# Pandas tutorial 5: Scatter plot with pandas and matplotlib

**Scatter plots are frequently used in data science and machine learning projects.** In this pandas tutorial, I'll show you two simple methods to plot one. Both solutions will be equally useful and quick:

- one will be using **pandas** (more precisely: `pandas.plot.scatter()`)
- the other one using **matplotlib** (`matplotlib.pyplot.scatter()`)

Let's see them — and as usual: I'll guide you through step by step.

*Note: If you don't know anything about pandas (or Python), you might want to start here:*

1. *Python libraries and packages for Data Scientists*
2. *Learn Python from Scratch*
3. *Pandas Tutorial 1 (Basics)*
4. *Pandas Tutorial 2 (Aggregation and grouping)*
5. *Pandas Tutorial 3 (Data Formatting)*
6. *Pandas Tutorial 4 (Plotting in pandas: Bar Chart, Line Chart, Histogram)*

# Download the code base!

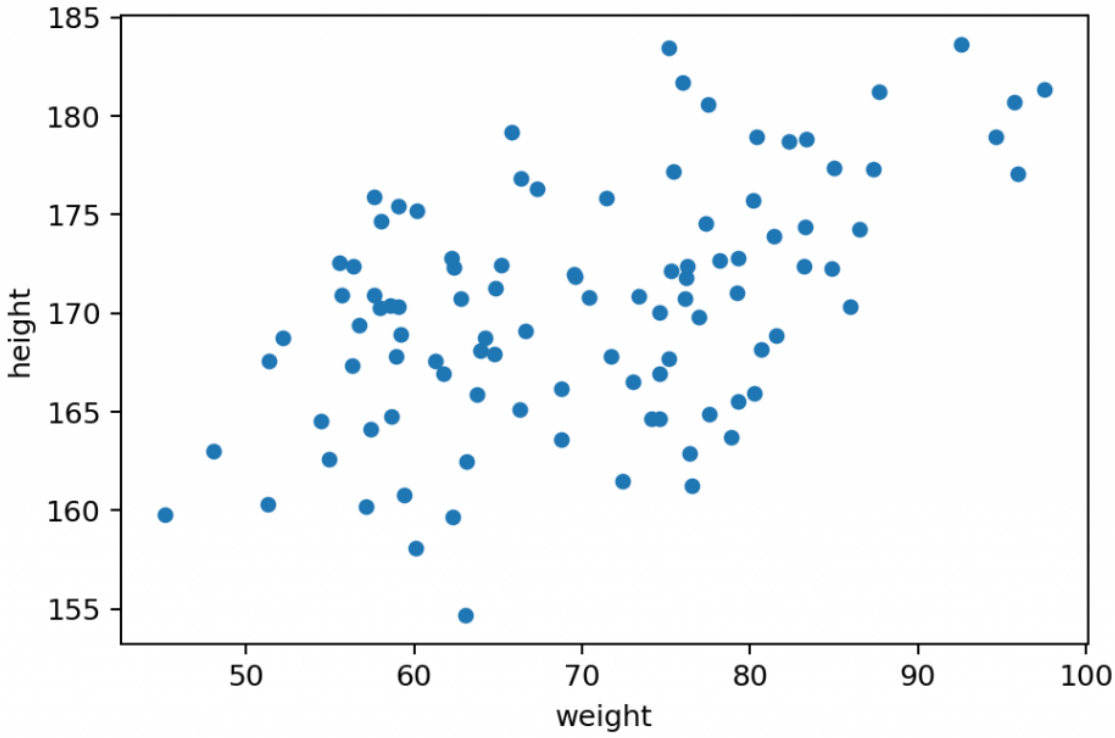
This is a hands-on tutorial, so it’s best if you do the coding part with me

# What is a scatter plot? And what is it good for?

**Scatter plots are used to visualize the relationship between two (or sometimes three) variables in a data set.** The idea is simple:

- you take a data point,
- you take two of its variables,
- the y-axis shows the value of the first variable,
- the x-axis shows the value of the second variable

Following this concept, you display each and every datapoint in your dataset. You’ll get something like this:

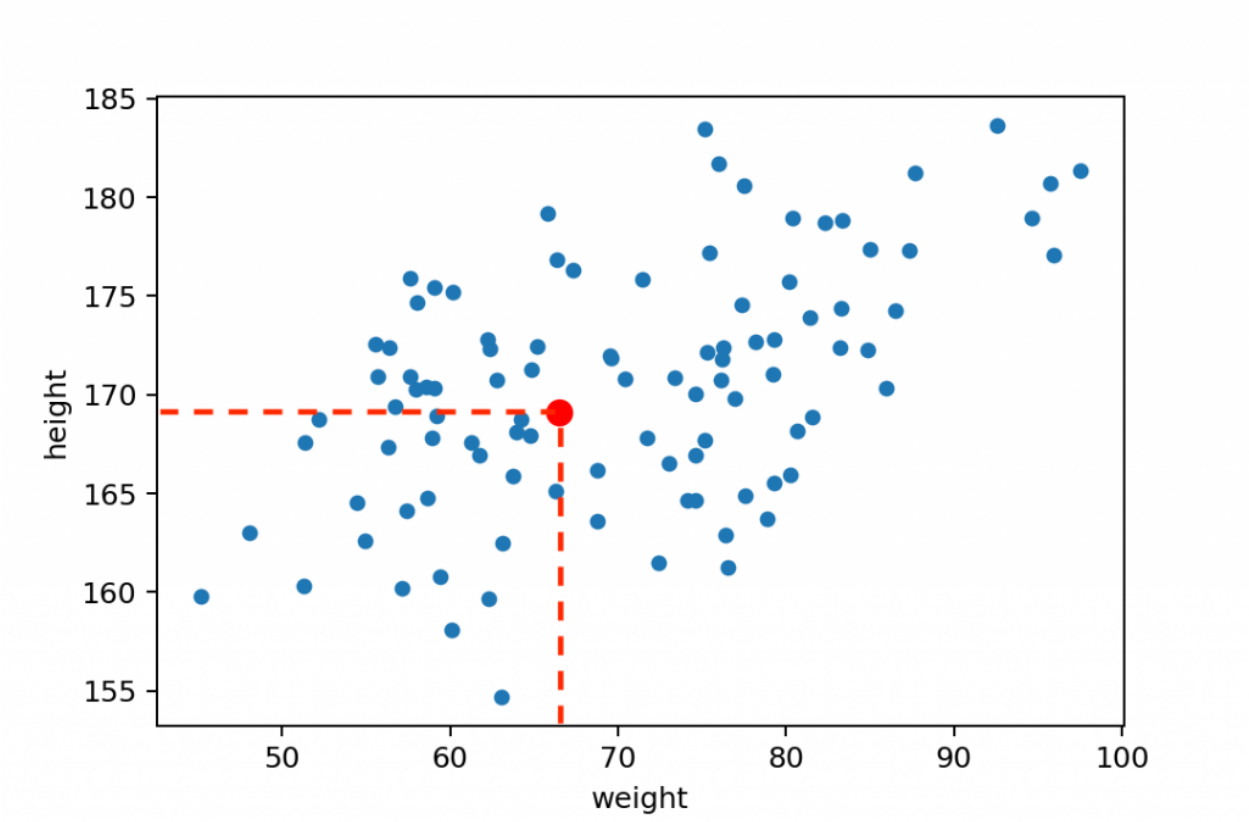



Boom! **This is a scatter plot.** At least, the easiest (and most common) example of it.

This particular scatter plot shows the relationship between the height and weight of people from a random sample. Again:

- y-axis shows the height
- x-axis shows the weight
- and each blue dot represents a person in this dataset

So, for instance, this person's *(highlighted with red)* weight and height is 66.5 kg and 169 cm.

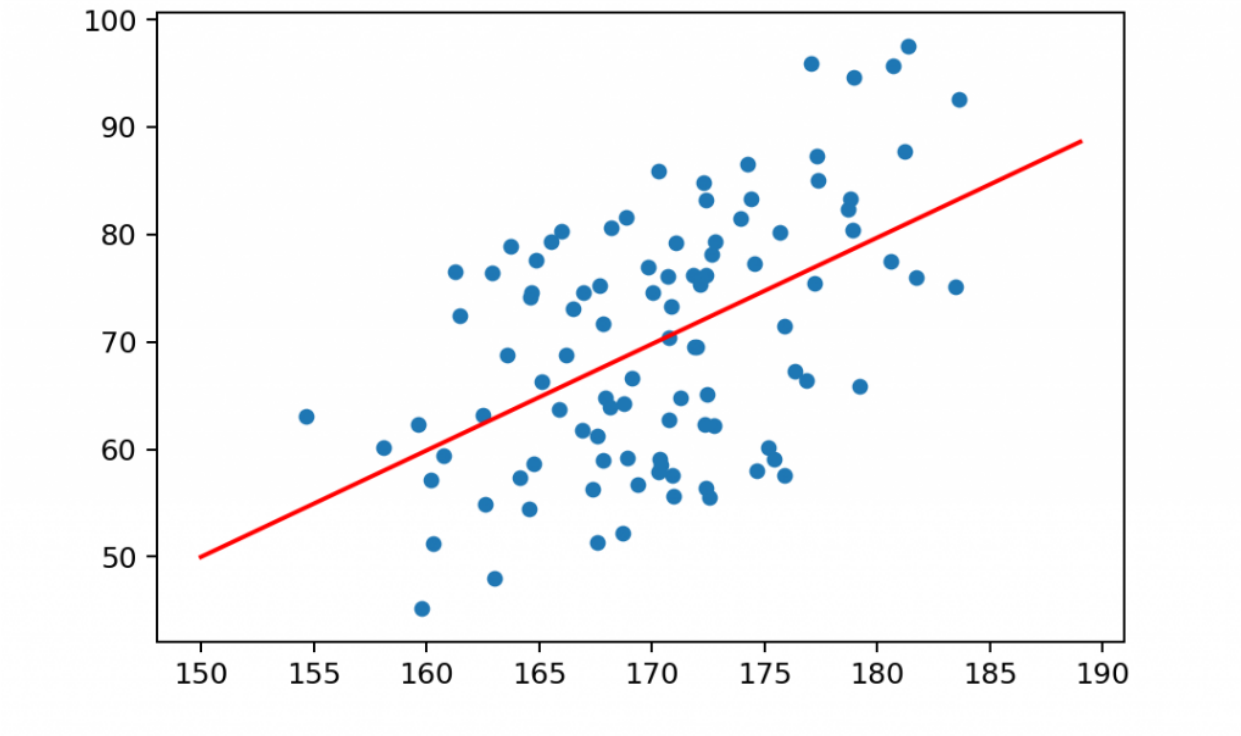



# How to read a scatter plot?

Scatter plots play an important role in data science – **especially in building/prototyping machine learning models**. Looking at the chart above, you can immediately tell that there's a strong correlation between weight and height, right? As we discussed in my linear regression work, you can even fit a

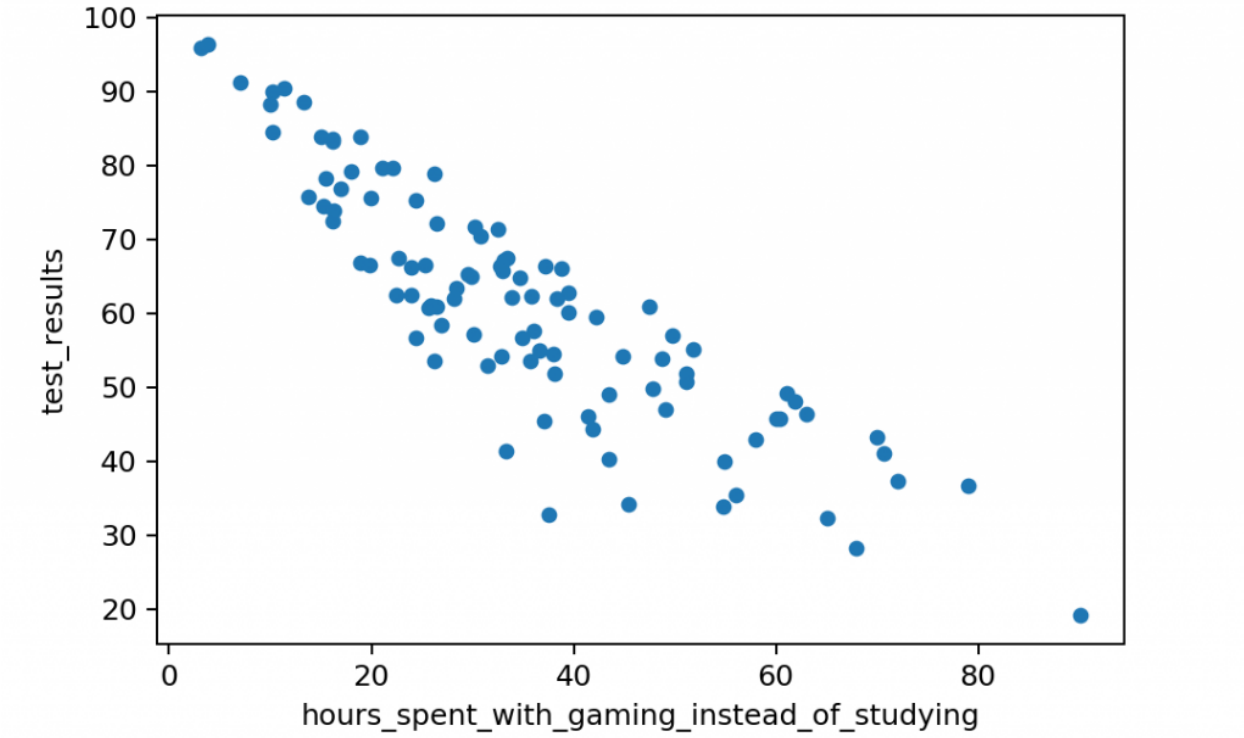
trend line (a.k.a. regression line) to this data set and try to describe this relationship with a mathematical formula.

Something like this:

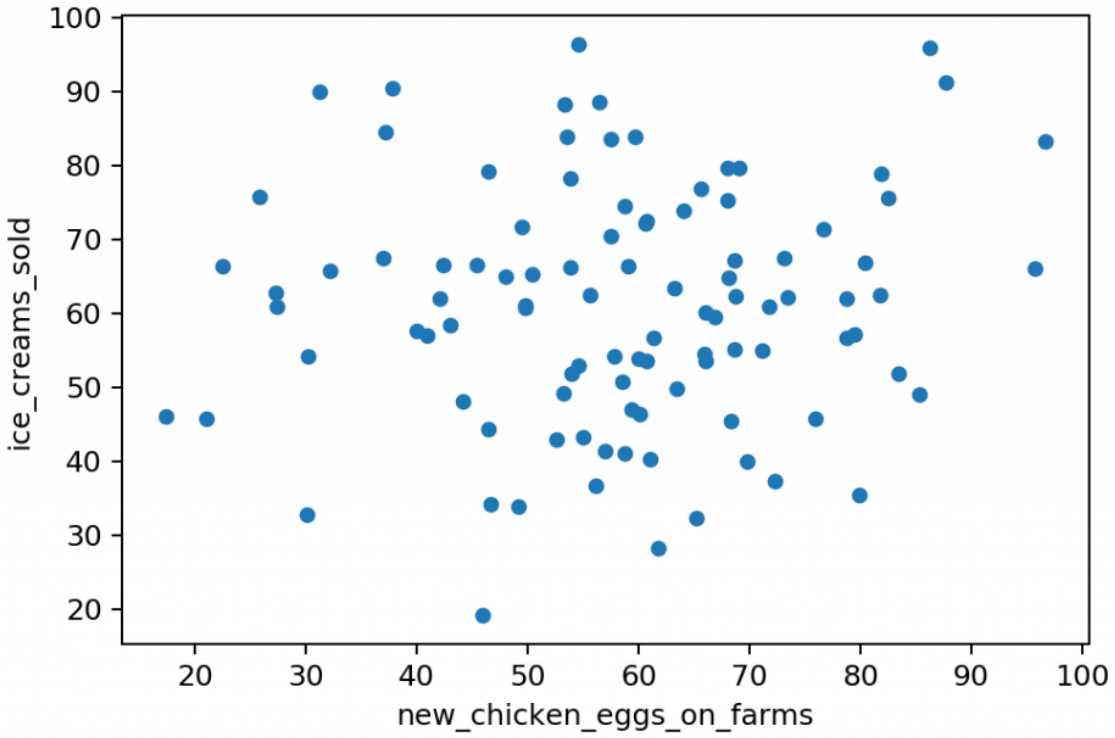



This above is called a **positive correlation**. The greater is the height value, the greater is the expected weight value, too. *(Of course, this is a generalization of the data set. There are always exceptions and outliers!)*

But it's also possible that you'll get a **negative correlation**:




And in real-life data science projects, you'll see **no correlation** often, too:




Anyway: if you see a sign of positive or negative correlation between two variables in a data science project, that's a good indicator that you found something interesting — something that's worth digging deeper into. Well, in 99% of cases it will turn out to be either a triviality, or a coincidence. But in the remaining 1%, *you might find gold!*

Okay, I hope I set your expectations about scatter plots high enough.

**It's time to see how to create one in Python!**

# Scatter plot in pandas and matplotlib

As I mentioned before, I'll show you two ways to create your scatter plot.

You'll see here the Python code for:

- a **pandas scatter plot**
  and
- a **matplotlib scatter plot**

The two solutions are fairly similar, the whole process is ~90% the same… The only difference is in the last few lines of code.

*Note: By the way, I prefer the matplotlib solution because I find it a bit more transparent*.

I'll guide you through these 4 steps:

1. **Importing pandas, numpy and matplotlib**
2. **Getting the data**
3. **Preparing the data**
4. **Plotting a scatter plot**

# Step #1: Import pandas, numpy and matplotlib!

Just as we have done in the histogram work, as a first step, you'll have to import the libraries you'll use. And you'll also have to make a small tweak in your Jupyter environment.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
```

The first two lines will import `pandas` and `numpy`.

The third line will import the `pyplot` from `matplotlib` — also, we will refer to it as `plt`.

And `%matplotlib inline` sets your environment so you can directly plot charts into your Jupyter Notebook!

Great!

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        %matplotlib inline
```

# Step #2: Get the data!

Well, in real data science projects, getting the data would be a bit harder. You should read .csv files or SQL tables into your Python environment. But this tutorial's focus is not on learning that — so you can take the lazy way and use the dataset I'll provide for you here.

*Note: What's in the data? This is the modified version of the dataset that we used in the pandas histogram work — the heights and weights of our hypothetical gym's members.*

```
np.random.seed(0)
mu = 170 #mean
sigma = 6 #stddev
sample = 100
height = np.random.normal(mu, sigma, sample)

weight = (height-100) * np.random.uniform(0.75, 1.25, 100)
```

This is a random generator, by the way, that generates 100 height and 100 weight values — in numpy array format. By using the `np.random.seed(0)` line, we also made sure you'll be able to work with the exact same data points that I do in this article.

```
In [4]: height
Out[4]: array([180.58431408, 172.40094325, 175.8724279 , 183.4453592 ,
               181.20534794, 164.13633272, 175.70053051, 169.09185675,
               169.38068689, 172.46359101, 170.86426143, 178.72564104,
               174.56622635, 170.7300501 , 172.6631794 , 172.00204596,
               178.96447444, 168.76905042, 171.87840621, 164.87542556,
               154.6820611 , 173.92171157, 175.18661719, 165.54700988,
               183.61852774, 161.27380595, 170.2745511 , 168.8768969 ,
               179.19667529, 178.81615262, 170.92968455, 172.26897512,
```

```
In [5]: weight
Out[5]: array([77.51626973, 76.25242778, 57.63243821, 75.16652885, 87.70882186,
               57.40450651, 80.1674458 , 66.6311054 , 56.73515698, 65.1549975 ,
               73.34326729, 82.30264923, 77.33730302, 76.14800093, 78.18933299,
               69.53303948, 94.6210212 , 64.2152282 , 69.57344273, 77.58852279,
               63.05372031, 81.45760811, 60.1578232 , 79.29492539, 92.57579877,
               76.55693329, 57.95711959, 81.55458715, 65.83195656, 83.37013275,
               57.58851958, 84.84407415, 74.61094889, 60.12312355, 64.76081868,
```

*Note: For now, you don't have to know line by line what's going on here. (I'll write a separate article about how* `numpy.random` *works.)*

In the next step, we will push these data sets into pandas dataframes.

# Step #3: Prepare the data!

Again, preparing, cleaning and formatting the data is a painful and time consuming process in real-life data science projects. But in this tutorial, we are lucky, everything is prepared – the data is clean – so you can push your `height` and `weight` data sets directly into a pandas dataframe (called `gym`) by running this one line of code:

```
gym = pd.DataFrame({'height': height, 'weight': weight})
```

*Note: If you want to experience the complexity of a true-to-life data science project, go and check out my 6-week course: The Junior Data Scientist's First Month!*

Your `gym` dataframe should look like this.

```
In [9]: gym
```

Out[9]:

| | height | weight |
|---|---|---|
| **0** | 180.584314 | 77.516270 |
| **1** | 172.400943 | 76.252428 |
| **2** | 175.872428 | 57.632438 |
| **3** | 183.445359 | 75.166529 |
| **4** | 181.205348 | 87.708822 |
| **...** | ... | ... |
| **95** | 174.239439 | 86.527865 |
| **96** | 170.063000 | 74.581428 |
| **97** | 180.715223 | 95.754513 |
| **98** | 170.761473 | 62.749223 |
| **99** | 172.411936 | 83.203010 |

100 rows × 2 columns

Perfect: ready for putting it on a scatter plot!

# Step #4a: Pandas scatter plot

Okay, all set, we have the `gym` dataframe. Let's create a **pandas scatter plot**!

Now, this is only one line of code and it's pretty similar to what we had for bar charts, line charts and histograms in pandas…

It starts with: `gym.plot` …and then you simply have to define the chart type that you want to plot, which is `scatter()`. But when plotting a scatter plot in pandas, you'll always have to specify the `x` and `y` values as parameters, too. *(This could seem unusual because for bar and line charts, you didn't have to do anything similar to this.)*

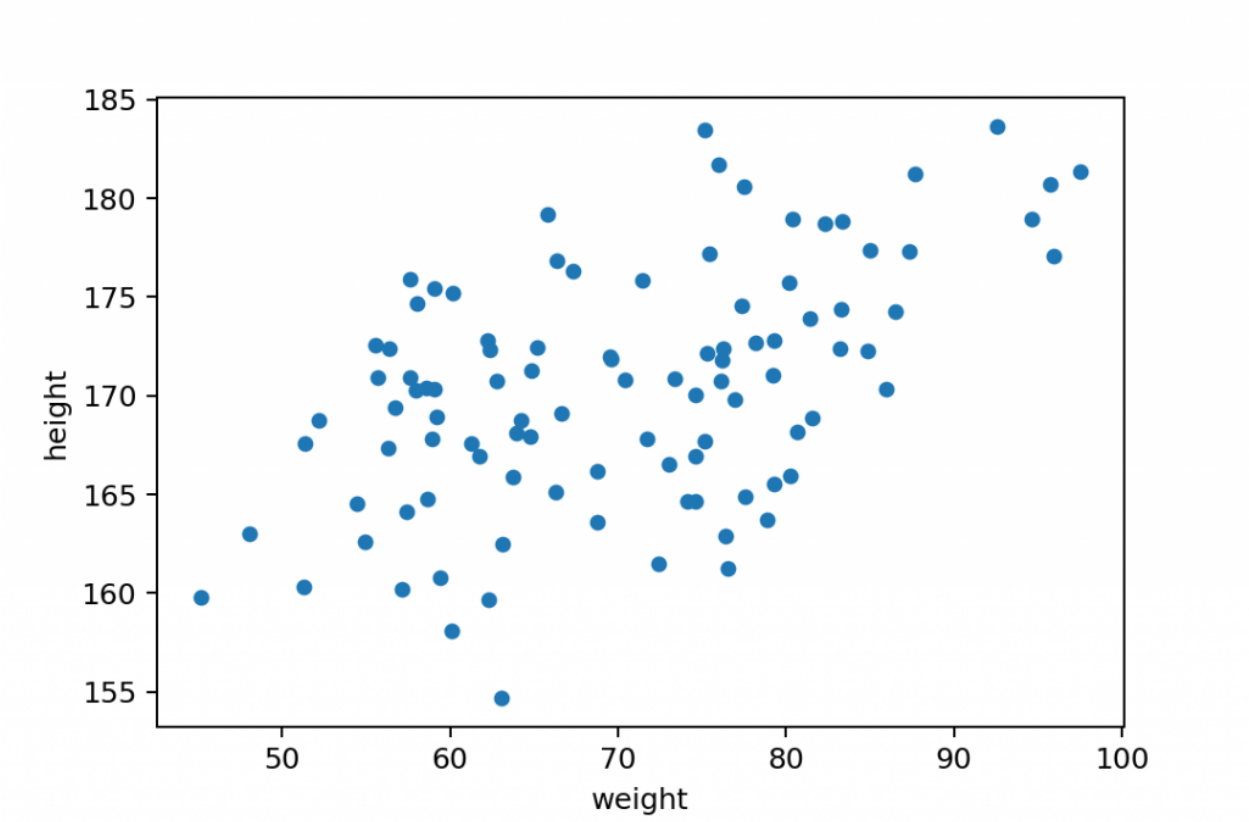
So the final line of code will be:

```
gym.plot.scatter(x = 'weight', y = 'height')
```

The `x` and `y` values – by definition – have to come from the `gym` dataframe, so you have to refer to the column names: `'weight'` and `'height'`!

A quick comment: Watch out for all the apostrophes! I know from my live workshops that the syntax might seem tricky at first. But you'll get used to it after your 5th or 6th scatter plot, I promise!

That's it! You have plotted a scatter plot in pandas!




# Step #4b: Matplotlib scatter plot

Here's an alternative solution for the last step. In this one, we will use the `matplotlib` library instead of `pandas`. *(Although, I have to mention here that the pandas solution I showed you is actually built on matplotlib's code.)*

In my opinion, this solution is a bit more elegant. But from a technical standpoint — and for results — both solutions are equally great.

Anyway, type and run these three lines:

```
x = gym.weight
y = gym.height

plt.scatter(x,y)
```

- `x = gym.weight`

  This line defines what values will be displayed on the x-axis of the scatter plot. It's the `weight` column again from the `gym` dataset. *(Note: This is in pandas Series format… But in this specific case, I could have added the original numpy array, too.)*
- `y = gym.height`

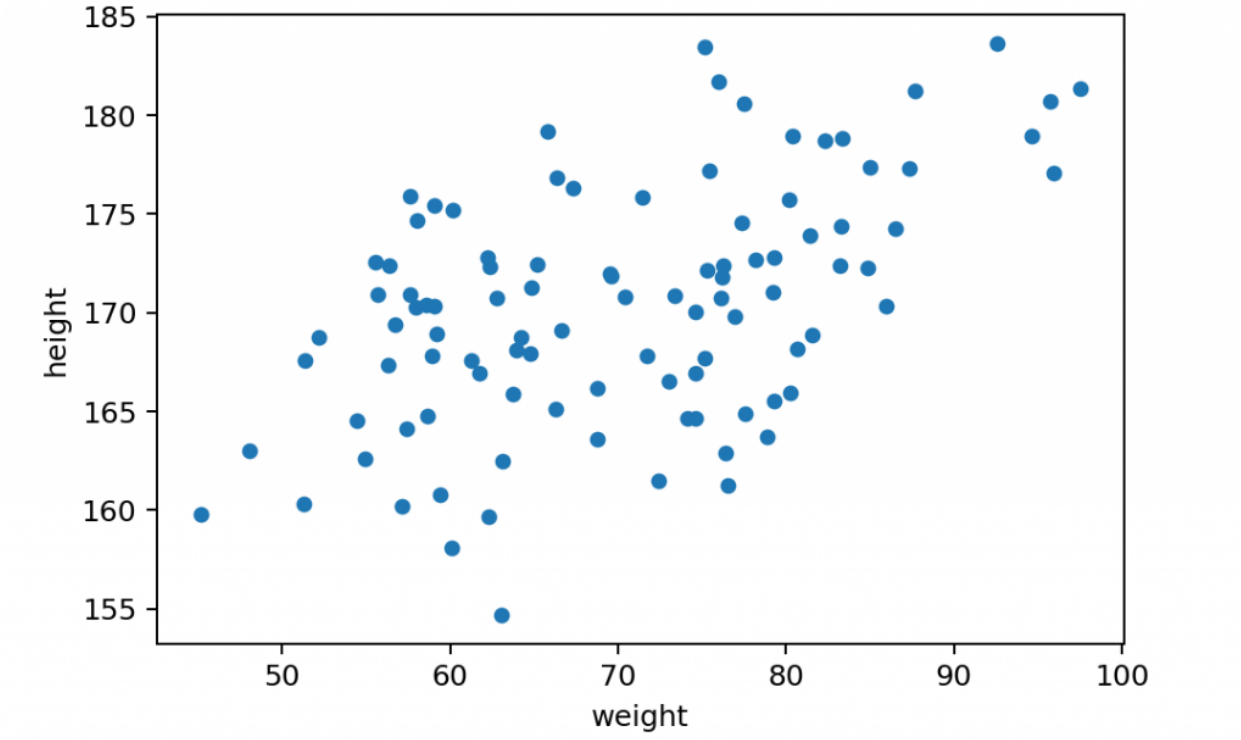
  On the y-axis we want to display the `gym.height` values. *(This is in pandas Series format, too!)*
- `plt.scatter(x,y)`

  And then this line does the plotting. Remember, you set `plt` at the very beginning of our Jupyter notebook (`import matplotlib.pyplot as plt`) — and so `plt` refers to `matplotlib.pyplot`! And the `x` and `y` values are parameters that have been defined in the previous two lines.

```
In [8]: x = gym.weight
        y = gym.height
        plt.scatter(x,y)
```

Again: this is slightly different (and in my opinion slightly nicer) syntax than with pandas.

But the result is exactly the same.



# Conclusion

This is how you make a scatter plot in pandas and/or in matplotlib. I think it's fairly easy and I hope you think the same. If you haven't done so yet, check out my Python histogram tutorial, too! If you have any questions, leave a comment below!