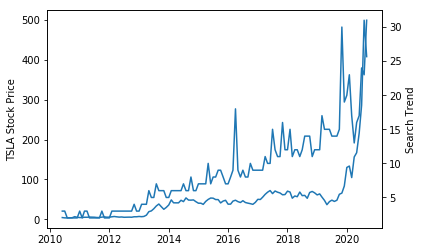
**Data Visualisation - Tesla Line Charts in Matplotlib**

Let's create a basic line chart of the Tesla stock price and the search popularity and then gradually add more and more styling to our chart.

Challenge

Plot the Tesla stock price against the Tesla search volume using a line chart and two different axes. Here's what you're aiming for:



.

.

..

...

..

.

.

**Solution: Creating a basic chart**

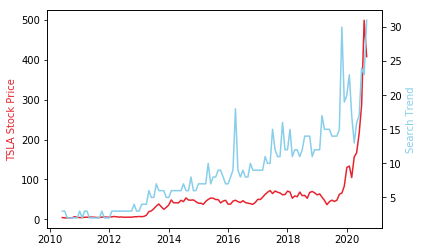
This bit should pretty much be review from the previous days' lessons. To create a line plot with two different y-axes we first have to get the current axis and make a copy of it using .twinx(). Then we can configure each axis separately and call .plot().

1. ax1 = plt.gca() # get current axis
2. ax2 = ax1.twinx()
4. ax1.set\_ylabel('TSLA Stock Price')
5. ax2.set\_ylabel('Search Trend')
7. ax1.plot(df\_tesla.MONTH, df\_tesla.TSLA\_USD\_CLOSE)
8. ax2.plot(df\_tesla.MONTH, df\_tesla.TSLA\_WEB\_SEARCH)

Challenge

Now let's style the chart a bit more. In particular, let's check out the different colours you can use with Matplotlib.

For our updated chart, let's differentiate the two lines and the axis labels using different colours. Try using one of the blue [colour names](https://matplotlib.org/3.1.1/gallery/color/named_colors.html) for the search volume and a [HEX code](https://htmlcolorcodes.com/color-picker/) for a red colour for the stock price. Here's what you're aiming for:



Hint: you can colour both the [axis labels](https://matplotlib.org/3.3.2/api/text_api.html#matplotlib.text.Text) and the [lines](https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.lines.Line2D.html#matplotlib.lines.Line2D) on the chart using keyword arguments (kwargs).

.

.

..

...

..

.

.

**Solution: Adding colours**

Your code should now look something like this (with your own choice of colours of course):

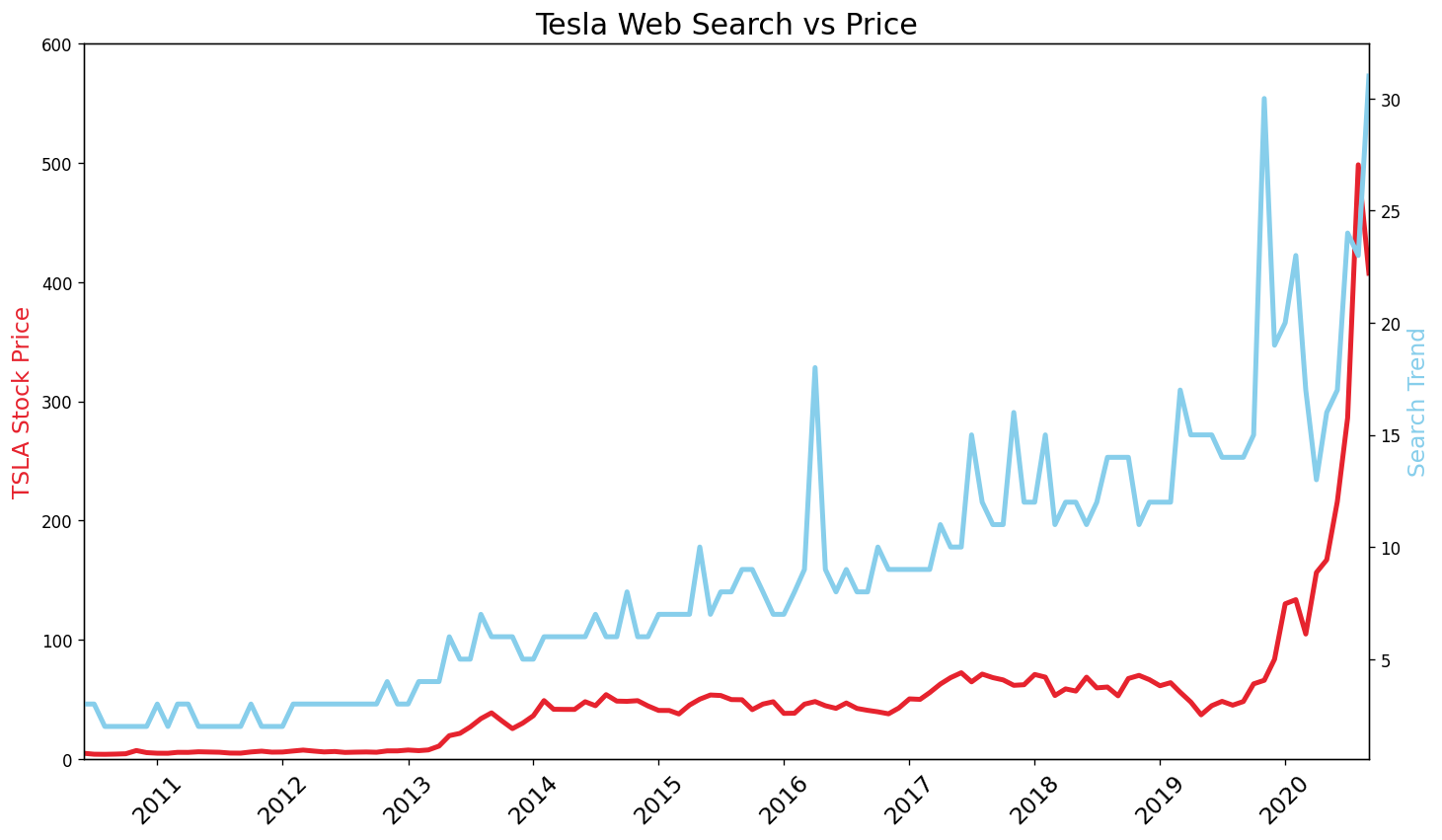
1. ax1 = plt.gca()
2. ax2 = ax1.twinx()
4. ax1.set\_ylabel('TSLA Stock Price', color='#E6232E') # can use a HEX code
5. ax2.set\_ylabel('Search Trend', color='skyblue') # or a named colour
7. ax1.plot(df\_tesla.MONTH, df\_tesla.TSLA\_USD\_CLOSE, color='#E6232E')
8. ax2.plot(df\_tesla.MONTH, df\_tesla.TSLA\_WEB\_SEARCH, color='skyblue')

Challenge

There are still some ways to improve the look of this chart. First off, let's make it larger. Can you make the following changes:

1. Increase the figure size (e.g., to 14 by 8).
2. Increase the font sizes for the labels and the ticks on the x-axis to 14.
3. Rotate the text on the x-axis by 45 degrees.
4. Add a title that reads 'Tesla Web Search vs Price'
5. Make the lines on the chart thicker.
6. Keep the chart looking sharp by changing the dots-per-inch or [DPI value](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.figure.html).
7. Set minimum and maximum values for the y and x-axis. Hint: check out methods like [set\_xlim()](https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.axes.Axes.set_xlim.html).
8. Finally use [plt.show()](https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.show.html) to display the chart below the cell instead of relying on the automatic notebook output.

After the changes your chart should then look something like this:



.

.

..

...

..

.

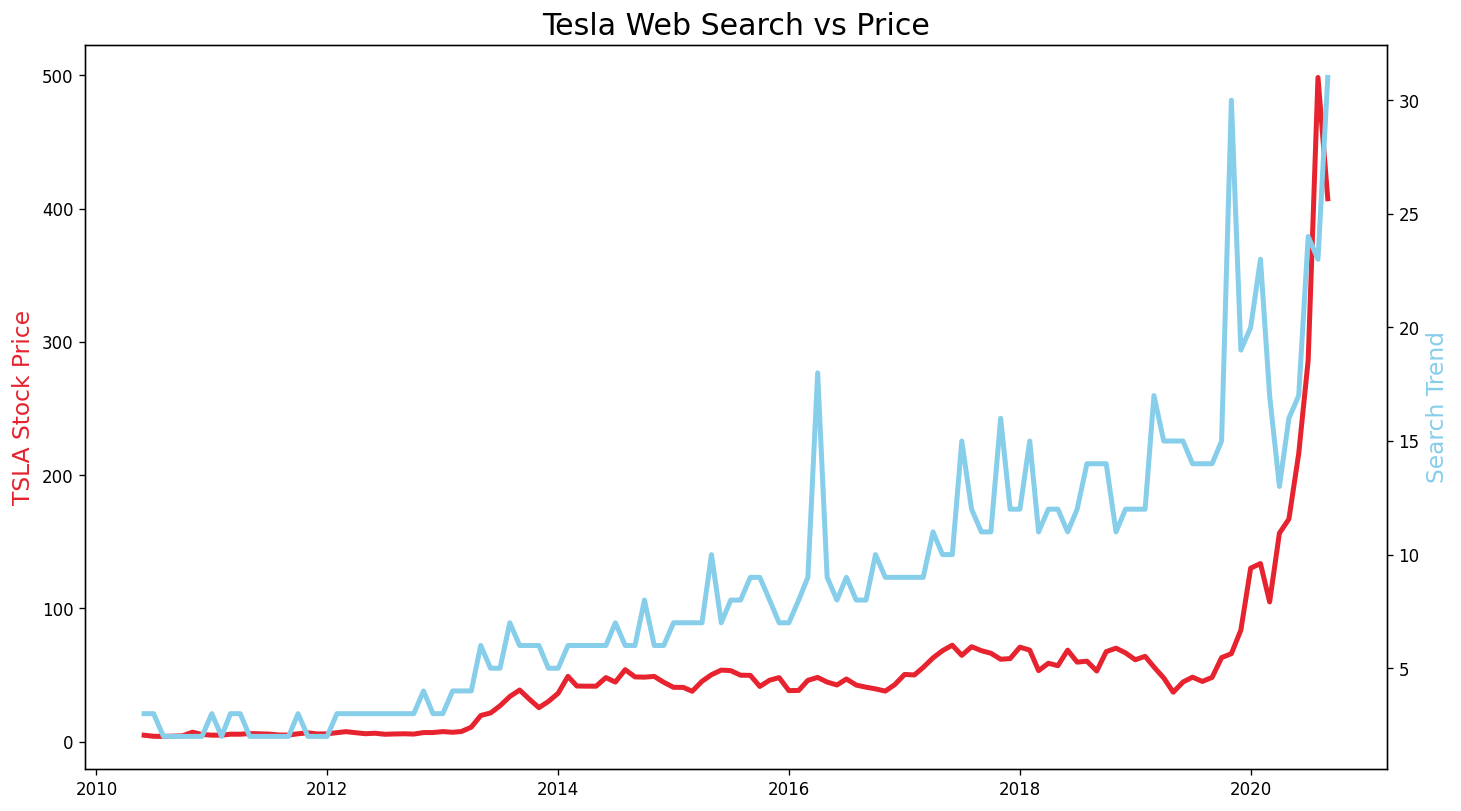
.

**Solution: Additional styling, increasing size & resolution**

There's a couple of tweaks to the code going on here. First, we use .figure() to increase the size and resolution of our chart. Since we now have a bigger chart, we should also increase the font size of our labels and the thickness of our lines.

Finally, we are calling .show() to explicitly display the chart below the cell. This .show() method is important to be aware of if you're ever trying to generate charts in PyCharm or elsewhere outside of an interactive notebook like Google Colab or Jupyter. Also, it gives our notebook a very clean look.

1. # increases size and resolution
2. plt.figure(figsize=(14,8), dpi=120)
3. plt.title('Tesla Web Search vs Price', fontsize=18)
5. ax1 = plt.gca()
6. ax2 = ax1.twinx()
8. # Also, increase fontsize and linewidth for larger charts
9. ax1.set\_ylabel('TSLA Stock Price', color='#E6232E', fontsize=14)
10. ax2.set\_ylabel('Search Trend', color='skyblue', fontsize=14)
12. ax1.plot(df\_tesla.MONTH, df\_tesla.TSLA\_USD\_CLOSE, color='#E6232E', linewidth=3)
13. ax2.plot(df\_tesla.MONTH, df\_tesla.TSLA\_WEB\_SEARCH, color='skyblue', linewidth=3)
15. # Displays chart explicitly
16. plt.show()



Here's the code with rotation added to the x-ticks. With .set\_ylim() and .set\_xlim() you have precise control over which data you want to show on the chart. You can either choose hard values like displaying the Tesla stock price between $0 and $600. Or you could use the .min() and .max() functions to help you work out the bounds for the chart as well.

1. plt.figure(figsize=(14,8), dpi=120)
2. plt.title('Tesla Web Search vs Price', fontsize=18)
4. # Increase the size and rotate the labels on the x-axis
5. plt.xticks(fontsize=14, rotation=45)
7. ax1 = plt.gca()
8. ax2 = ax1.twinx()
10. ax1.set\_ylabel('TSLA Stock Price', color='#E6232E', fontsize=14)
11. ax2.set\_ylabel('Search Trend', color='skyblue', fontsize=14)
13. # Set the minimum and maximum values on the axes
14. ax1.set\_ylim([0, 600])
15. ax1.set\_xlim([df\_tesla.MONTH.min(), df\_tesla.MONTH.max()])
17. ax1.plot(df\_tesla.MONTH, df\_tesla.TSLA\_USD\_CLOSE, color='#E6232E', linewidth=3)
18. ax2.plot(df\_tesla.MONTH, df\_tesla.TSLA\_WEB\_SEARCH, color='skyblue', linewidth=3)
20. plt.show()

**Fix the Matplotlib Warning (if you see it)**

At this point, you might have seen this warning below from Matplotlib:



This is not an error, but an FYI to be explicit about which datetime converter to use. We have a timeline on our x-axis after all. To address this simply follow the instructions in the warning message and add the following code:

