**1. Loading your friend's data into a dictionary**

*# Create the years and durations lists*

years = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]

durations = [103, 101, 99, 100, 100, 95, 95, 96, 93, 90]

*# Create a dictionary with the two lists*

movie\_dict = {'years': years, 'durations': durations}

*# Print the dictionary*

movie\_dict

{'years': [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020],

'durations': [103, 101, 99, 100, 100, 95, 95, 96, 93, 90]}

%%**nose**

test\_years = [2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]

test\_durations = [103, 101, 99, 100, 100, 95, 95, 96, 93, 90]

test\_movie\_dict = {'years': test\_years, 'durations': test\_durations}

def test\_years\_list():

assert (type(years) == list), \

'Did you correctly initalize a `years` as a list?'

assert len(test\_years) == len(years), \

"Did you correctly define the `years` list as a list containing \*\*all\*\* 10 years from 2011 to 2020?"

assert test\_years == years, \

"Did you correctly define the `years` list as a list containing the years (in order) from 2011 to 2020?"

def test\_durations\_list():

assert (type(durations) == list), \

'Did you correctly initalize a `durations` as a list?'

assert len(test\_durations) == len(durations), \

"Did you correctly define the `durations` list as a list containing \*\*all\*\* 10 average durations our friend provided us?"

assert test\_durations == durations, \

"Did you correctly define the `durations` list as a list containing all of the average movie durations (in order) that our friend provided us?"

def test\_movie\_dict\_dict():

assert (type(movie\_dict) == dict), \

'Did you correctly initalize `movie\_dict` as a dictionary?'

assert len(test\_durations) == len(durations), \

"Did you correctly define the `movie\_dict` dictionary as a two-element dictionary containing the years and durations?"

assert list(movie\_dict.keys()) == ['years', 'durations'], \

"Did you correctly define the `movie\_dict` dictionary as a two-element dictionary containing the keys `\"years\"` and `\"durations\"`?"

assert list(movie\_dict['years']) == test\_years, \

"Does your `movie\_dict` dictionary contain a key `\"years\"` with the value set to the `years` list you created above?"

assert list(movie\_dict['durations']) == test\_durations, \

"Does your `movie\_dict` dictionary contain a key `\"durations\"` with the value set to the `durations` list you created above?"

## 2. Creating a DataFrame from a dictionary

*# Import pandas under its usual alias*

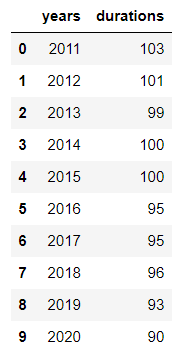
**import** **pandas** **as** **pd**

*# Create a DataFrame from the dictionary*

durations\_df = pd.DataFrame(movie\_dict)

*# Print the DataFrame*

durations\_df



## 3. A visual inspection of our data

*# Import matplotlib.pyplot under its usual alias and create a figure*

**import** **matplotlib.pyplot** **as** **plt**

fig = plt.figure()

*# Draw a line plot of release\_years and durations*

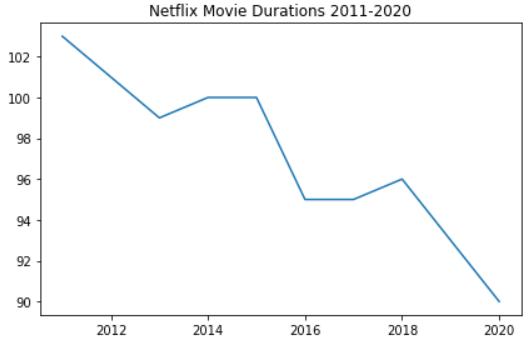
plt.plot(durations\_df['years'], durations\_df['durations'])

*# Create a title*

plt.title("Netflix Movie Durations 2011-2020")

*# Show the plot*

plt.show()



## 4. Loading the rest of the data from a CSV

*# Read in the CSV as a DataFrame*

netflix\_df = pd.read\_csv("datasets/netflix\_data.csv")

*# Print the first five rows of the DataFrame*

netflix\_df[0:5]



## 5. Filtering for movies!

*# Subset the DataFrame for type "Movie"*

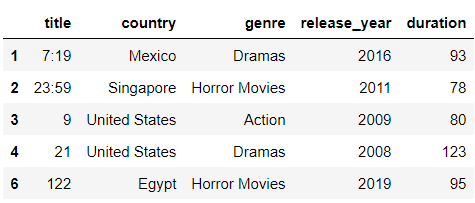
netflix\_df\_movies\_only = netflix\_df[netflix\_df['type'] == 'Movie']

*# Select only the columns of interest*

netflix\_movies\_col\_subset = netflix\_df\_movies\_only[['title', 'country', 'genre', 'release\_year', 'duration']]

*# Print the first five rows of the new DataFrame*

netflix\_movies\_col\_subset[0:5]



## 6. Creating a scatter plot

*# Create a figure and increase the figure size*

fig = plt.figure(figsize=(12,8))

*# Create a scatter plot of duration versus year*

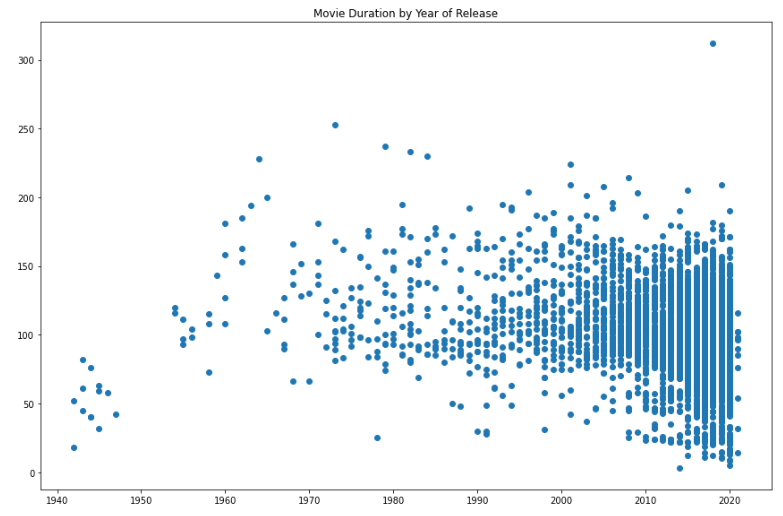
plt.scatter(netflix\_movies\_col\_subset["release\_year"], netflix\_movies\_col\_subset["duration"])

*# Create a title*

plt.title("Movie Duration by Year of Release")

*# Show the plot*

plt.show()



## 7. Digging deeper

*# Filter for durations shorter than 60 minutes*

short\_movies = short\_movies = netflix\_movies\_col\_subset[netflix\_movies\_col\_subset['duration'] < 60]

*# Print the first 20 rows of short\_movies*

short\_movies[0:20]



## 8. Marking non-feature films

*# Define an empty list*

colors = []

*# Iterate over rows of netflix\_movies\_col\_subset*

**for** lab, row **in** netflix\_movies\_col\_subset.iterrows():

**if** row['genre'] == "Children":

colors.append("red")

**elif** row['genre'] == "Documentaries":

colors.append("blue")

**elif** row['genre'] == "Stand-Up":

colors.append("green")

**else**:

colors.append("black")

*# Inspect the first 10 values in your list*

colors[0:10]

['black',

'black',

'black',

'black',

'black',

'black',

'black',

'black',

'black',

'blue']

## 9. Plotting with color!

*# Set the figure style and initalize a new figure*

plt.style.use('fivethirtyeight')

fig = plt.figure(figsize=(12,8))

*# Create a scatter plot of duration versus release\_year*

plt.scatter(netflix\_movies\_col\_subset["release\_year"], netflix\_movies\_col\_subset["duration"], c=colors)

*# Create a title and axis labels*

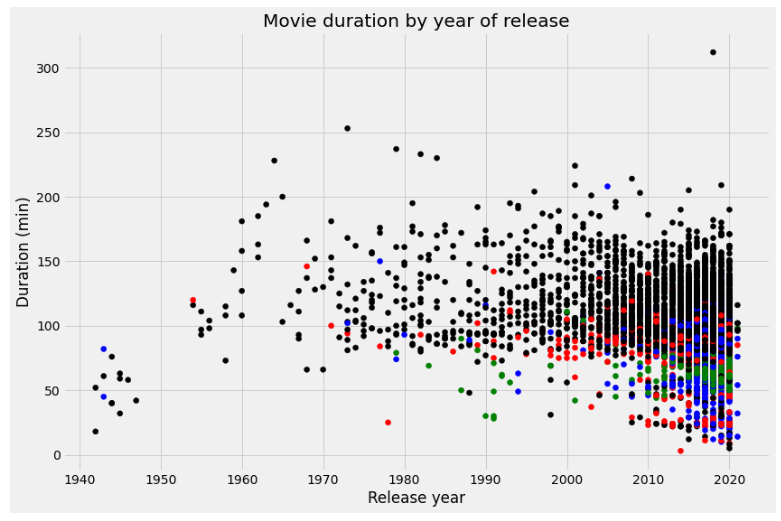
plt.title("Movie duration by year of release")

plt.xlabel("Release year")

plt.ylabel("Duration (min)")

*# Show the plot*

plt.show()



## 10. What next?

*# Are we certain that movies are getting shorter?*

are\_movies\_getting\_shorter = "maybe"