Viviana SANCHES RODRIGUES RIBEIRO
viviana.sanches-rodrigues-ribeiro@etu.univ-cotedazur.fr

PYTHON PROJECT

The purpose of this project is to compare observations produced from a sample of flights departing from France.

Data base

For this project, we recovered a database from the site:
https://www.eurocontrol.int/Economics/DailyTrafficVariation-States.html.
But it was reduced and modified beforehand to be manipulable as we saw in class.
(the file was cleaned and put in txt format and this is only a sample of the real base)
So use the database provided in the project's zip file!

When you open the txt file, you will see that you have:

the days of the month	the number of flights for the year 2022	the number of flights for the year 2021	the number of flights for the year 2019

You understand that you will need to measure the impact of COVID-19 on flights (between 2019 and 2021) and then comment on the situation in 2022 (recovery or not and by how much).
All this will be done with some exercises.

Viviana SANCHES RODRIGUES RIBEIRO
viviana.sanches-rodrigues-ribeiro@etu.univ-cotedazur.fr

Setting up

1. Data recovery

To start, you will need to retrieve the information contained in the text file.
As seen in class, you will have to use the functions:
open(path**)**, **readlines(**file**)** ou **readline(**file**)** (optional)

2. Data reorganization

As seen in class, the data are grouped line by line in the list, so they are not in a usable form for the work below.
You will have to separate them and organize them with methods such as list.**split()**, list.**append()** (or list = list + [object]) or functions like **del** list[index].
In addition, you will see that the elements are strings, so you will have to transform them into numbers with the command **int(**object**)**

Note: you can

- either put all lines in a list and you will have a large list containing sub-lists for each day ;
- or put each line in a list and you will have a total of 31 lists.

This will depend on which function you have chosen to extract the data.

Now, that the data is organized in the list or lists, you can start the exercises.

Exercise 1 - Comparison

As previously announced, this is a data comparison exercise.
Your code should answer the questions and instructions below.

Is there a day when the number of flights in 2021 exceeded the number of flights in 2019?
Note: you can make a **for** loop containing an **if** loop that would stop on the first day that answers the question.
If not, the function must still return a response.
In general, do you find that flights are more important in 2022 than in 2019?
Note: this time, we must see all the cases where this took place.

On 14 July of each year, how many flights left France?
What do you notice?

Write a function that returns the number of flights for July 1 and 31 and comment.
Note: are there any points to note at the beginning and end of the month?

Viviana SANCHES RODRIGUES RIBEIRO
viviana.sanches-rodrigues-ribeiro@etu.univ-cotedazur.fr

Exercise 2 - Minimums, Maximums, Averages

Write a function or a script that returns the min, the max, and the average number of flights for each year.

For this exercise, you will use the functions **min(**object**), max(**object**)**, **mean(**object**)**.

Note: For each result, make sure to display it with a sentence.
Use the **print(**object**)** function or the **str(**object**)** function with the "**+**" operator.

Write a function or script that compares previously calculated information.
Note: you must not comment on the results yourself. It is up to you to return the correct comments.

Exercise - Graphics

This is a bonus exercise, if you are interested in graphic representations, do not hesitate.
For this exercise, you will need to import the pyplot module from the matplotlib library.
Use import **matplotlib.pyplot as plt**
You will need to:

- define a list containing the data on the x-axis (from 1 to 31)
- define 3 lists each containing data for a single year (if not done before)

You will use the commands:
plt.plot(x,y**)**
plt.show()

You can have fun customizing your graph with the commands and functions below (there are many others, this is just a preview).

plot(x,y, **parameters**)

c = 'color' ('r' red, 'b' blue, 'c' cyan, 'g' green, 'k' black)
marker = 'style de point' ('o' point, '+' plus, 'x' croix)
ls = 'line style between dots' ('-' continuous line, '--' dashed, ':' small dots)

These parameters can be written together. Example: plt.plot(x,y, 'or-'), displays a graph with round dots, connected by a line and all in red.

ms = dot size (number)
lw = line width (number)
label = 'title that is associated with the curve'

Viviana SANCHES RODRIGUES RIBEIRO
viviana.sanches-rodrigues-ribeiro@etu.univ-cotedazur.fr

after the plot:

plt.title("title of the graph")
plt.legend() display the legend that was previously defined with the label parameter

plt.xlabel(axis name)
plt.ylabel(axis name)

xlim(xmin, xmax) et ylim(ymin, ymax)