**Adding some style**

Previously, we learned how to create line plots and add labels, legends, and text. Now, we're going to make those plots beautiful.

**And miles to go**

After his success in graphing productivity, Officer Deshaun wants to plot the miles driven by each office over the course of a day. He's including more officers in this plot, so it's becoming more difficult to read. Officer Deshaun will improve is plot by customizing the the style of each line.

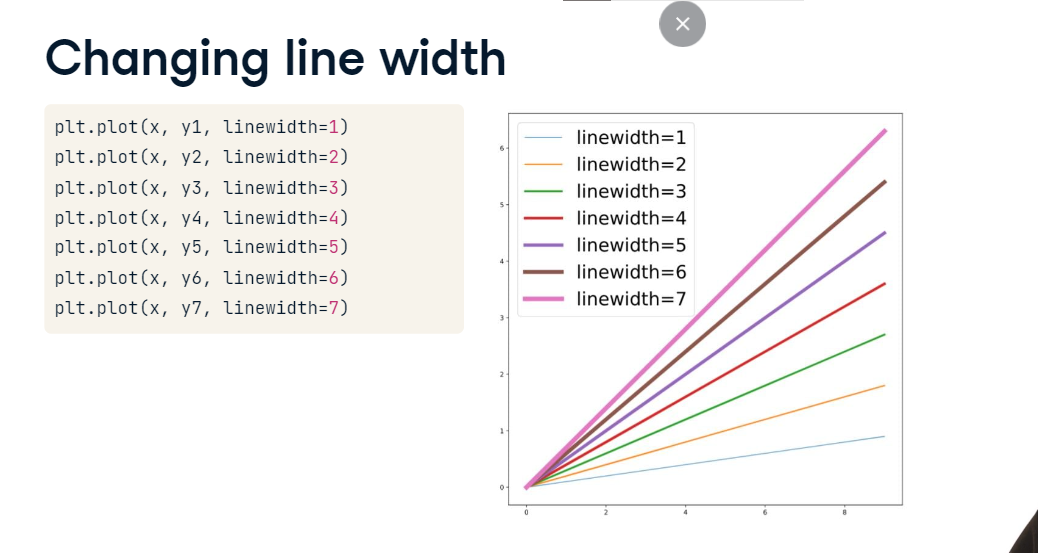
**Changing line color**

The first change Officer Deshaun might want to make to a line is to change its color. He can do this by adding the keyword argument "color" to the plt-dot-plot command. In this case, we plot six different lines in six different colors. The color keyword will accept a string corresponding to a "web color". For a list of allowed color names, visit Wikipedia and look up "web colors". In this example, we plot six different lines, each with a different color. You can't see the code, but we've added a legend so that you can see what each color appears on our plot.



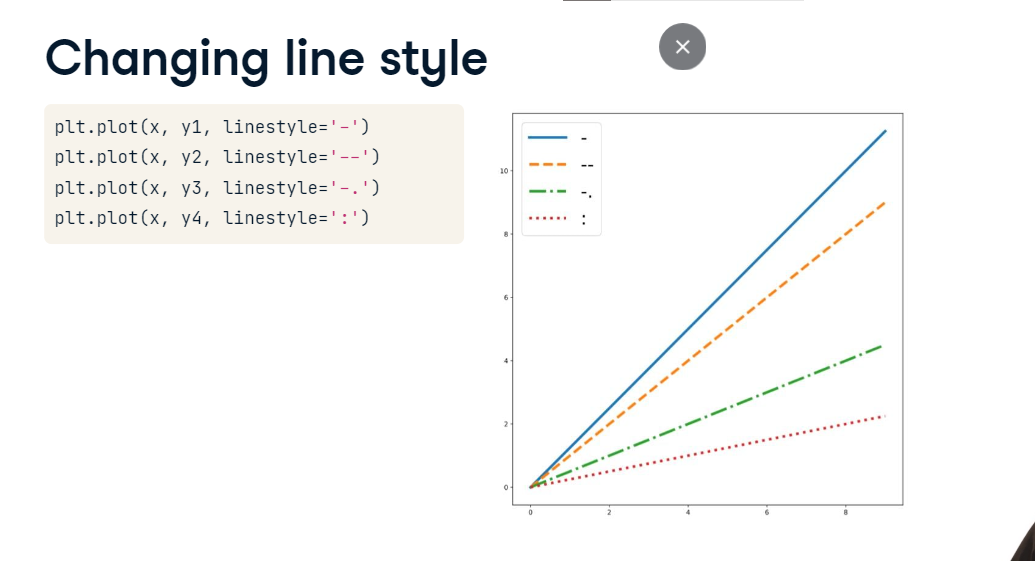
**Changing line width**

Officer Deshaun can also modify the width of a line. The default line width is 1, but you can increase it using the keyword argument linewidth. In this example, Officer Deshaun plotted seven dataset, each with a slightly thicker line than the previous one. Again, we've hidden the extra code, but are showing a legend so that you can see what each linewidth looks like. Increasing line width can make graphs easier to use or can emphasize one line in a multi-line plot.



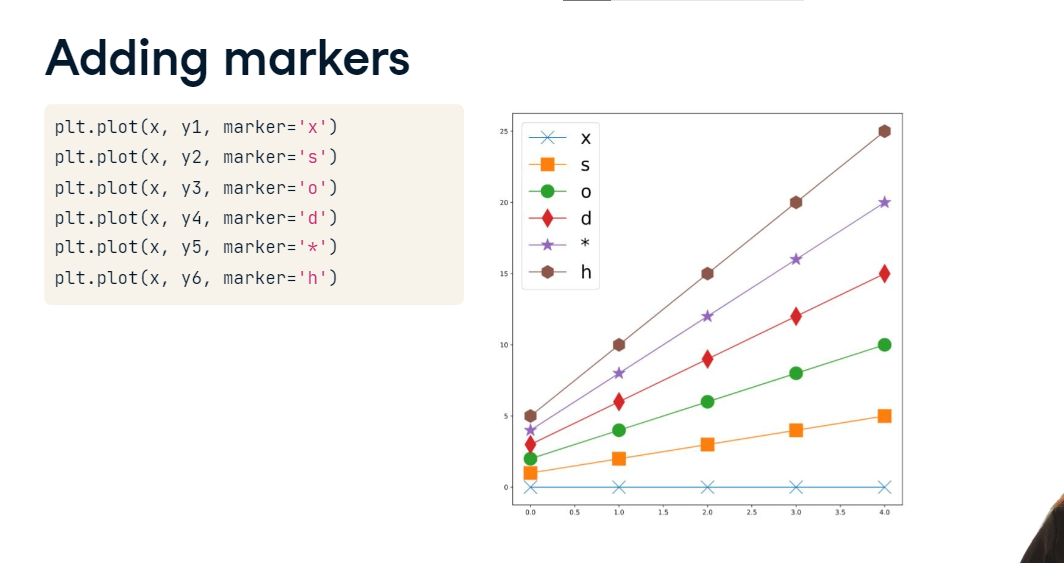
**Changing line style**

Officer Deshaun can further modify a line by changing its style. He can create different types of dashed lines using the keyword argument linestyle. Linestyle accepts several strings which correspond to different types of dashing. In this example, we plot four datasets with four different values of linestyle: a single hyphen for a normal line, two hyphens for a dashed line, a hyphen followed by a dot for a dot/dash line, and colon for a dotted line.



**Adding markers**

Adding markers is a great way to distinguish between different lines or to emphasize the location of data points. Officer Deshaun can add a marker using the keyword marker, which accepts several different strings. For example, if he sets marker equal to the letter s, the plot will have square markers. If he sets marker equal to the letter d, the plot will have diamond-shaped markers. In this example, we plot six different datasets, each with a different marker. Again, we hid the legend code but you can use it to see which symbols make each type of marker.



**Setting a style**

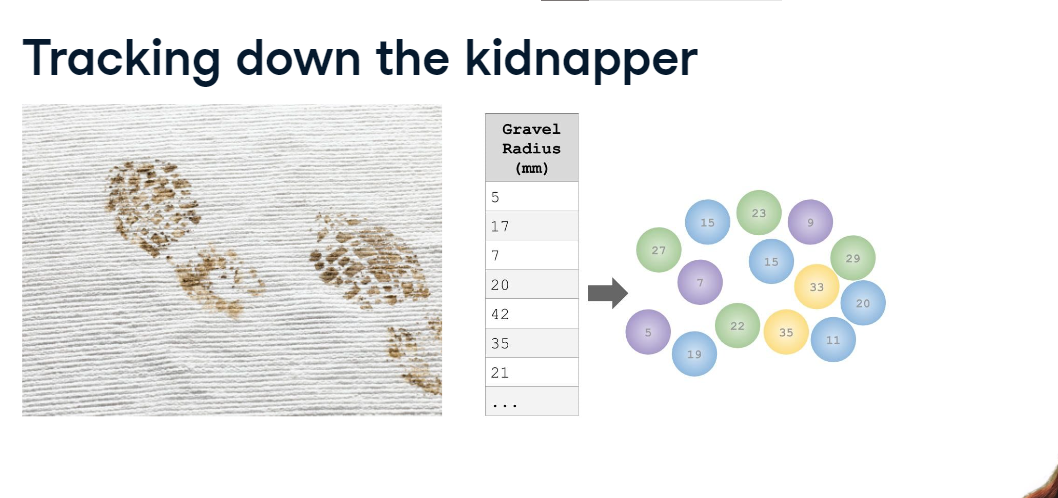
Sometimes, we want to make more drastic change to our graph. We can change the background, colors, and fonts for our entire graph by setting a style. The function plt-dot-style-dot-use accepts several different strings which correspond to different plotting styles. Here we show the same graph plotted in four different styles: fivethirtyeight (from the famous news site), ggplot (from another plotting library), seaborn (from yet another plotting library), and default (which is the normal style in matplotlib). Which do you think is best?



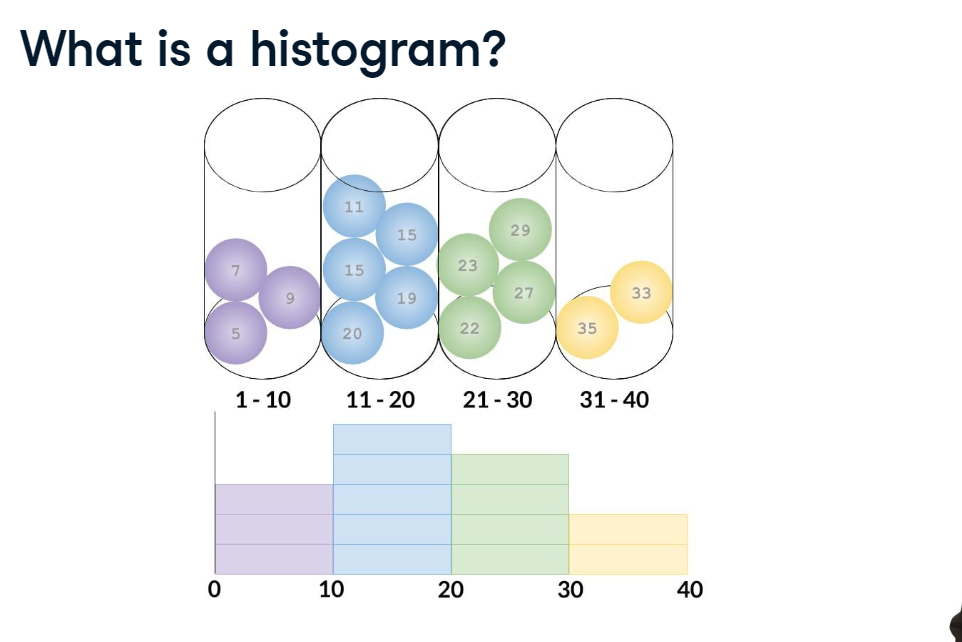
**Making a histogram**

**Tracking down the kidnapper**

Freddie Frequentist left a dirty shoe print at the scene of the kidnapping. In that shoe print, there were many tiny pieces of gravel. Our crime lab was able to give us a Pandas DataFrame with the radius of each piece of gravel. We want to compare the distribution of gravel radii to samples from the three sites where Freddie could be hiding Bayes. If the distributions match, then the gravel in the footprint came from that site, and Freddie is likely hiding there.

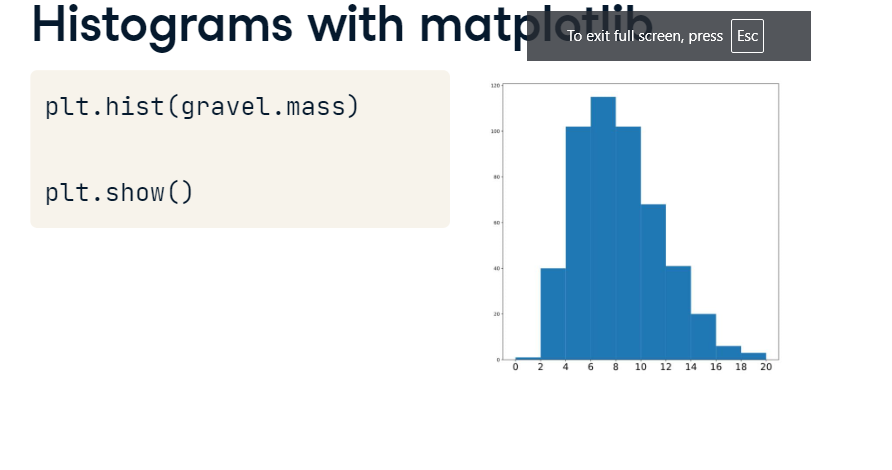
**What is a histogram?**

The best way of visualizing the distributions of gravel radii is by creating a histogram. A histogram visualizes the distribution of values in a dataset. To create a histogram, we p

lace each piece of data into a bin. In this example, we have divided our data into four bins: 1 - 10 mm, 11 - 20 mm, 21 - 30 mm, and 31 - 40 mm. The histogram tells us how many pieces of data (or pieces of gravel) fall into each of those bins. When we look at a histogram, we can quickly understand the entire dataset. Plotting a histogram in matplotlib is easy.

**Histograms with matplotlib**

We simply use the command plt-dot-hist. This function takes just one positional argument: our dataset. The output is shown on the right. By default, matplotlib will create a histogram with 10 bins of equal size spanning from the smallest sample to the largest sample in our dataset. If we want to change the number of bins,

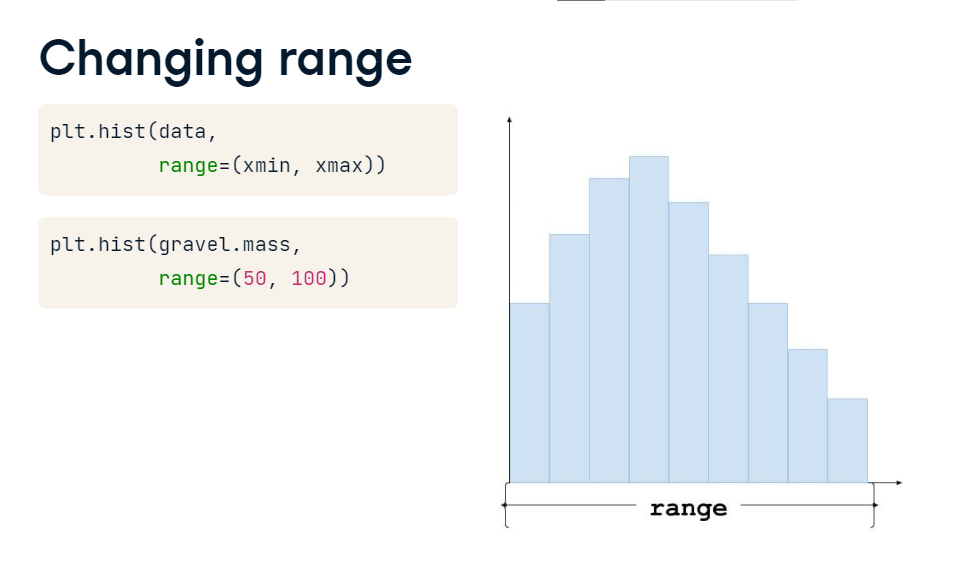


**Changing bins**

we can use the keyword argument "bins". Bins accepts one integer. In this case, we divide our histogram into 40 bins. This allows us to see more detail in our histogram. If we want to zoom in on a specific piece of our dataset,

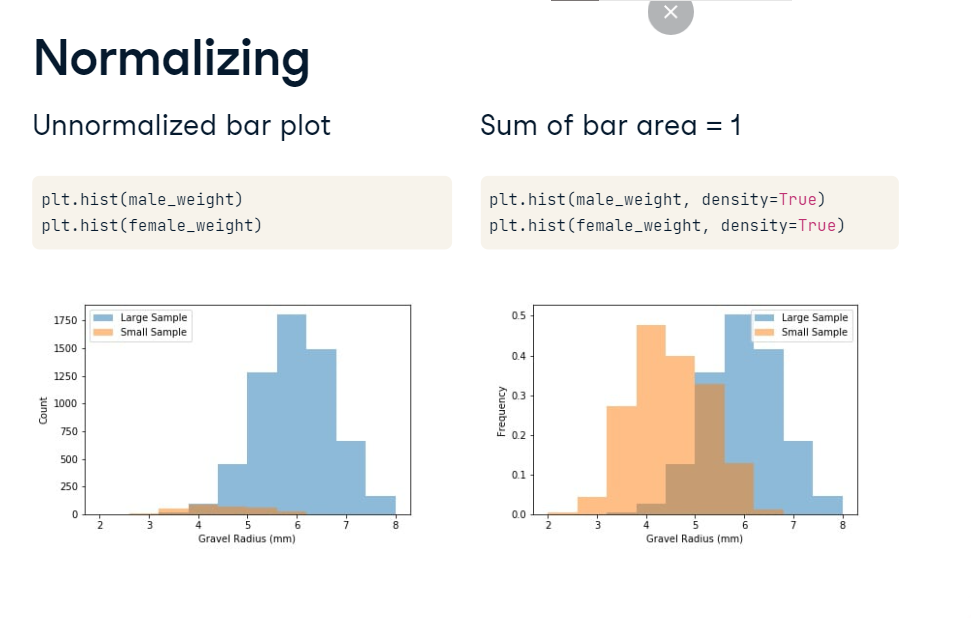
**Changing range**

we can use the keyword "range" to set the minimum and maximum value for our histogram. Note that we give the minimum and maximum values inside of a second set of parenthesis, separated by a comma. In this case, the minimum value is 50 and the maximum value is 100. Suppose we wanted to compare the distributions of weights



**7. Normalizing**

of male and female puppies. For some reason, we were able to collect many more samples of male puppy weights than female puppy weights. When we plot both histograms on the same axes, we can't actually see the difference in the distributions. In this case, we don't actually care about the absolute number of male puppies with a given weight. Instead, we care about what proportion of the dataset has that weight. We can solve this problem with normalization. Normalization reduces the height of each bar by a constant factor so that the sum of the areas of each bar adds to one. This would make our two histograms comparable, even if the sample sizes are different. We can normalize our histogram by using the keyword argument density equals True. Now each bar represents a proportion of the entire dataset. If a bar from the male puppies has the same height as a bar from the female puppies, both bars represent the same proportion of each population. Now that you've learned how



**Recap of the rescue**

Congratulations! You found that Freddy Frequentist kidnapped Bayes the Golden Retriever and was hiding him at the Shady Groves Campsite.

**You did it!**

Bayes was safely returned to his DataCamp family. Now that you've solved the mystery of Bayes' kidnapping, let's recap what you've learned.

**Modules and variables**

First, you learned about modules. Modules help group together related tools in Python. You can use a module by importing it at the top of a script file, before writing any other code. You also learned about variables as you filled out Bayes' missing puppy report. Variables store data. Two common types of variables are strings (which represent text like Bayes' fur color) and floats (which represent numbers like Bayes' age).

**Using functions**

Next, you learned about functions. Functions are like machines that turn inputs into outputs. They can take positional arguments, which must come in a specific order, or keyword arguments, which come at the end and can go in any order. You used a license plate lookup function to get your first list of suspects.

**Working with tabular data**

You narrowed your list of suspects by using Pandas to examine credit card records. Pandas is a module that uses tools like DataFrames to examine and modify tabular data. You investigated the credit reports using functions like head and info, and selected specific rows that corresponded to suspicious purchases using logic.

**Creating line plots**

After narrowing the suspect pool down to two people, you collected handwriting samples and analyzed them using line plots. Matplotlib is a module for creating charts and visualizations. You used the plt-dot-plot function to create line plots and used keyword arguments to modify the color and linestyle of your plot. You used functions like plt-dot-legend and plt-dot-title to add annotations to your line plots. Line plots of the handwriting samples helped you learn that Freddy Frequentist was the kidnapper.

**More plot types**

You learned to create several other plot types using matplotlib. You used a scatter plot to investigate cell phone data and narrow Freddy's location to three possible campsites. You used bar charts to track how Officer Deshaun and his friends used their time to solve the mystery. Finally, you used a histogram to investigate the distribution of gravel radii on Freddy's shoeprint. This helped you determine that Bayes was being kept at the Shady Groves Campsite.