**#GETTING STARTED IN PYTHON**

-Importting pandas and matplotlib

**\*import pandas as pd**

**\*df = pd.read\_csv(‘ransom.csv’)**

**\*from matplotlib import pyplot as plt**

**\*plt.plot(df.letters, df.frequency)**

**\*plt.show()**

-Importing python module exercise :

# Import statsmodels under the alias sm

import statsmodels as sm

# Use an import statement to import seaborn with alias sns

import seaborn as sns

-creating a float exercise :

# Fill in Bayes' age (4.0)

bayes\_age = 4.0

# Display the variable bayes\_age

print(bayes\_age)

-creating strings :

# Bayes' favorite toy

favorite\_toy = "Mr. Squeaky"

# Bayes' owner

owner =  'DataCamp'

# Display variables

print(favorite\_toy)

print(owner)

-Correcting string error :

# One or more of the following lines contains an error

# Correct it so that it runs without producing syntax errors

birthday = "2017-07-14"

case\_id = 'DATACAMP!123-456?'

-Fun with Function

-Load a data frame :

# Import pandas

import pandas as pd

# Load the 'ransom.csv' into a DataFrame

r = pd.read\_csv('ransom.csv')

# Display DataFrame

print(r)

-correcting the function error :

# One or more of the following lines contains an error

# Correct it so that it runs without producing syntax errors

# Plot a graph

plt.plot(x\_values, y\_values)

# Display the graph

plt.show()

-Snooping for suspects :

# Define plate to represent a plate beginning with FRQ

# Use \* to represent the missing four letters

plate = 'FRQ\*\*\*\*'

# Call the function lookup\_plate()

lookup\_plate(plate)

# Call lookup\_plate() with the keyword argument for color

lookup\_plate(plate, color='Green')

**#LOADING DATA IN PANDAS**

-What is pandas ?

-Displaying a data frame :

**\*df = pd.read\_csv(‘filename.csv’)**

**\*print(df)**

**\*print(df.head())** -> inspect top 5 data

**\*print(df.info())** -> to see about number of column, row and etc

-Loading a Data Frame :

# Import pandas under the alias pd

import pandas as pd

# Load the CSV "credit\_records.csv"

credit\_records = pd.read\_csv("credit\_records.csv")

# Display the first five rows of credit\_records using the .head() method

print(credit\_records.head())

-Inspecting a data frame :

#Use .info() to inspect the DataFrame credit\_records

print(credit\_records.info())

-Use in calculation :

**\*credit\_records.price.sum()**

-Plot Data :

**\*plt.plot(ransom[‘letter’], ransom[‘frequency’])**

-print top 5 data :

**\*print(credit\_records.head())**

-Selecting with brackets and string (subsetting column):

**\*suspect = credit\_records[‘suspect’]**

-Selecting with a dot :

\*price = credit\_records.price

-Common mistakes in column selection :

1.use square bracket if there is special character in column name

\*police\_report[‘Is golden retriever?’]

Not

\*police\_report.Is golden retriever?

2.when subsetting column, using quotation mark

\*credit\_report[‘location’]

Not

\*credit\_report[location]

3.Using square bracket not parentheses

\*credit\_report[‘location’]

Not

\*credit\_report(‘location’)

-Two method for selecting column :

# Select the column item from credit\_records

# Use brackets and string notation

items = credit\_records['item']

# Display the results

print(items)

# Select the column item from credit\_records

# Use dot notation

items = credit\_records.item

# Display the results

print(items)

-Correcting column selection errors :

# One or more lines of code contain errors.

# Fix the errors so that the code runs.

# Select the location column in credit\_records

location = credit\_records['location']

# Select the item column in credit\_records

items = credit\_records.item

# Display results

print(location)

print(items)

-more column selection mistakes :

# Use info() to inspect mpr

print(mpr.info())

# Use info() to inspect mpr

print(mpr.info())

# The following code contains one or more errors

# Correct the mistakes in the code so that it runs without errors

# Select column "Dog Name" from mpr

name = mpr['Dog Name']

# Select column "Missing?" from mpr

is\_missing = mpr['Missing?']

# Display the columns

print(name)

print(is\_missing)

-Selecting rows with logic :

-Using logic with data frames :

**\*credit\_records.price > 20**

-Filter the result only with true answer :

**\*credit\_records[credit\_records.price > 20]** -> selecting only price > 20

**\*credit\_records[credit\_records.suspect==’Ronald’]** -> selecting only suspect with Ronalds

-Logical Testing :

# Is height\_inches greater than 70 inches?

print(height\_inches > 70)

# Is plate1 equal to "FRQ123"?

print(plate1 == "FRQ123")

# Is fur\_color not equal to "brown"?

print(fur\_color != "brown")

-Selecting Missing Puppies :

# Select the dogs where Age is greater than 2

greater\_than\_2 = mpr[mpr.Age > 2]

print(greater\_than\_2)

# Select the dogs whose Status is equal to Still Missing

still\_missing = mpr[mpr.Status == 'Still Missing']

print(still\_missing)

# Select all dogs whose Dog Breed is not equal to Poodle

not\_poodle = mpr[mpr['Dog Breed'] != 'Poodle']

print(not\_poodle)

-Narrowing the list of suspects :

# Select purchases from 'Pet Paradise'

purchase = credit\_records[credit\_records.location == 'Pet Paradise']

# Display

print(purchase)

#PLOTTING DATA WITH MATPLOTLIB

-Line Plot

**\*from matplotlib import pyplot as plt**

**\*plt.plot(x\_value, y\_value)**

**\*plt.show()**

-Multiple Line plot

**\*plt.plot(x1, y1)**

**\*plt.plot(x2,y2)**

**\*plt.show()**

-Working Hard

# From matplotlib, import pyplot under the alias plt

from matplotlib import pyplot as plt

# Plot Officer Deshaun's hours\_worked vs. day\_of\_week

plt.plot(deshaun.day\_of\_week, deshaun.hours\_worked)

# Display Deshaun's plot

plt.show()

-Or Hardly working ?

# Plot Officer Deshaun's hours\_worked vs. day\_of\_week

plt.plot(deshaun.day\_of\_week, deshaun.hours\_worked)

# Plot Officer Aditya's hours\_worked vs. day\_of\_week

plt.plot(aditya.day\_of\_week, aditya.hours\_worked)

# Plot Officer Mengfei's hours\_worked vs. day\_of\_week

plt.plot(mengfei.day\_of\_week, mengfei.hours\_worked)

# Display all three line plots

plt.show()

-Axes and title labels

**\*plt.xlabel (“Letter”)**

**\*plt.ylabel (“Frequency”)**

**\*plt.title (“Ransom Note Letters”)**

-Legends

**\*plt.plot(x1,y1, label = “x”)**

**\*plt.plot(x2,y2, label = “y”)**

**\*plt.plot(x3,y3, label = “z”)**

**\*plt.legend()**

-Arbitrary text : Put the text in the middle of graph based on coordinate

**\*plt.text(xcoord, ycoord, “Text Message”)**

**\*plt.text(5,9,”Unusually low h frequency”)**

-Modifying Text

-Change font size :

**\*plt.title(“Plot title”, fontsize = 20)**

-Change font color :

**\*plt.legend(color = “green”)**

Source of color = en.wikipedia.org/wiki/Web\_colors

-Adding a legend

# Officer Deshaun

plt.plot(deshaun.day\_of\_week, deshaun.hours\_worked, label='Deshaun')

# Add a label to Aditya's plot

plt.plot(aditya.day\_of\_week, aditya.hours\_worked, label='Aditya')

# Add a label to Mengfei's plot

plt.plot(mengfei.day\_of\_week, mengfei.hours\_worked, label='Mengfei')

# Add a command to make the legend display

plt.legend()

# Display plot

plt.show()

-Adding Labels

# Lines

plt.plot(deshaun.day\_of\_week, deshaun.hours\_worked, label='Deshaun')

plt.plot(aditya.day\_of\_week, aditya.hours\_worked, label='Aditya')

plt.plot(mengfei.day\_of\_week, mengfei.hours\_worked, label='Mengfei')

# Add a title

plt.title("Officers Schedule")

# Add y-axis label

plt.ylabel("Hours Worked")

# Legend

plt.legend()

# Display plot

plt.show()

-Adding floating text

# Create plot

plt.plot(six\_months.month, six\_months.hours\_worked)

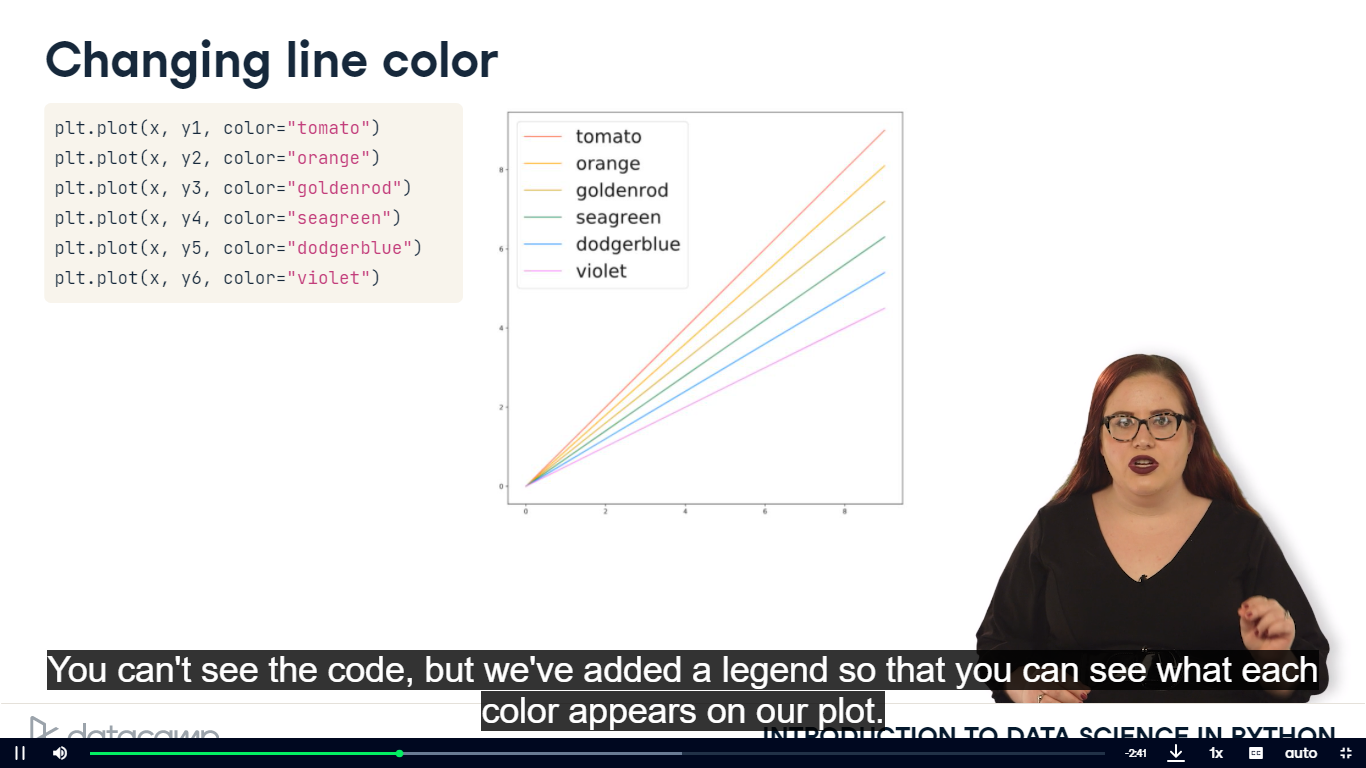
# Add annotation "Missing June data" at (2.5, 80)

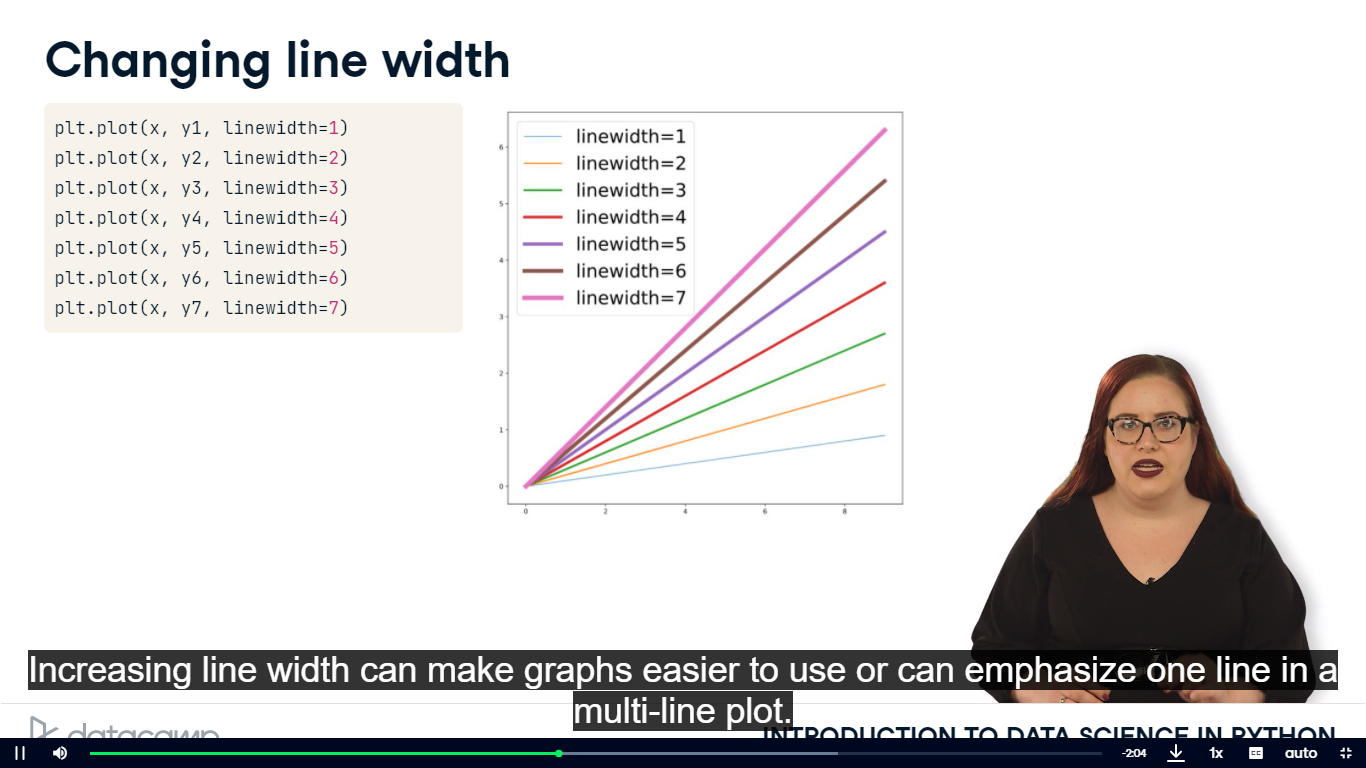
plt.text(2.5, 80, "Missing June data")

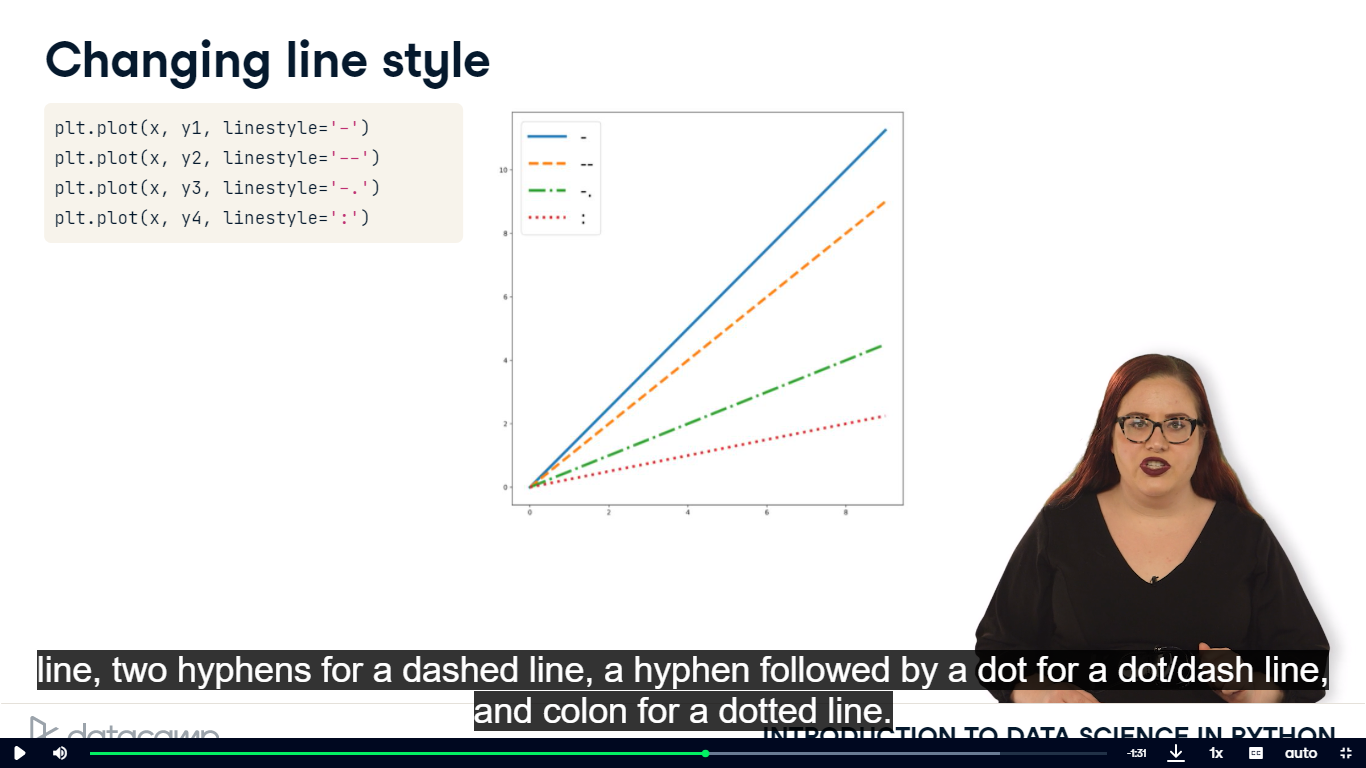
# Display graph

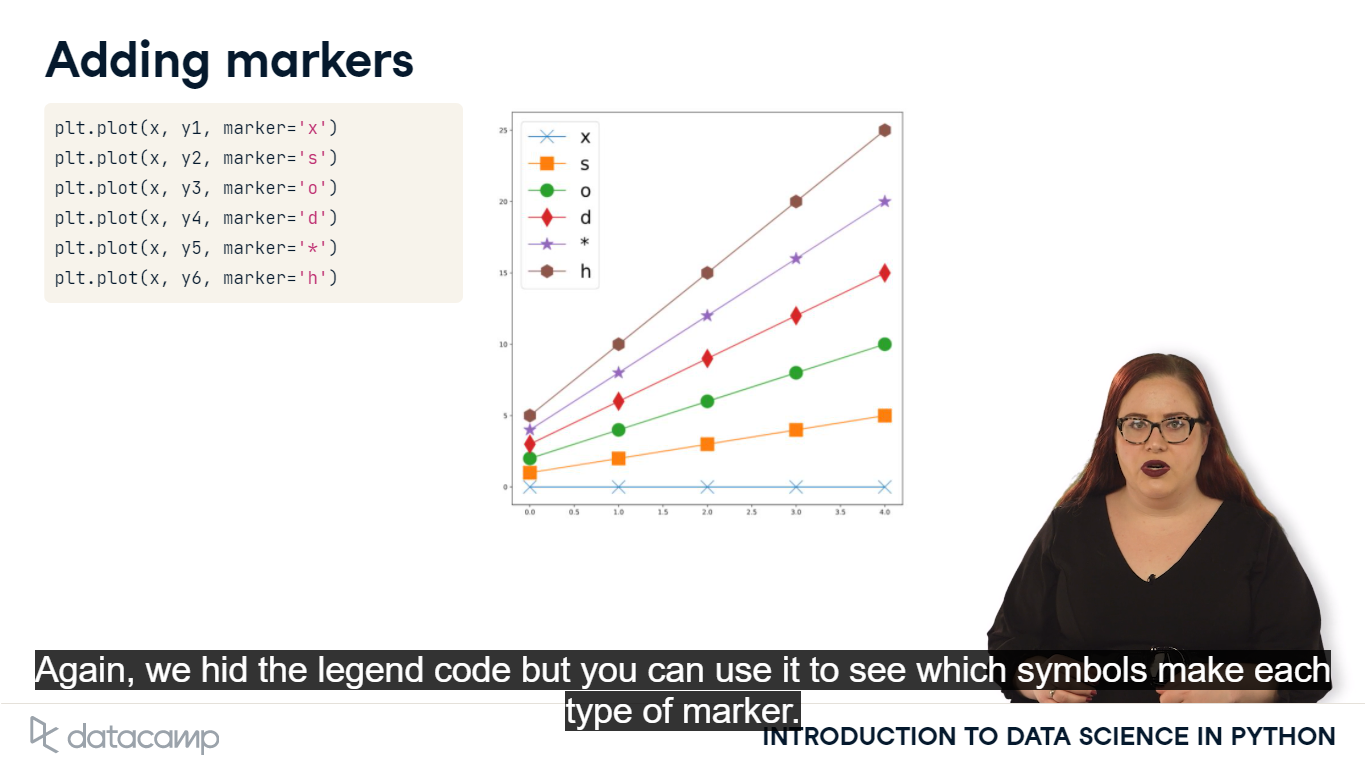
plt.show()

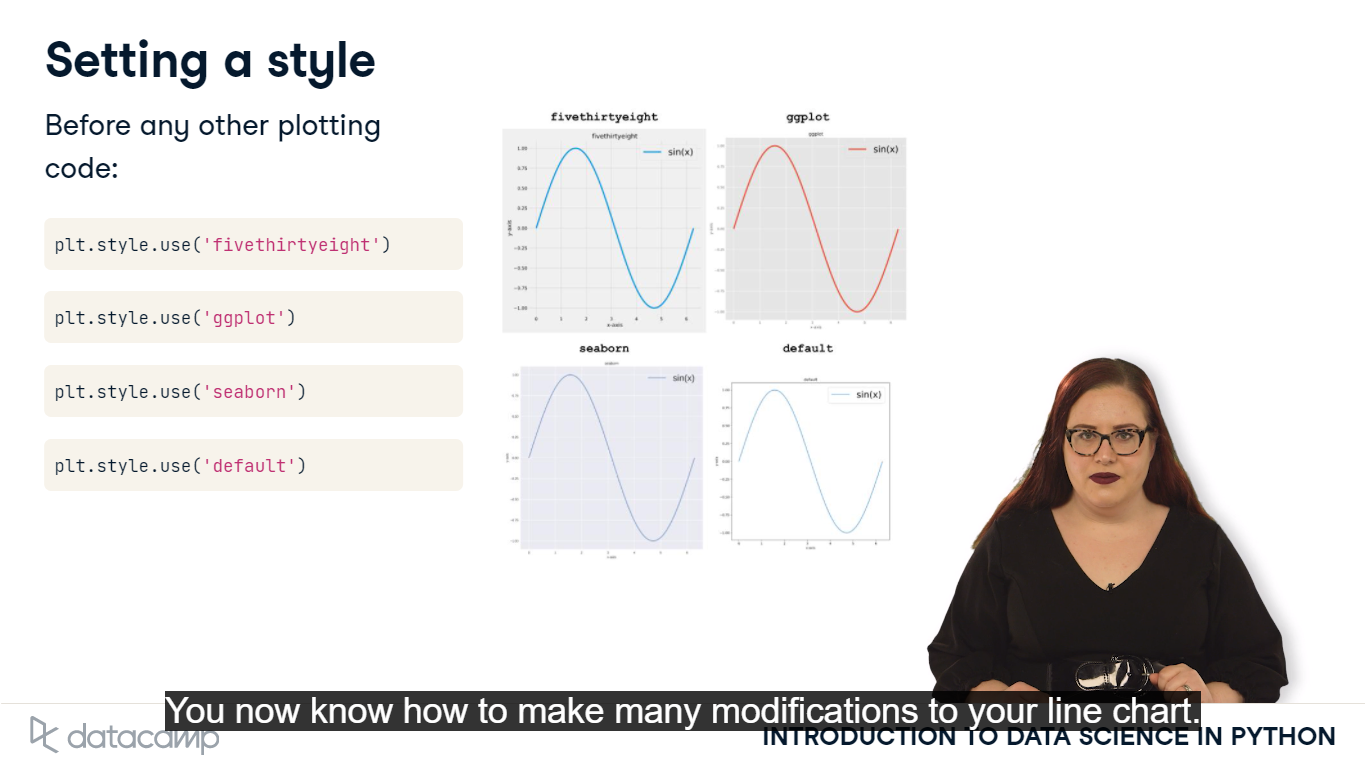
-Styling Graphs











-Tracking Crime Statistics

# Change the color of Phoenix to `"DarkCyan"`

plt.plot(data["Year"], data["Phoenix Police Dept"], label="Phoenix", color = "DarkCyan")

# Make the Los Angeles line dotted

plt.plot(data["Year"], data["Los Angeles Police Dept"], label="Los Angeles", linestyle = ':')

# Add square markers to Philedelphia

plt.plot(data["Year"], data["Philadelphia Police Dept"], label="Philadelphia", marker = 's')

# Add a legend

plt.legend()

# Display the plot

plt.show()

-Playing with styles

# Change the style to fivethirtyeight

plt.style.use('fivethirtyeight')

# Plot lines

plt.plot(data["Year"], data["Phoenix Police Dept"], label="Phoenix")

plt.plot(data["Year"], data["Los Angeles Police Dept"], label="Los Angeles")

plt.plot(data["Year"], data["Philadelphia Police Dept"], label="Philadelphia")

# Add a legend

plt.legend()

# Display the plot

plt.show()

# Change the style to ggplot

plt.style.use("ggplot")

# Plot lines

plt.plot(data["Year"], data["Phoenix Police Dept"], label="Phoenix")

plt.plot(data["Year"], data["Los Angeles Police Dept"], label="Los Angeles")

plt.plot(data["Year"], data["Philadelphia Police Dept"], label="Philadelphia")

# Add a legend

plt.legend()

# Display the plot

plt.show()

# Choose any of the styles

print(plt.style.available) -> to see style available

plt.style.use("ggplot")

# Plot lines

plt.plot(data["Year"], data["Phoenix Police Dept"], label="Phoenix")

plt.plot(data["Year"], data["Los Angeles Police Dept"], label="Los Angeles")

plt.plot(data["Year"], data["Philadelphia Police Dept"], label="Philadelphia")

# Add a legend

plt.legend()

# Display the plot

plt.show()

-Identifying bayes kidnapper

# x should be ransom.letter and y should be ransom.frequency

plt.plot(ransom.letter, ransom.frequency,

         # Label should be "Ransom"

         label="Ransom",

         # Plot the ransom letter as a dotted gray line

         linestyle=':', color='gray')

# Display the plot

plt.show()

# Plot each line

plt.plot(ransom.letter, ransom.frequency,

         label='Ransom', linestyle=':', color='gray')

# X-values should be suspect1.letter

# Y-values should be suspect1.frequency

# Label should be "Fred Frequentist"

plt.plot(suspect1.letter, suspect1.frequency, label='Fred Frequentist')

# Display the plot

plt.show()

# Plot each line

plt.plot(ransom.letter, ransom.frequency,

         label='Ransom', linestyle=':', color='gray')

plt.plot(suspect1.letter, suspect1.frequency,

         label='Fred Frequentist')

# X-values should be suspect2.letter

# Y-values should be suspect2.frequency

# Label should be "Gertrude Cox"

plt.plot(suspect2.letter, suspect2.frequency, label = 'Gertrude Cox')

# Display plot

plt.show()

# Plot each line

plt.plot(ransom.letter, ransom.frequency,

         label='Ransom', linestyle=':', color='gray')

plt.plot(suspect1.letter, suspect1.frequency, label='Fred Frequentist')

plt.plot(suspect2.letter, suspect2.frequency, label='Gertrude Cox')

# Add x- and y-labels

plt.xlabel("Letter")

plt.ylabel("Frequency")

# Add a legend

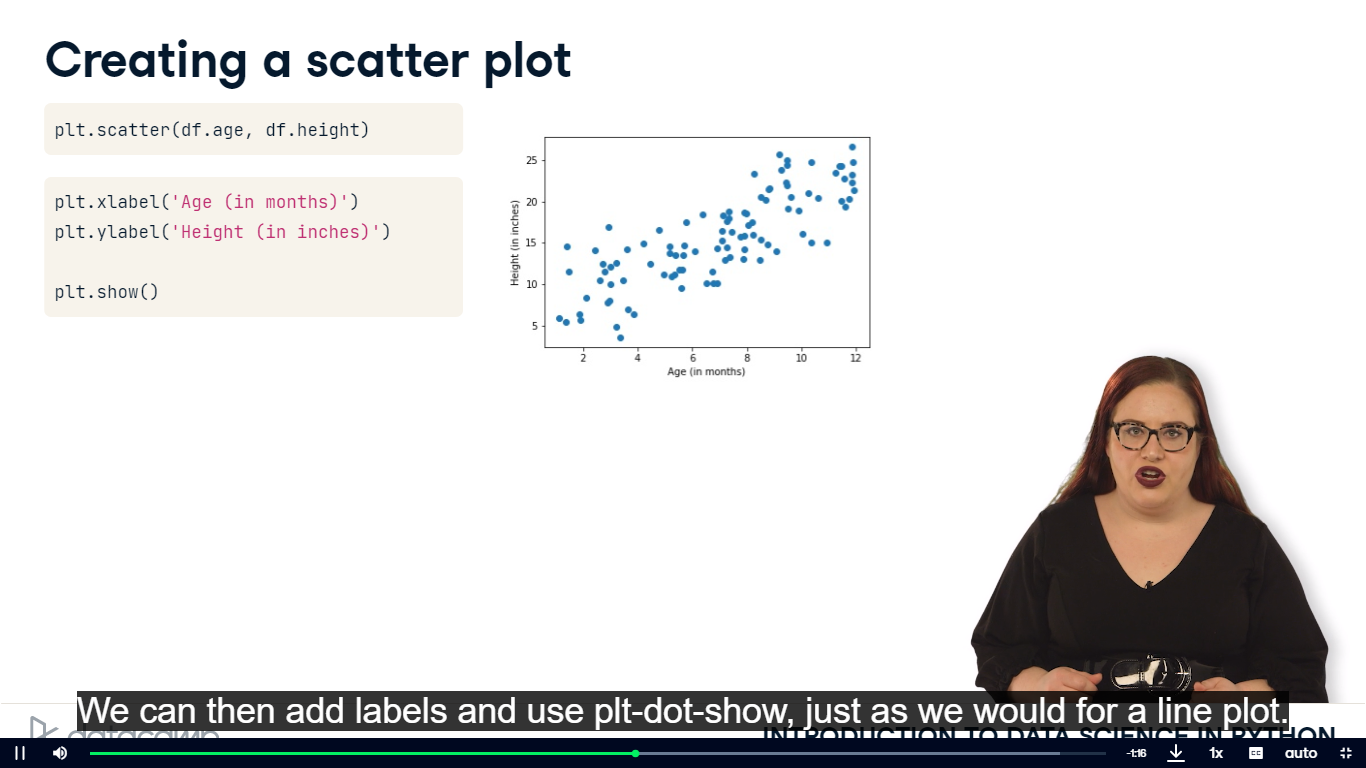
plt.legend()

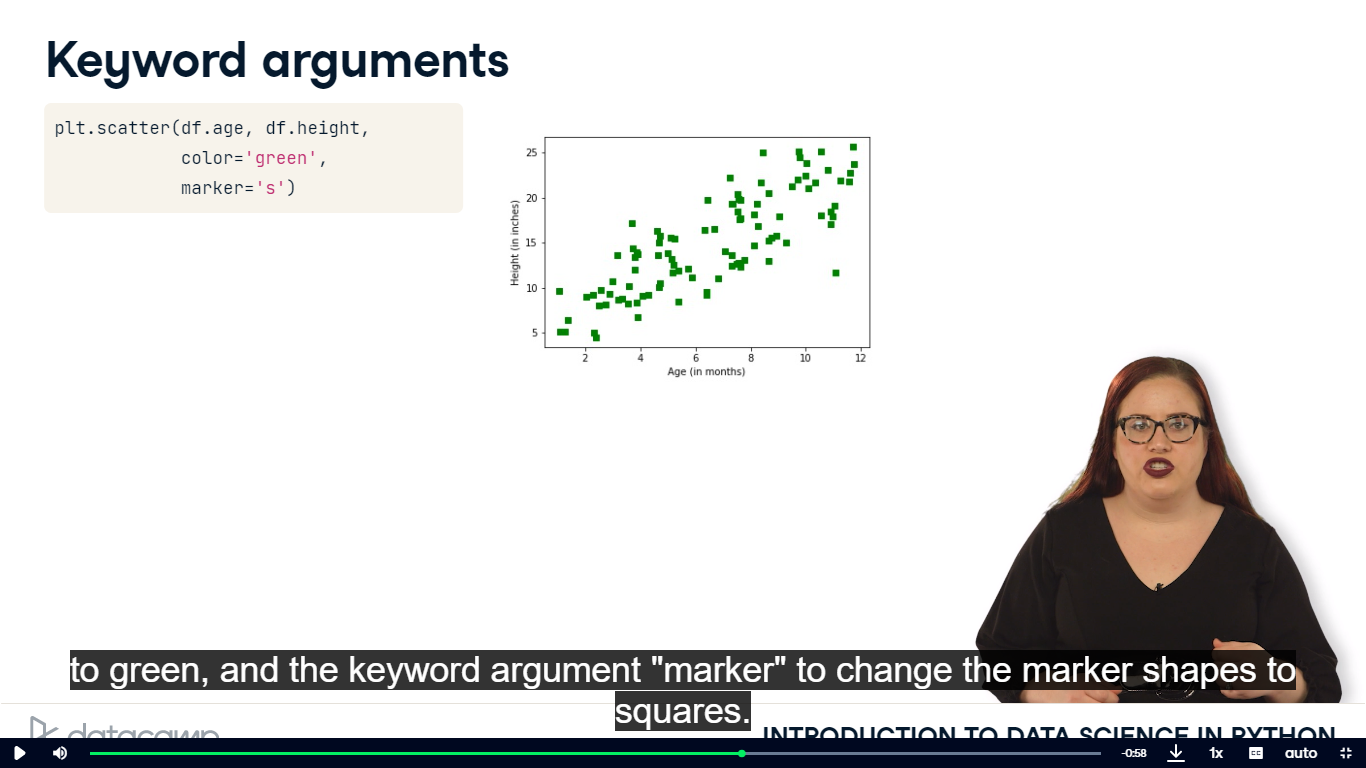
# Display plot

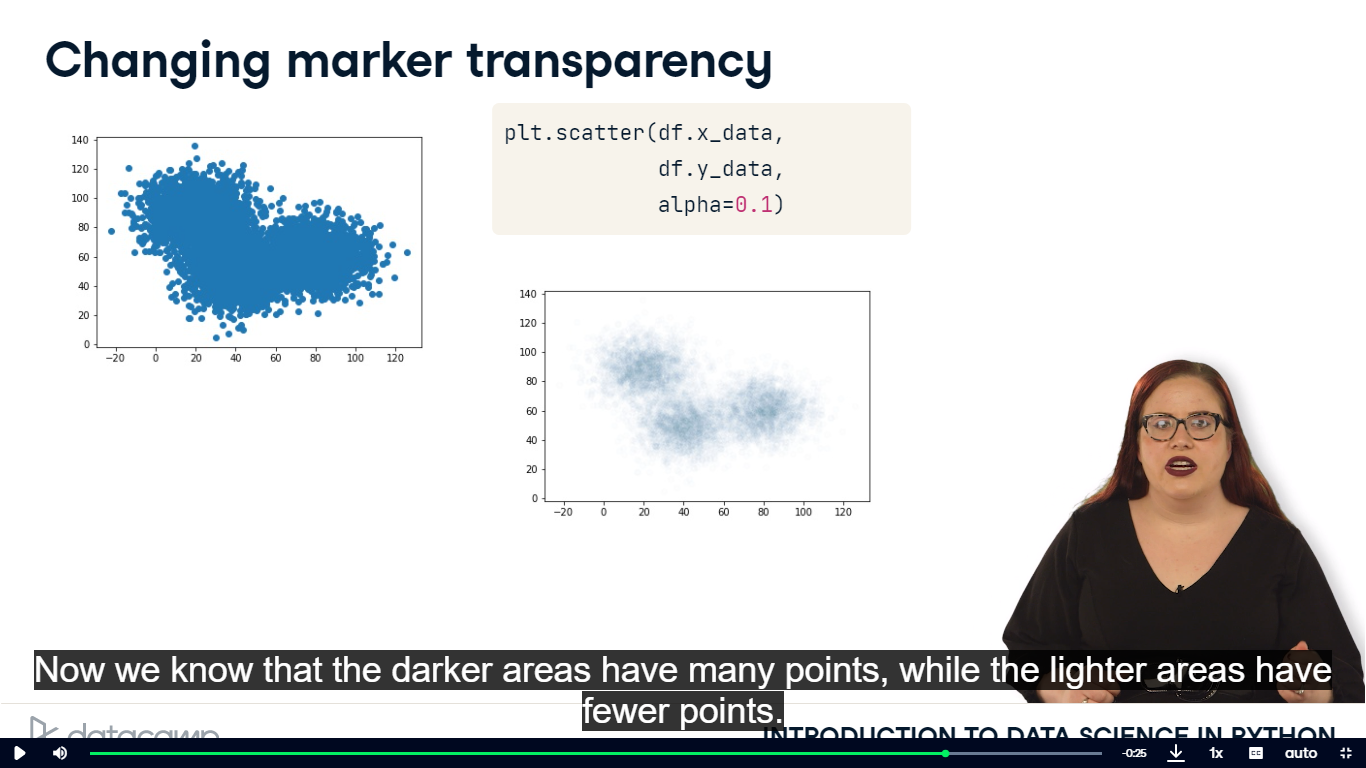
plt.show()

**#DIFFERENT TYPES OF PLOT**

-Scatter Plot







-Charting Cellphone data

# Explore the data

print(cellphone.head())

# Create a scatter plot of the data from the DataFrame cellphone

plt.scatter(cellphone.x, cellphone.y)

# Add labels

plt.ylabel('Latitude')

plt.xlabel('Longitude')

# Display the plot

plt.show()

-Modifying a scatterplot

# Change the transparency to 0.1

plt.scatter(cellphone.x, cellphone.y,

           color='red',

           marker='s',

           alpha=0.1)

# Add labels

plt.ylabel('Latitude')

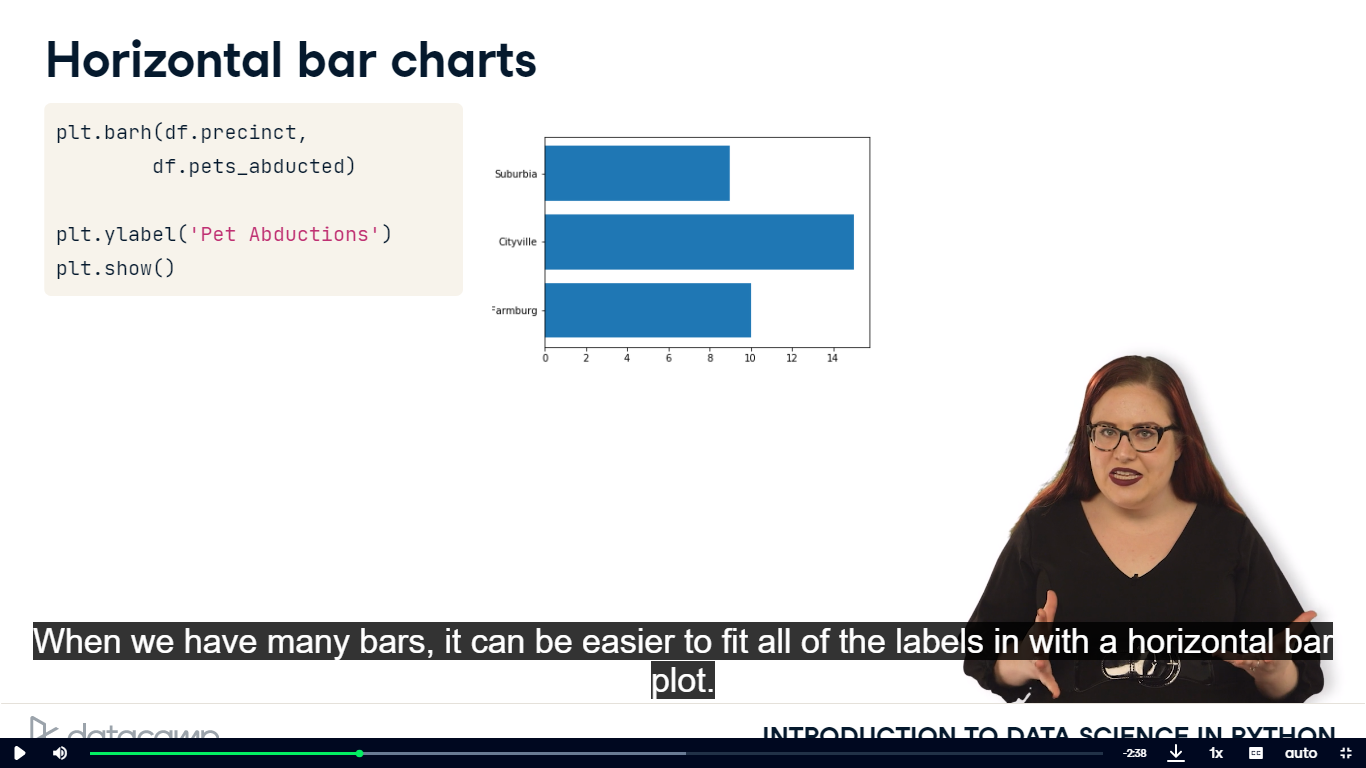
plt.xlabel('Longitude')

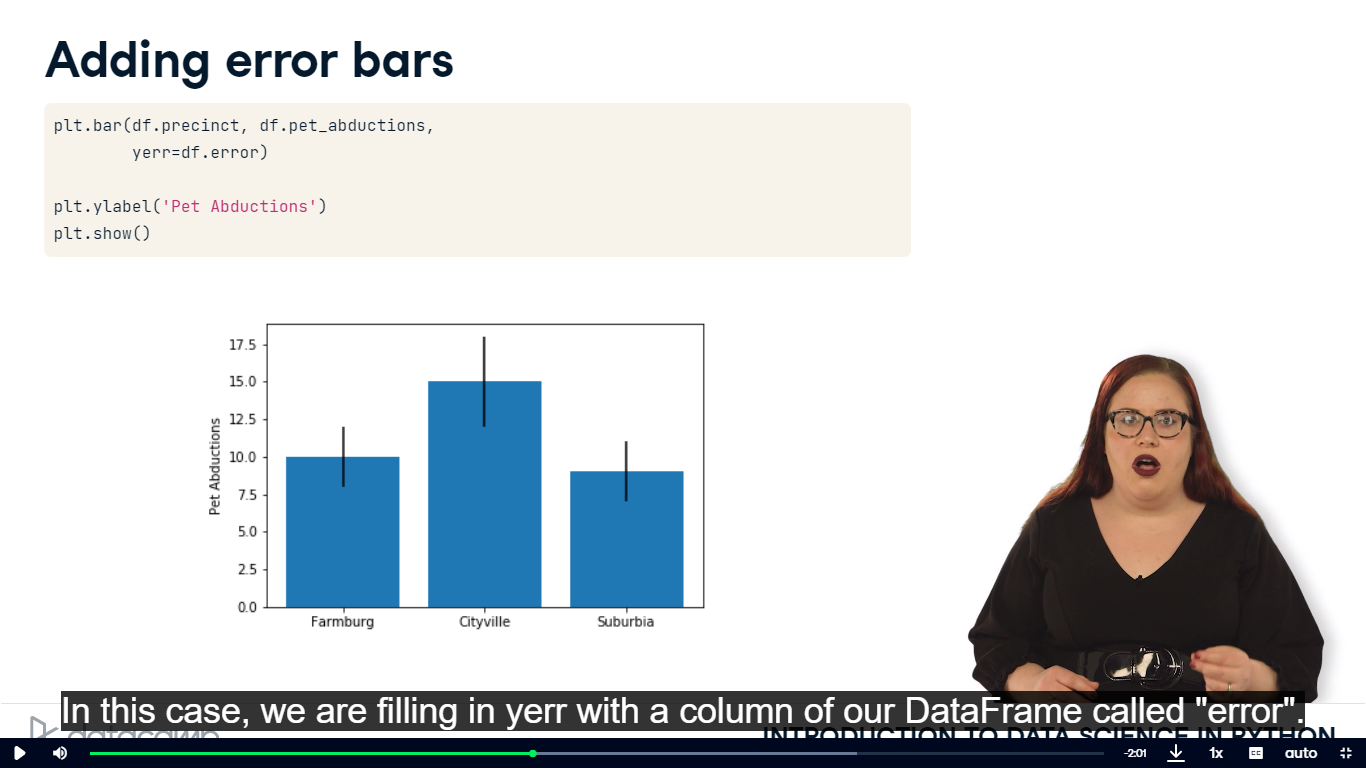
# Display the plot

plt.show()

-Bar Chart









-Build a simple bar chart

# Display the DataFrame hours using print

print(hours)

# Create a bar plot from the DataFrame hours

plt.bar(hours.officer, hours.avg\_hours\_worked,

        # Add error bars

        yerr=hours.std\_hours\_worked)

# Display the plot

plt.show()

-Where did the time go ?

# Plot the number of hours spent on desk work

plt.bar(hours.officer, hours.desk\_work, label='Desk Work')

# Plot the hours spent on field work on top of desk work

plt.bar(hours.officer, hours.field\_work, bottom = hours.desk\_work, label = 'Field Work')

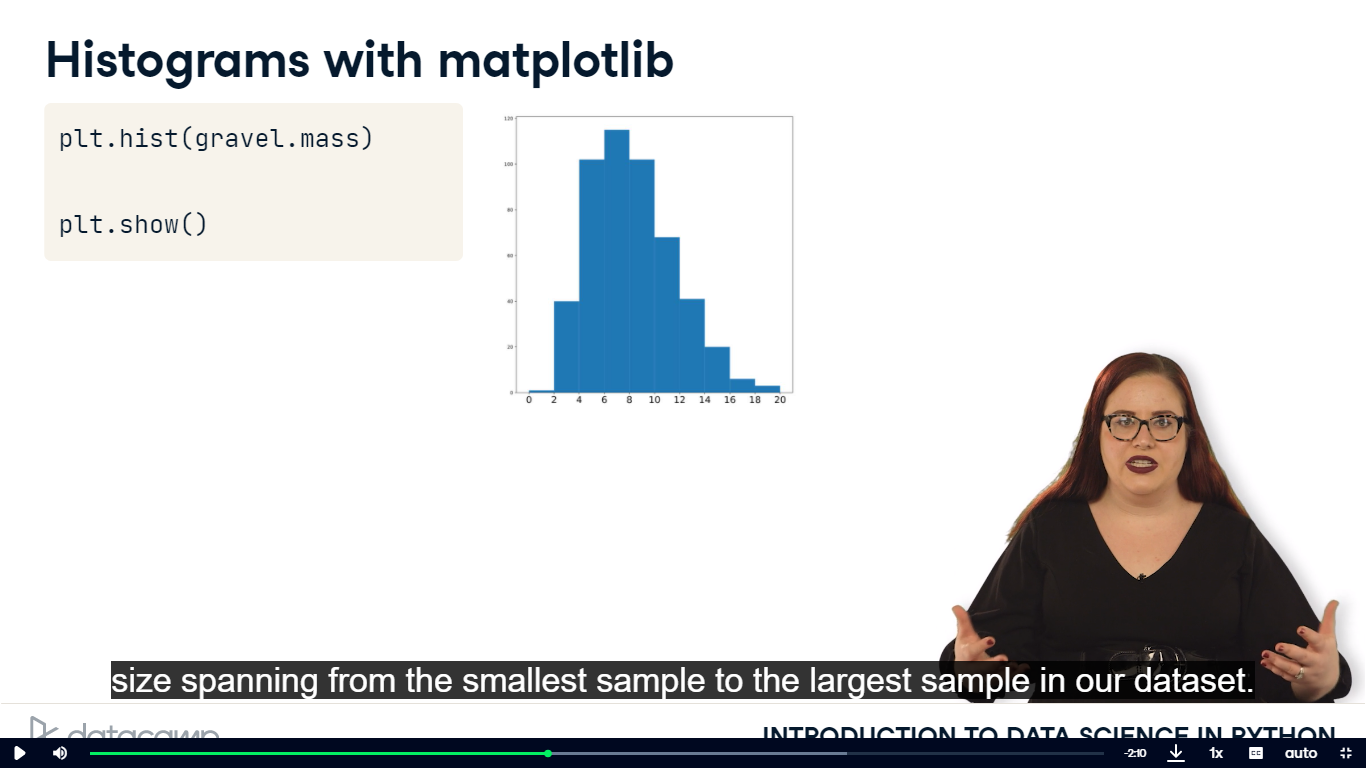
# Add a legend

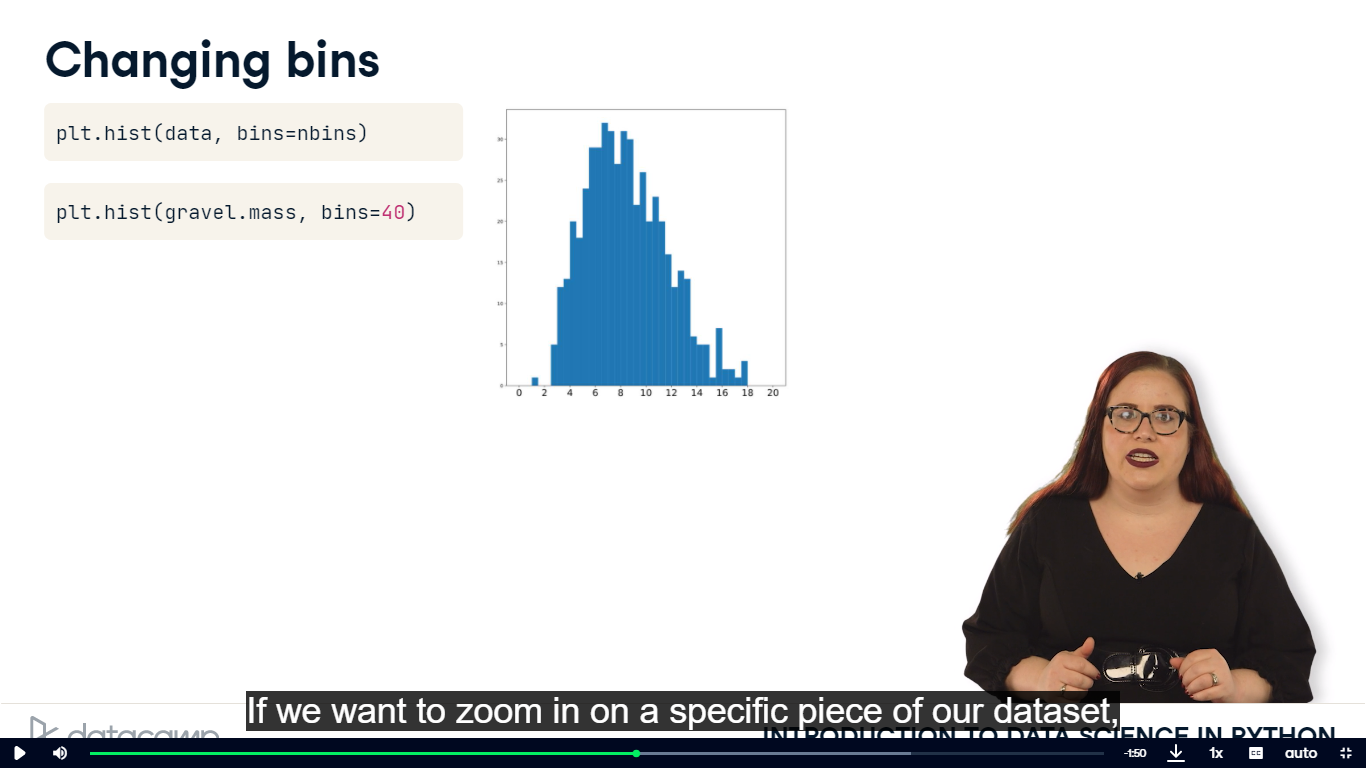
plt.legend()

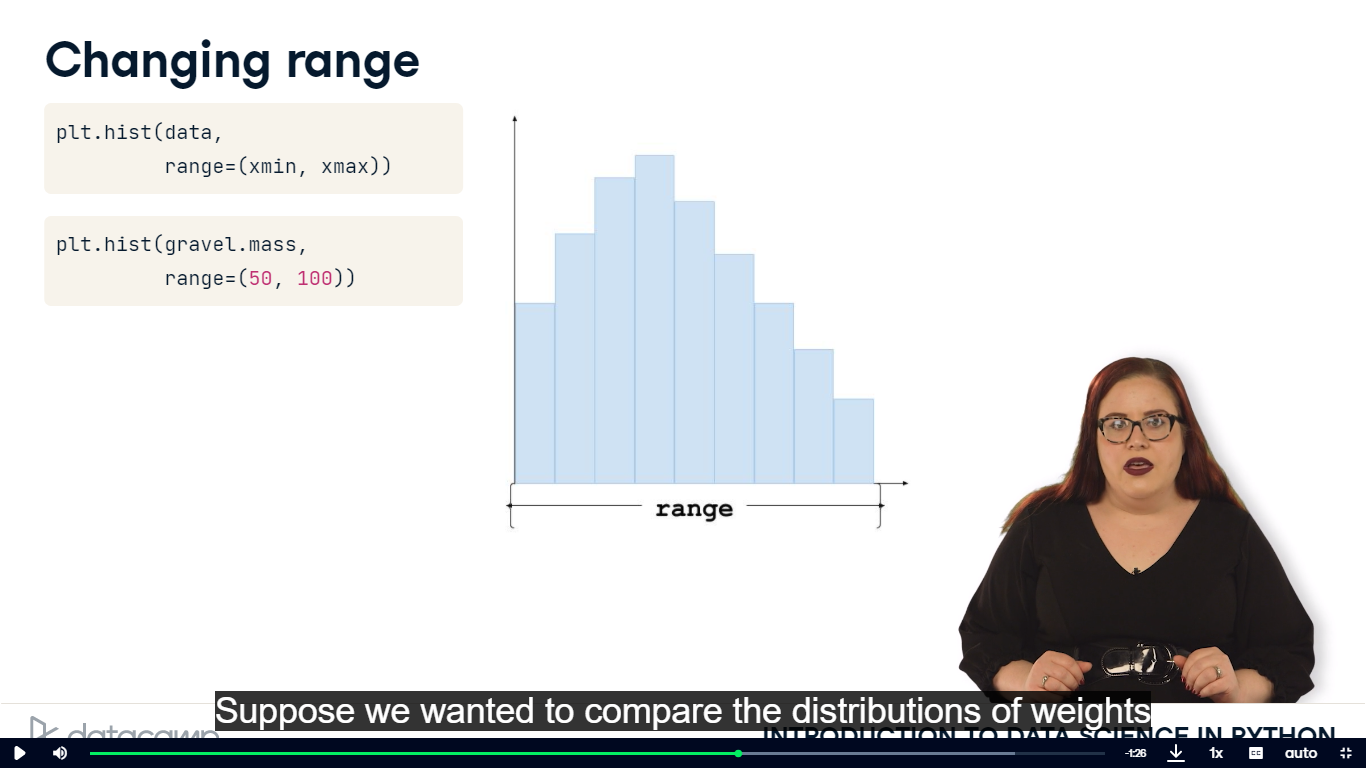
# Display the plot

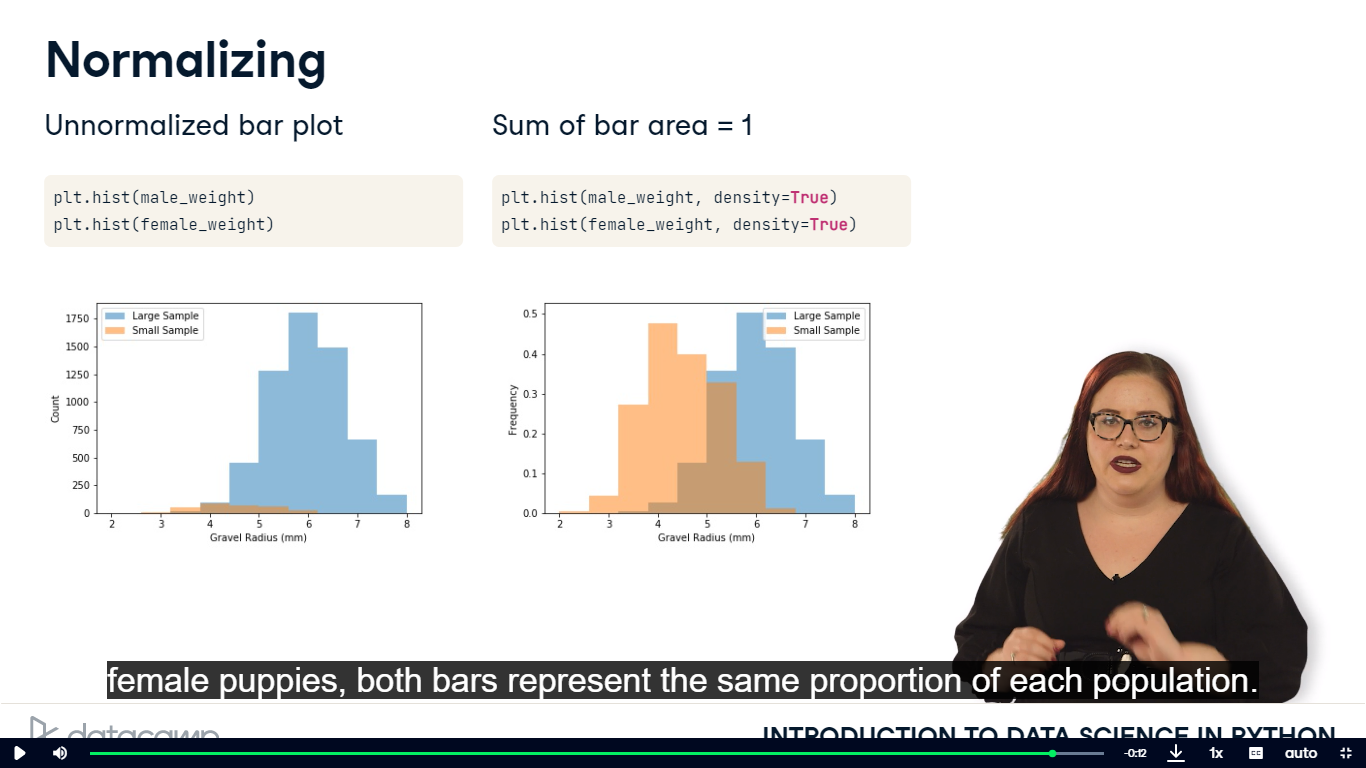
plt.show()

-Histogram









-Modifying Histogram

# Change the number of bins to 50

plt.hist(puppies.weight,

        bins=50)

# Add labels

plt.xlabel('Puppy Weight (lbs)')

plt.ylabel('Number of Puppies')

# Display

plt.show()

# Change the range to start at 5 and end at 35

plt.hist(puppies.weight,

        range=(5, 35))

# Add labels

plt.xlabel('Puppy Weight (lbs)')

plt.ylabel('Number of Puppies')

# Display

plt.show()

-Heroes with histogram

# Create a histogram

plt.hist(gravel.radius,

         bins=40,

         range=(2, 8),

         density=True)

# Label plot

plt.xlabel('Gravel Radius (mm)')

plt.ylabel('Frequency')

plt.title('Sample from Shoeprint')

# Display histogram

plt.show()