for the 14th of July.

```
sum(ref_value < vols_jour(1, 13) for ref_value in short_list[1])
```
,

where *vols_jour(1, 13)* corresponds to the 14th of July 2022 and *short_list[1]* corresponds to the year 2022.

We follow the same idea to test the hypothesis that the number of flights at the beginning of the month is usually higher, but this time we compare the first day and the last day each year. It is not a function, so it prints a response only for the beginning of the month and the end of the month.



```
print("\n", (vols_jour(1, 0) > vols_jour(1,30))
```

,

where *vols_jour(1, 0)* corresponds to the 1st July 2022 and *vols_jour(1,30)* corresponds to the 31st July 2022.

The second part (exercise 2) of our code deals with calculating the means for each year, comparing them, displaying the largest and the smallest means, as well as displaying for each year a day of July with the smallest and the largest number of flights. To do the task, we could have used the same list of lists dataset, but we decided to utilize the Pandas library to demonstrate how much easier it is to manipulate the same dataset with the help of a third-party library and to get the results much faster.

First of all, we import the Pandas library and read the dataset line by line. We specify that our initial dataset doesn't have a header (header=None), assign each element of a line to a separate column by specifying that the elements should be separated by a simple space (sep=“ ”), eventually we also name the columns for a convenient data accessing.

If you want to use the program on a different dataset, make sure to rename the columns according to the years you want to analyze. This header is used for automatic referencing.

To calculate the minimum, the maximum and the mean, all we need to do is to specify the name of the column that contains the number of flights in a particular year, to use a command with a name of the statistics as well as to round up the result. For example, to get the mean number of flights in 2022, we would use the following command : print(round(lines["2022"].mean()). In our case, the program calculates and displays all the required statistics (min, max, mean) for three years.

To find the largest and the smallest means we created two functions *lmean(m22, m21, m19)* and *smean(m22, m21, m19)* that compare the means and display the result. We could have achieved the same result by assigning the means to a list and displaying the largest element of the list, which would give us a much shorter code. But we decided not to combine different libraries.

m22, m21, m19 correspond to the previously calculated mean number of flights in 2022, 2021, 2019. The values have already been assigned to the variables and the function will automatically display the largest and the smallest result as well as the corresponding year.

Finally, we create two variables (*minmin, maxmax*) that contain the smallest and the largest number of flights in the given years.

```
minmin = (min((lines["2022"].min()), (lines["2021"].min()), (lines["2019"].min())))
if minmin == lines["2022"].min():
    y = 1
elif minmin == lines["2021"].min():
    y = 2
else:
    y = 3
print("\nThe July day with the least number of flights was in", lines.columns[y])
```

We compare these numbers with the smallest and the largest number of flights in a particular year. Once we have a match between the smallest (the largest) number of flights and a particular year, we display the corresponding year. It happens automatically based on your dataset, so you don't have to modify the code.

The last part of the analysis program plots two different graphs for a visual representation of the dataset. It automatically creates the graphs for three years based on a given data and doesn't require you to make any modifications.

```
#line chart
plt.style.use('bmh')
plt.plot(short_list[0],short_list[1], 'o-', color= '#864ecb', label = year[1], ms=4)
plt.plot(short_list[0],short_list[2], 'o-', color ='purple', label = year[2], ms=4)
plt.plot(short_list[0],short_list[3], 'o-', color ='#3da8f4', label = year[3], ms=4)
plt.title("The number of flights for each day in July 2019, 2021, 2022")
plt.legend()
plt.xlabel("Day of July")
plt.ylabel("Number of flights")
plt.show()
```

6. OUTPUT AND COMMENTS ON THE RESULTS

The final output is presented below.

```
Hello Mr. CROCE.

There is no a single day in 2021 that exceeded the number of flights in 2019

The number of flights in 2022 is superior to the number of flights of the same day of 2019 for the following days of July: [7, 28]

The total number of flights in 2022 is lower than in 2019

Number of flights [day of July, 2022, 2021, 2019] respectively = [14, 4188, 3007, 4857]

During July 2022 only 1 days had less flights than the 14th July
During July 2021 only 0 days had less flights than the 14th July
During July 2019 only 8 days had less flights than the 14th July

 True that the number of flights at the beginning of July 2022 is more than at the end
False that the number of flights at the beginning of July 2021 is more than at the end
True that the number of flights at the beginning of July 2019 is more than at the end

        Statistics July 2022:
Mean: 4585      Minimum: 4184   Maximum: 4976

        Statistics July 2021:
Mean: 3550      Minimum: 3007   Maximum: 4068

        Statistics July 2019:
Mean: 5099      Minimum: 4650   Maximum: 5539

The largest mean in July was in 2019 and it was equal to 5099 flights

The smallest mean in July was in 2021 and it was equal to 3550 flights

The July day with the least number of flights was in 2021

The July day with the highest number of flights was in 2019
```

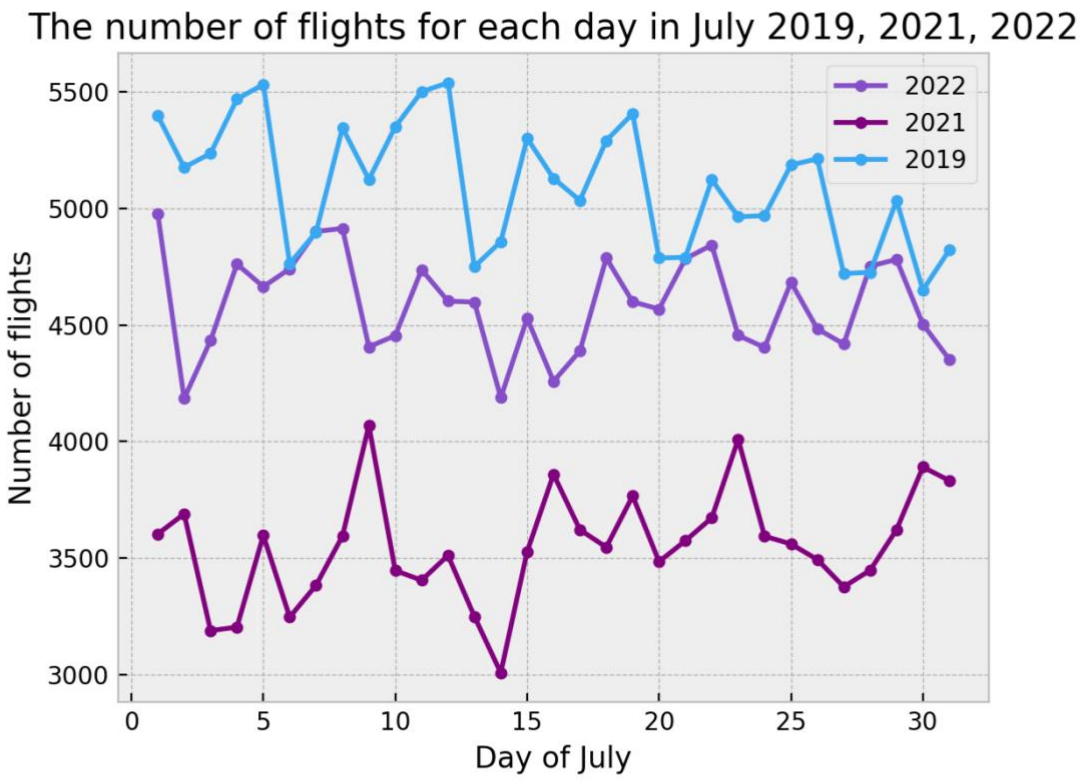
Looking at the dynamics, the French airline industry is gradually recovering. However, the number of flights in July 2022 has not reached the pre-pandemic level (July 2019). This is evidenced by the fact that the total number of flights in July 2019 exceeds the total number of flights in July 2022, as well as the day with the highest number of flights was in 2019.

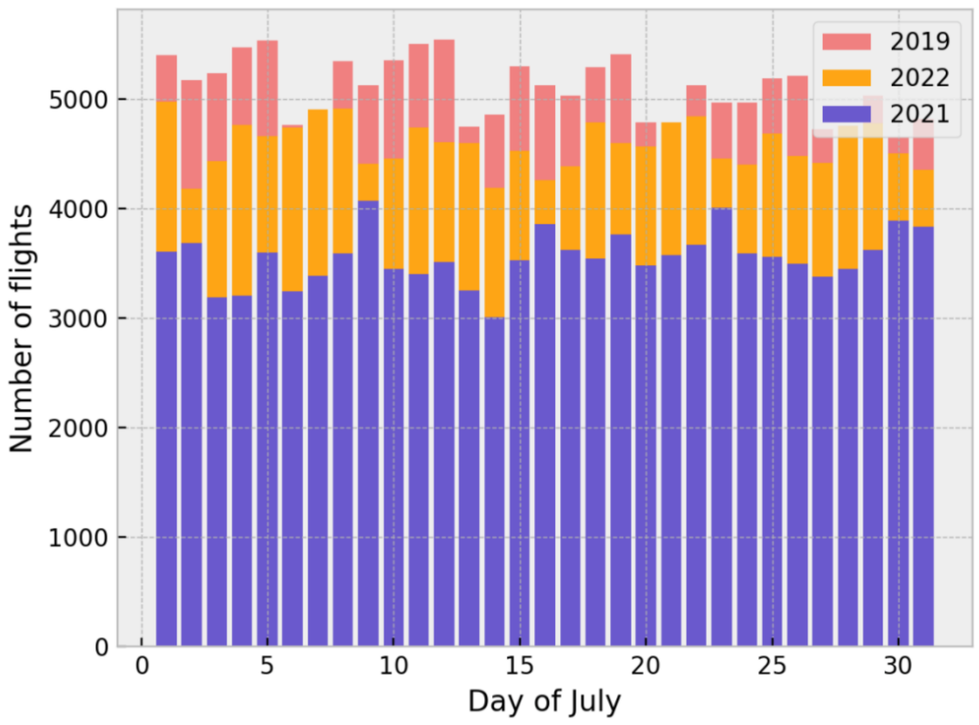
The statistics show that there are usually fewer flights on July 14 than on other days in July. We can assume that the number of flights is lower because of the day off (fewer people working and servicing airports), and that people tend to spend the national holiday in France (lower demand).

An analysis of the number of flights on the 1st and 31st of July showed the following results. In 2022 and 2019, there were more flights at the beginning of the month than at the end. However, in 2021, there were more flights on the last day of the month than on the first day.

Based on the three years it is impossible to unequivocally assert the existence of any pattern, but we can assume that the anomaly in 2021 is explained by the influence of the coronavirus pandemic.

Finally, the graphical representation of the dataset below allows us to succinctly sum up our results in a simple, yet informative way, making it easier for anyone to see a full picture with just a few lines.




2019
2022
2021
Number of flights
0
1000
2000
3000
4000
5000
Day of July
0
5
10
15
20
25
30