

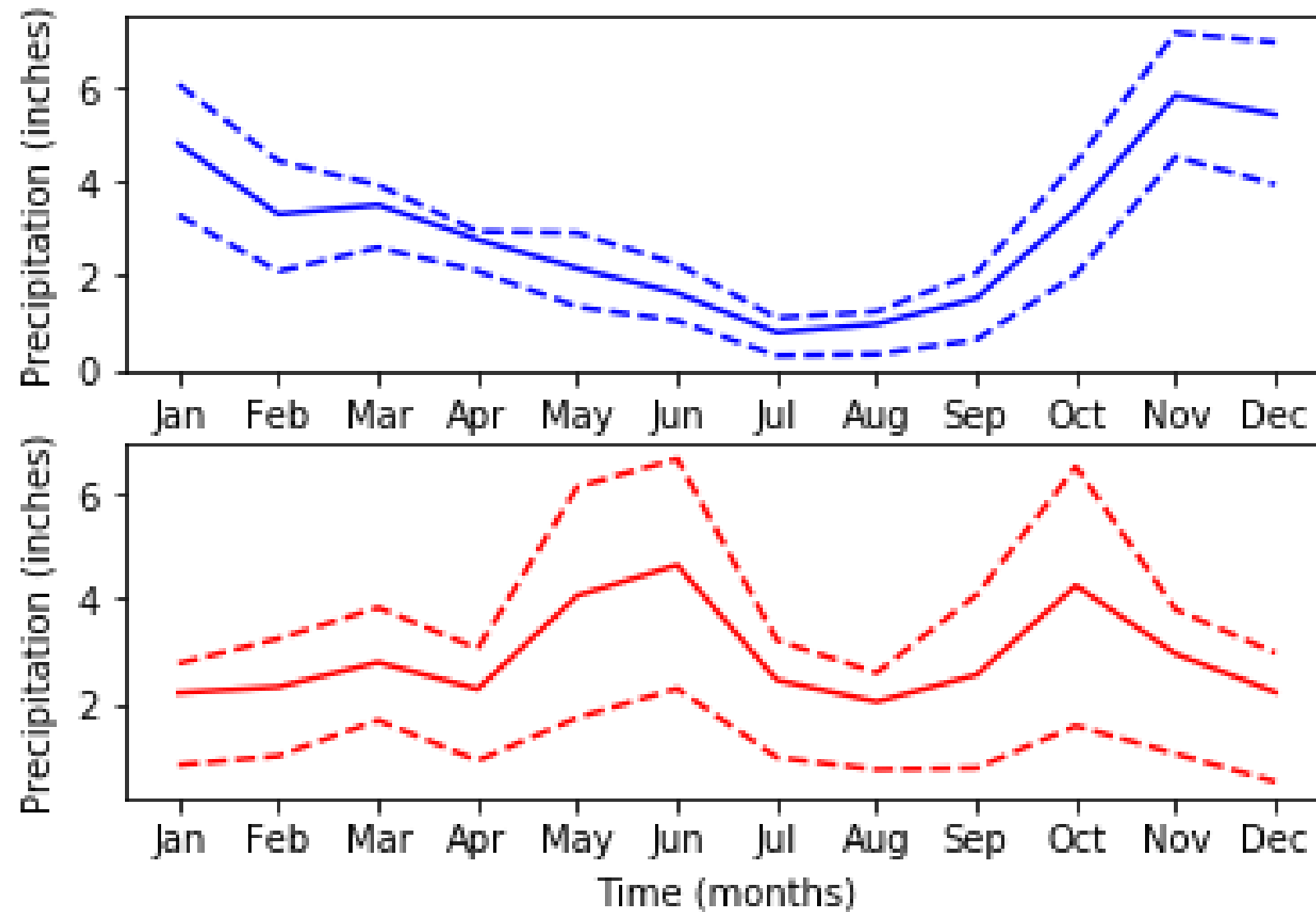
# Plotting time-series data

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB



**Ariel Rokem**  
Data Scientist

# Time-series data



# Climate change time-series

```
date,co2,relative_temp
1958-03-06,315.71,0.1
1958-04-06,317.45,0.01
1958-05-06,317.5,0.08
1958-06-06,-99.99,-0.05
1958-07-06,315.86,0.06
1958-08-06,314.93,-0.06
...
2016-08-06,402.27,0.98
2016-09-06,401.05,0.87
2016-10-06,401.59,0.89
2016-11-06,403.55,0.93
```

A measurement was taken on the 6th day of every month from 1958 to 2016.

```
climate_change =
pd.read_csv('climate_change.csv',
parse_dates = ['date'], index_col =
['date'])
```

# DateTimeIndex

```
climate_change.index
```

```
DatetimeIndex(['1958-03-06', '1958-04-06', '1958-05-06', '1958-06-06',  
              '1958-07-06', '1958-08-06', '1958-09-06', '1958-10-06',  
              '1958-11-06', '1958-12-06',  
              ...  
              '2016-03-06', '2016-04-06', '2016-05-06', '2016-06-06',  
              '2016-07-06', '2016-08-06', '2016-09-06', '2016-10-06',  
              '2016-11-06', '2016-12-06'],  
              dtype='datetime64[ns]', name='date', length=706, freq=
```

# Time-series data

```
climate_change['relative_temp']
```

```
0      0.10
1      0.01
2      0.08
3     -0.05
4      0.06
5     -0.06
6     -0.03
7      0.04
...
701     0.98
702     0.87
703     0.89
704     0.93
705     0.81
Name:co2, Length: 706, dtype: float64
```

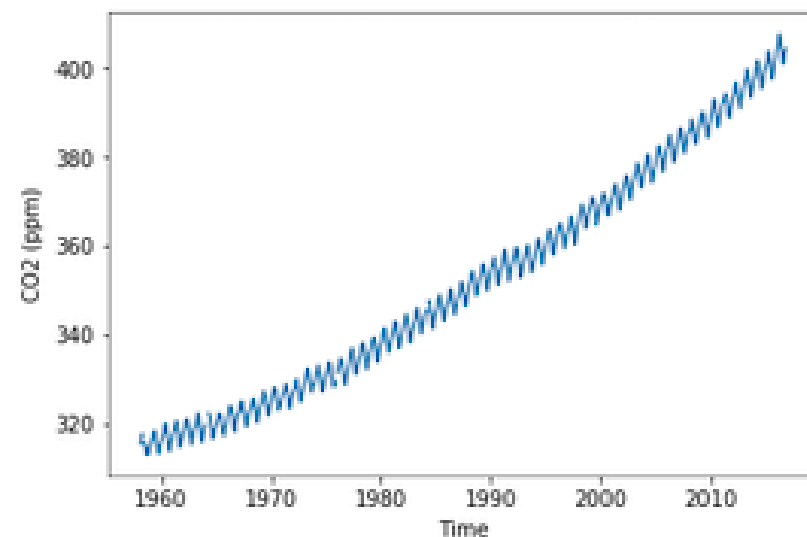
```
climate_change['co2']
```

```
0      315.71
1      317.45
2      317.50
3         NaN
4      315.86
5      314.93
6      313.20
7         NaN
...
701     402.27
702     401.05
703     401.59
704     403.55
705     404.45
Name:co2, Length: 706, dtype: float64
```

# Plotting time-series data

```
import matplotlib.pyplot as plt  
fig, ax = plt.subplots()
```

```
ax.plot(climate_change.index, climate_change['co2'])  
ax.set_xlabel('Time')  
ax.set_ylabel('CO2 (ppm)')  
plt.show()
```

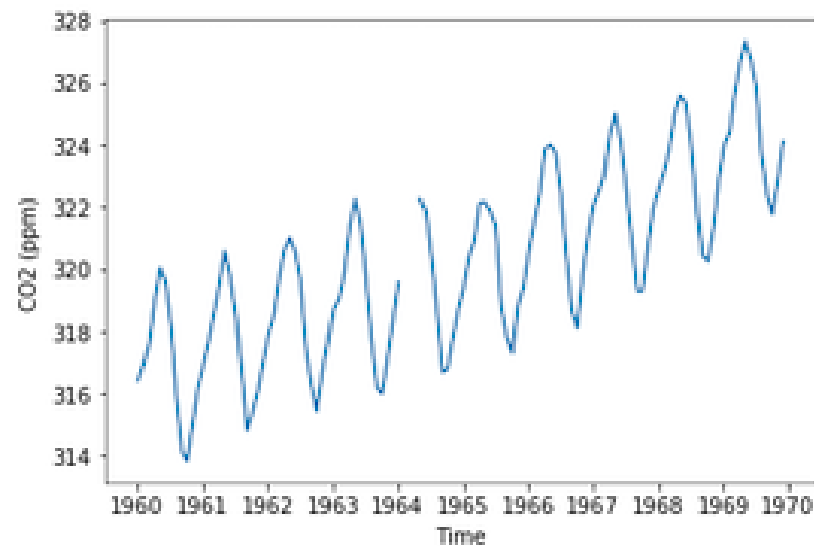


There are some small seasonal fluctuations in the amount of CO2 measured, and an overall increase in the amount of CO2 in the atmosphere from about 320 parts per million to about 400 parts per million.

# Zooming in on a decade

```
sixties = climate_change["1960-01-01":"1969-12-31"]
```

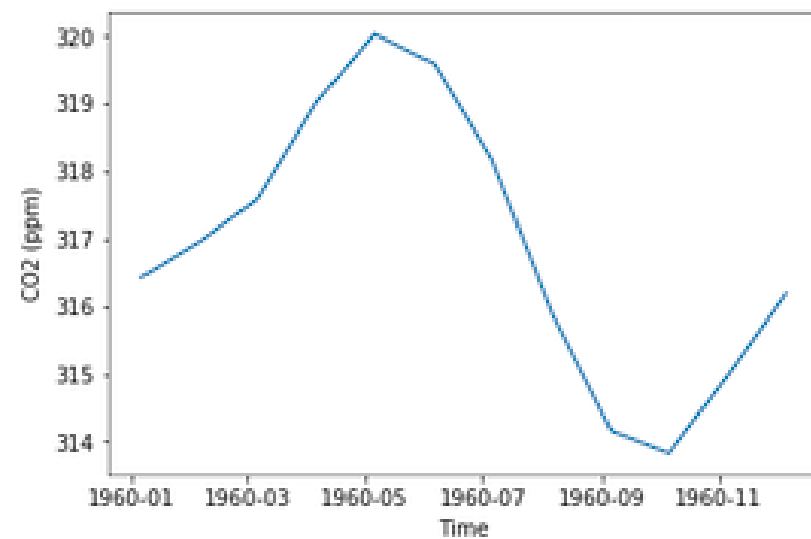
```
fig, ax = plt.subplots()
ax.plot(sixties.index, sixties['co2'])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)')
plt.show()
```



The missing values in this time series are represented as breaks in the line plotted by Matplotlib.

# Zooming in on one year

```
sixty_nine = climate_change["1969-01-01":"1969-12-31"]  
  
fig, ax = plt.subplots()  
ax.plot(sixty_nine.index, sixty_nine['co2'])  
ax.set_xlabel('Time')  
ax.set_ylabel('CO2 (ppm)')  
plt.show()
```





# Let's practice time-series plotting!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

# Plotting time-series with different variables

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB



**Ariel Rokem**  
Data Scientist

# Plotting two time-series together

```
import pandas as pd
climate_change = pd.read_csv('climate_change.csv',
                             parse_dates=["date"],
                             index_col="date")
```

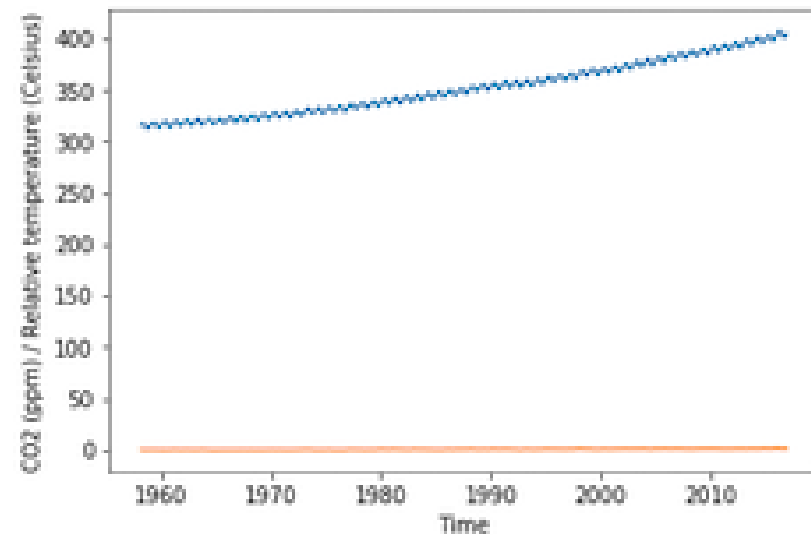
```
climate_change
```

```
      co2  relative_temp
date
1958-03-06  315.71         0.10
1958-04-06  317.45         0.01
1958-07-06  315.86         0.06
...      ...         ...
2016-11-06  403.55         0.93
2016-12-06  404.45         0.81
```

```
[706 rows x 2 columns]
```

# Plotting two time-series together

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"])
ax.plot(climate_change.index, climate_change["relative_temp"])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm) / Relative temperature')
plt.show()
```

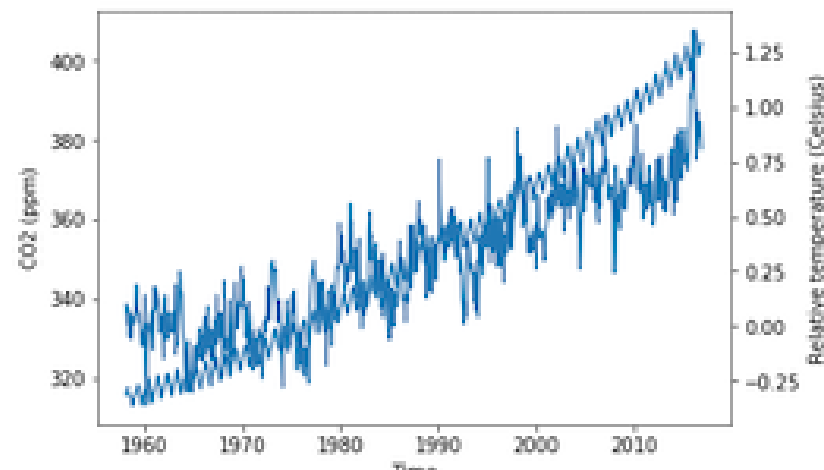


# Using twin axes

```
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"])
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)')

ax2 = ax.twinx()

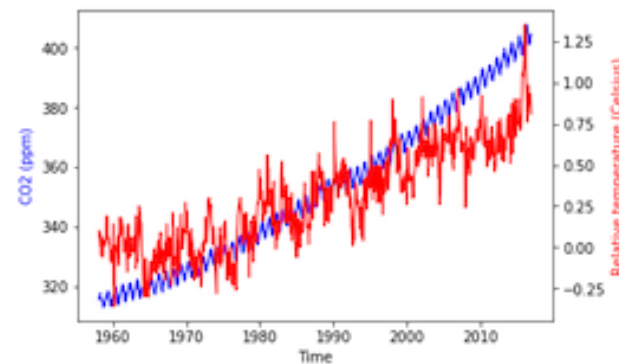
ax2.plot(climate_change.index, climate_change["relative_temp"])
ax2.set_ylabel('Relative temperature (Celsius)')
plt.show()
```



# Separating variables by color

```
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"], color='blue')
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)', color='blue')

ax2 = ax.twinx()
ax2.plot(climate_change.index, climate_change["relative_temp"],
         color='red')
ax2.set_ylabel('Relative temperature (Celsius)', color='red')
plt.show()
```



# Coloring the ticks

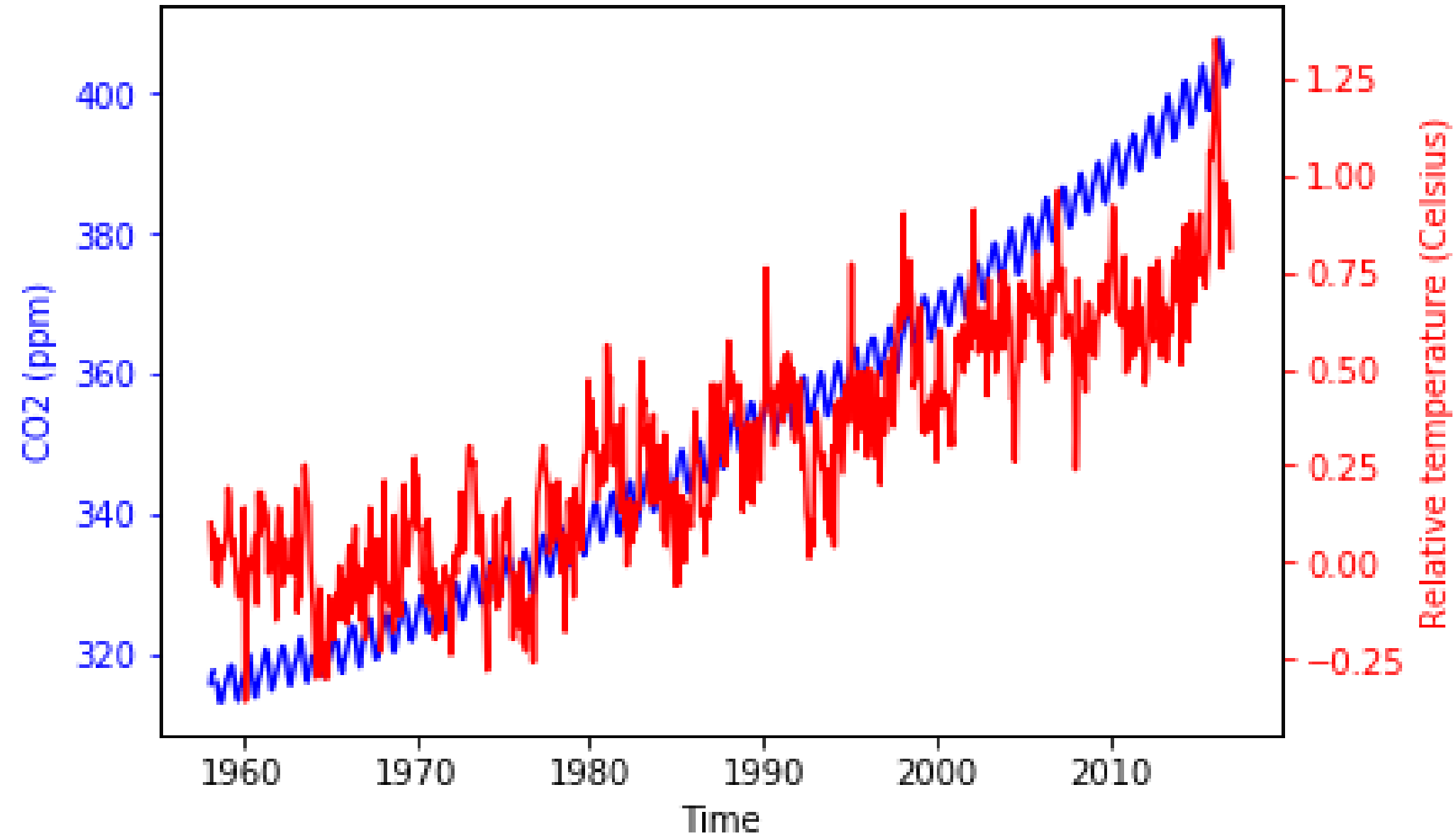
```
fig, ax = plt.subplots()
ax.plot(climate_change.index, climate_change["co2"],
        color='blue')
ax.set_xlabel('Time')
ax.set_ylabel('CO2 (ppm)', color='blue')
ax.tick_params('y', colors='blue')

ax2 = ax.twinx()  Use the Axes twinx method to create a twin Axes that shares the x-axis.
ax2.plot(climate_change.index,
         climate_change["relative_temp"],
         color='red')
ax2.set_ylabel('Relative temperature (Celsius)',
               color='red')
ax2.tick_params('y', colors='red')

plt.show()
```

This method takes either y or x as its first argument, pointing to the fact that we are modifying the parameters of the y-axis ticks and tick labels.

# Coloring the ticks





# A function that plots time-series

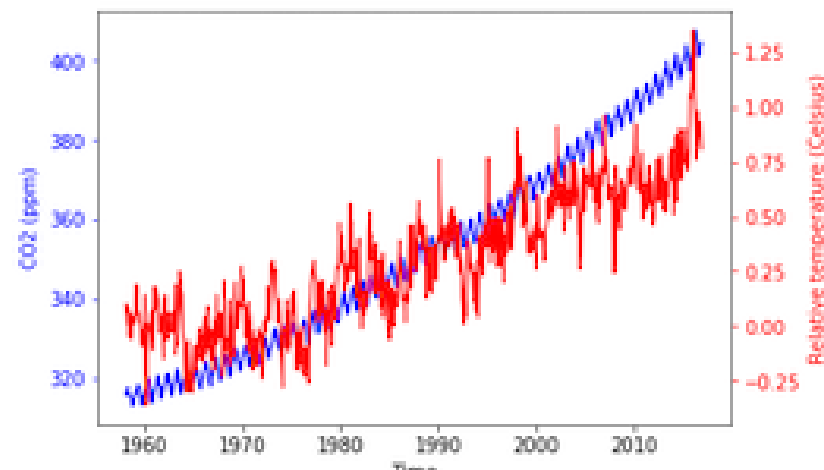
```
def plot_timeseries(axes, x, y, color, xlabel, ylabel):  
    axes.plot(x, y, color=color)  
    axes.set_xlabel(xlabel)  
    axes.set_ylabel(ylabel, color=color)  
    axes.tick_params('y', colors=color)
```

# Using our function

```
fig, ax = plt.subplots()
plot_timeseries(ax, climate_change.index, climate_change['co2'],
               'blue', 'Time', 'CO2 (ppm)')

ax2 = ax.twinx()
plot_timeseries(ax, climate_change.index,
               climate_change['relative_temp'],
               'red', 'Time', 'Relative temperature (Celsius)')

plt.show()
```



# Create your own function!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB

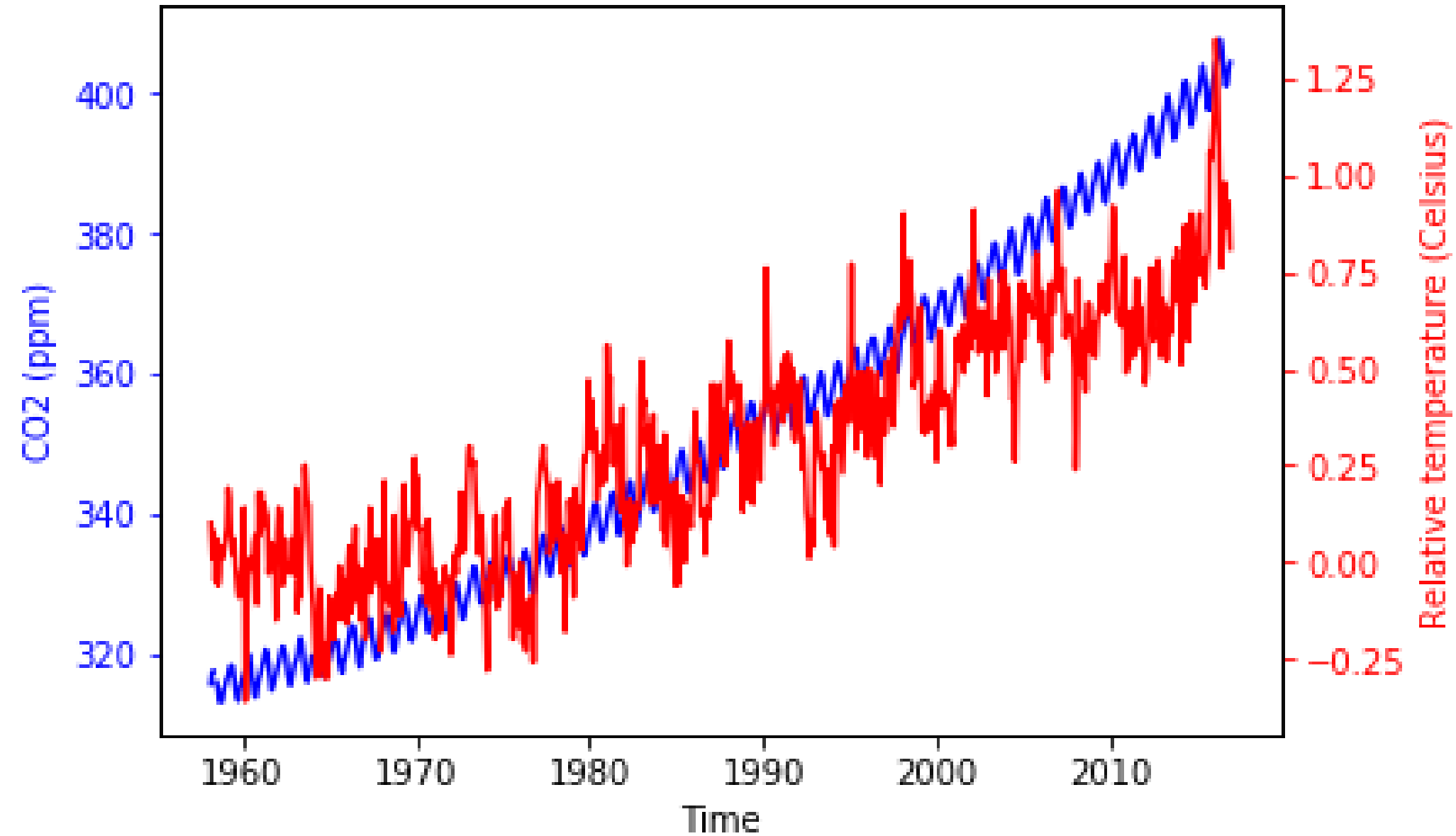
# Annotating time-series data

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB



**Ariel Rokem**  
Data Scientist

# Time-series data



# Annotation

Annotations are usually small pieces of text that refer to a particular part of the visualization, focusing our attention on some feature of the data and explaining this feature.

```
fig, ax = plt.subplots()
plot_timeseries(ax, climate_change.index, climate_change['co2'],
                'blue', 'Time', 'CO2 (ppm)')

ax2 = ax.twinx()
plot_timeseries(ax2, climate_change.index,
                climate_change['relative_temp'],
                'red', 'Time', 'Relative temperature (Celsius)')

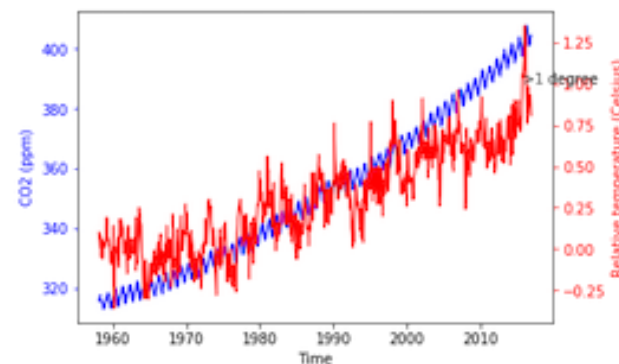
ax2.annotate(">1 degree", xy=[pd.Timestamp("2015-10-06"), 1])

plt.show()
```

take the annotation text as input: the string and the xy coordinate

The value to annotate has the x position of the TimeStamp of that data.

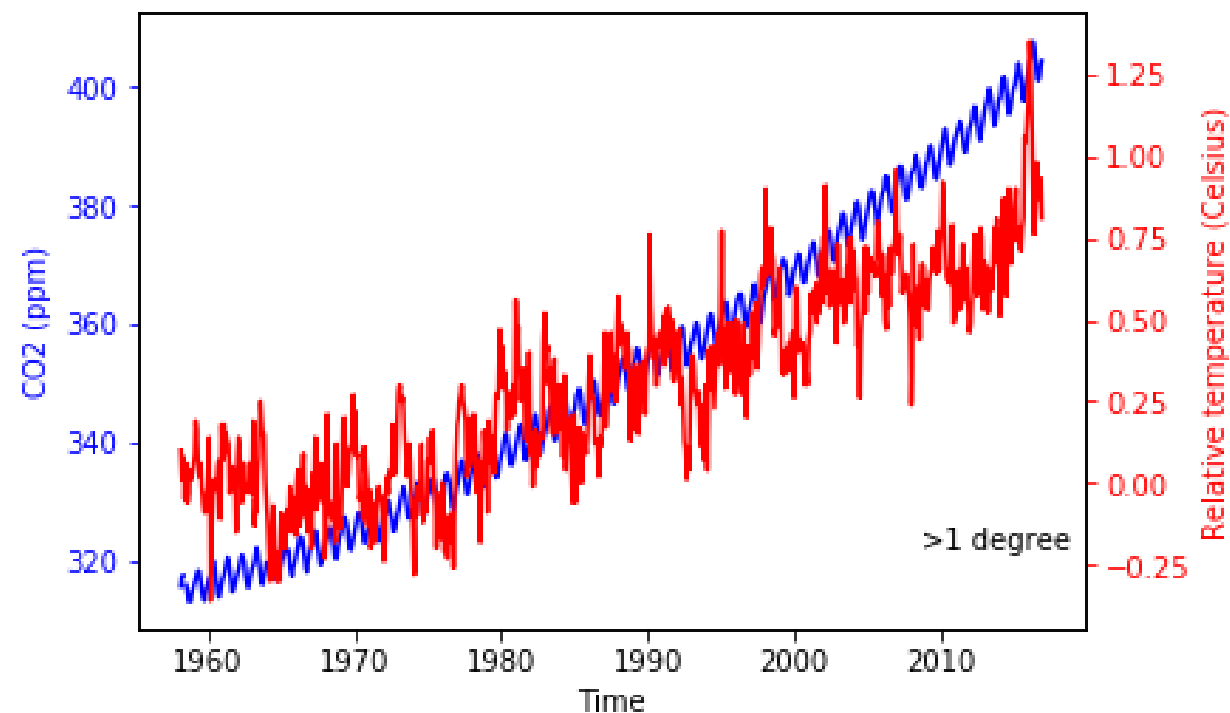
The y position of the data is 1, which is the 1 degree Celsius value at that date.



# Positioning the text

```
ax2.annotate(">1 degree",  
            xy=(pd.Timestamp('2015-10-06'), 1),  
            xytext=(pd.Timestamp('2008-10-06'), -0.2))
```

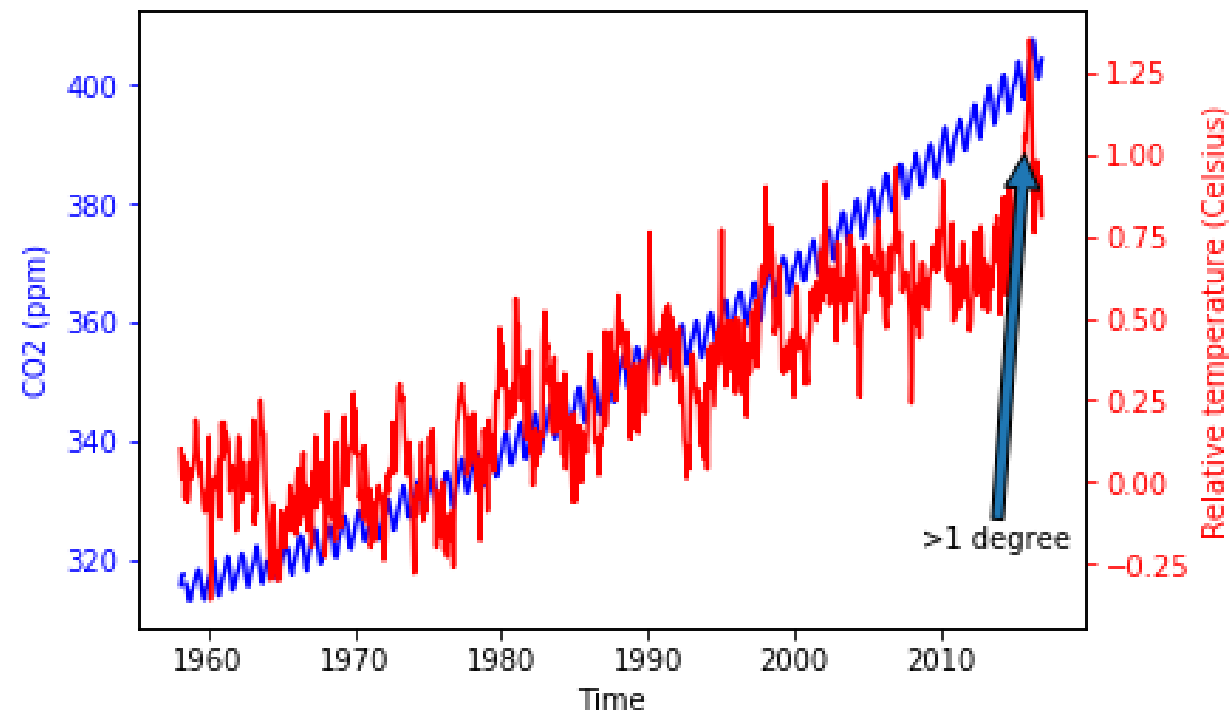
The `annotate` method takes an optional `xytext` argument that selects the `xy` position of the text.



# Adding arrows to annotation

```
ax2.annotate(">1 degree",  
            xy=(pd.Timestamp('2015-10-06'), 1),  
            xytext=(pd.Timestamp('2008-10-06'), -0.2),  
            arrowprops={})
```

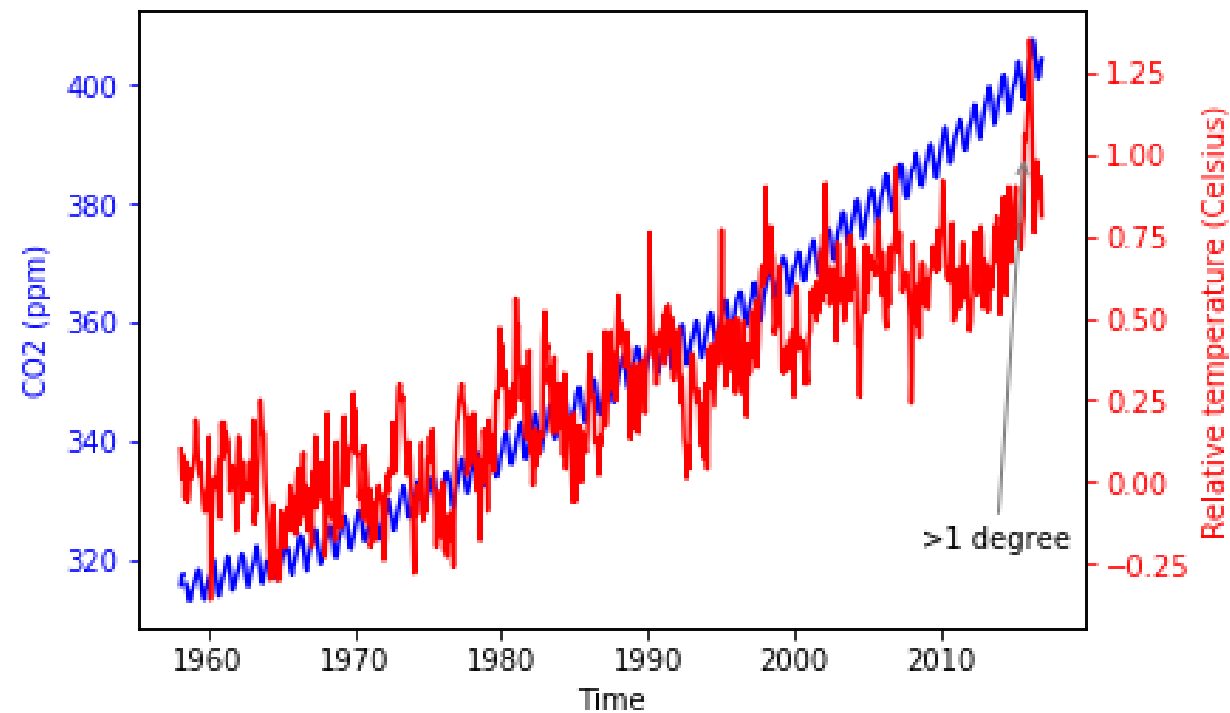
arrow properties  
add an arrow that connects the text to the data





# Customizing arrow properties

```
ax2.annotate(">1 degree",  
            xy=(pd.Timestamp('2015-10-06'), 1),  
            xytext=(pd.Timestamp('2008-10-06'), -0.2),  
            arrowprops={"arrowstyle": "->", "color": "gray"})
```



# Customizing annotations

<https://matplotlib.org/users/annotations.html>

# Practice annotating plots!

INTRODUCTION TO DATA VISUALIZATION WITH MATPLOTLIB