1. Automating figures from data

One of the strengths of Matplotlib is that, when programmed correctly, it can flexibly adapt to the inputs that are provided.

2. Why automate?

This means that you can write functions and programs that automatically adjust what they are doing based on the input data. Why would you want to automate figure creation based on the data? Automation makes it easier to do more. It also allows you to be faster. This is one of the major benefits of using a programming language like Python and software libraries such as Matplotlib, over tools that require you to interact with a graphical user interface every time you want to create a new figure. Inspecting the incoming data and changing the behavior of the program based on the data provides flexibility, as well as robustness. Finally, an automatic program that adjusts to the data provides reproducible behavior across different runs.

3. How many different kinds of data?

Let's see what that means for Matplotlib. Consider the data about Olympic medal winners that we've looked at before. Until now, we always looked at two different branches of sports and compared them to each other, but what if we get a new data file, and we don't know how many different sports branches are included in the data? For example, what if we had a data-frame with hundreds of rows and a "Sport" column that indicates which branch of sport each row belongs to.

4. Getting unique values of a column

A column in a pandas DataFrame is a pandas Series object, so we can get the list of different sports present in the data by calling the unique method of that column. This tells us that there are 10 different branches of sport here.

5. Bar-chart of heights for all sports

Let's say that we would like to visualize the height of athletes in each one of the sports, with a standard deviation error bar. Given that we don't know in advance how many sports there are in the DataFrame, once we've extracted the unique values, we can loop over them. In each iteration through, we set a loop variable called sport to be equal to one of these unique values. We then create a smaller DataFrame, that we call sport-underscore-d-f, by selecting the rows in which the "Sport" column is equal to the sport selected in this iteration. We can call the bar method of the Axes we created for this plot. As before, it is called with the string that holds the name of the sport as the first argument, the mean method of the "Height" column is set to be the height of the bar and an error bar is set to be equal to the standard deviation of the values in the column. After iterating over all of the sports, we exit the loop. We can then set the y-label to indicate the meaning of the height of each bar and we can set the x-axis tick labels to be equal to the names of the sports. As we did with the country names in the stacked bar chart that you saw in a previous lesson, we rotate these labels 90 degrees, so that they don't run over each other.

6. Figure derived automatically from the data

This is what this figure would look like. Importantly, at no point during the creation of this figure did we need to know how many different sports are recorded in the DataFrame. Our code would automatically add bars or reduce the number of bars, depending on the input data.