Python for Data Analysis
ESILV 2023–2024
[Practical work 4: Matplotlib]

Matplotlib

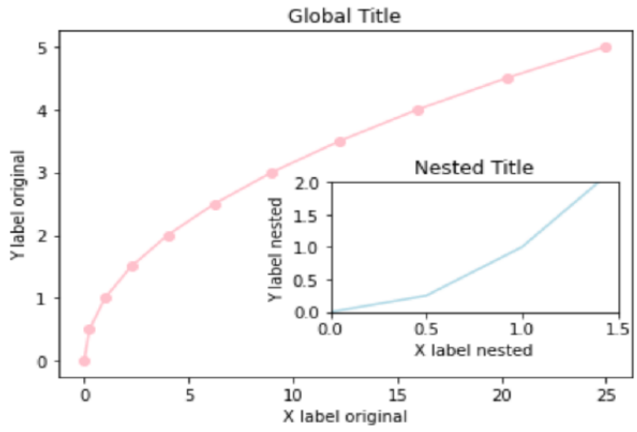
Exercise 1 [Matplotlib basics]

1. Create '*x'* and 'y' NumPy arrays, such as 'x' have values from 1 to 5, with a 0.5 step. '*y'* will be the square of '*x'*. Then, create a plot using '*x'* and '*y'*.
2. On the same plot, add a title for the graph, x axis and y axis.
3. Change the color of the line. Where can you find all named colors known by matplotlib?
4. Create a multiplot having 2 rows and 2 columns. Such as:
    a. First plot containing $y=x^2$
    b. Second plot containing $y=x^3$
    c. (Second row) Third plot containing y=log(x)
    d. Fourth plot containing $y=e^x$

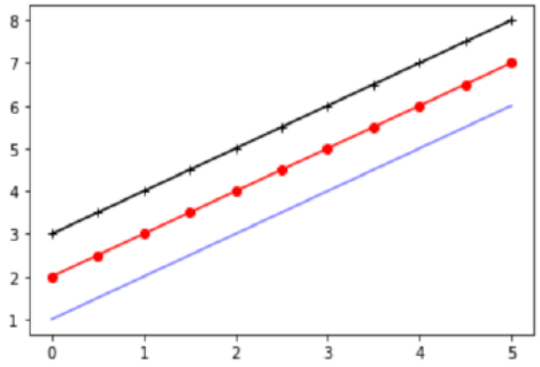
Exercise 2 [Matplotlib's Object Oriented API]

Matplotlib have an object-oriented API used to create more complex figures. Code is a little more complicated, but the advantage is that we now have full control of where the plot axes. Here's a good read when exploring matplotlib https://github.com/rougier/matplotlibtutorial

1. Create the same chart as 1.1 using a new method: instantiate figure objects and then call methods or attributes from that object.
2. Create the figure below. (The blue curve represents $y = x^2$, while the red curve represents $x = y^2$).



3. Explore colors and markers style. Create a new chart, such as x goes from 0 to 10, and y = x + *index*. *Index* is the number of curves you are going to draw. For example, the first curve will match y = x +0, the second y = x + 1, etc.
   Create multiple combinations of different colors, markers & marker size in this graphic. Here's an example using three curves:



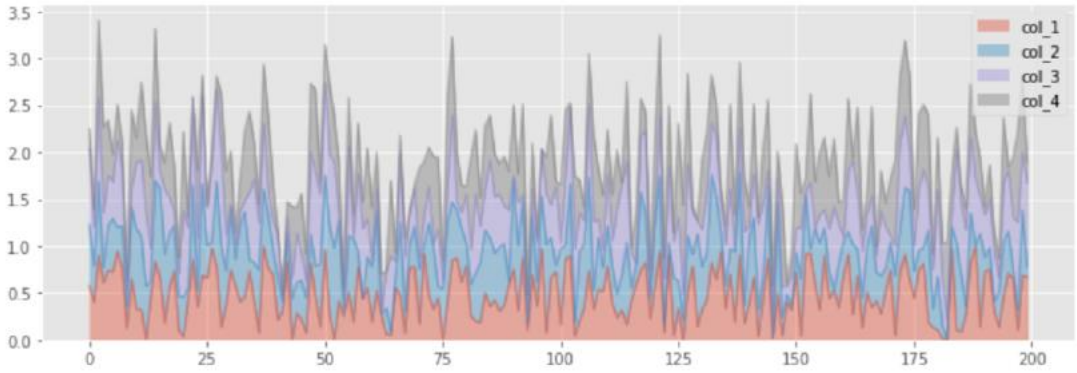
4. Create the same subplots you've created in 1.4. using Matplotlib API (use *plt.suplots())*
   a. Add a title for each plot 'title 1', 'title 2', etc.
   b. Add an x axis and y axis label for each plot (for example, x-axis: *X*, y-axis: *f(x)= x^2*
   c. Use *figsize* to make it clearer
   d. Show legend on chart
   e. Differentiate between each graph using a different color and markers

Exercise 3 [Pandas built-in]

On every plot graph, you can use plt.style.use(*theme*) to change the style of your plot.
https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html

You can find all available graphs on pandas here:
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.plot.html

1. Create a pandas dataframe, having 5000 rows and 4 random columns. First column should be called 'col_1', second one 'col_2', etc.
2. Using pandas built-in visualization, create a scatter plot that will show col_1 vs col_2. Try it again with col_3 vs col_4.
   Hint: use df.plot.scatter
3. On the first plot, add col_3 as a color column, and col_4 as a size column. This means that the more col_3 is increased, the more the color will be different. And the more col_4 increases, the bigger its size will be.
   *Hint: if the size is very small, consider multiply it by * 10, even by 100 if needed.*
4. Display the frequency bar (histogram) of the first column. As you can see, we can't identify anything using the default plot configuration. Can you refine the result so that we can see more detail in the plot?
5. Create a boxplot representing the distribution of the 4 columns on the same graph. Can you display the first two only?
6. Plot the first 200 rows of the dataframe.



7. Create two plots side by side, the first one is using scatter plot (as seen previously) and the second one use hexbin plot.