Pandas

-What’s is a library? collections of prewritten code that users can use to optimize tasks

-Pandas : library that has functions for analyzing, cleaning, exploring, and manipulating data.

-Introduce alias(pandas as pd) when importing

**\* A Pandas Series is like a column in a table.**

Ways to create Series:

We load data into a Series object

-pd.Series(list) -> indexes start from 0 to len-1 -> access: series[index]

-pd.Series(list, index = ["label1"," label2"," label3"]) -> labels instead of indexes -> access: series[label]

-pd.Series(dictionary)

**\* Series is like a column, a DataFrame is the whole table( rows and columns).**

Ways to create DataFrames:

#load data into a DataFrame object:

-X = Pd.dataframe( **{** Key1:[item1, item2, item3]

Key2:[item1, item2, item3] **}** )

-X = Pd.dataframe( **{** Key1:[item1, item2, item3]

Key2:[item1, item2, item3] **}** , index = ["label1"," label2"," label3"] )

Pandas use the loc attribute to return one or more specified row(s)

x.loc[0], x.loc[ [0, 1] , “columnName”]

**Use print and type**

Load CSV into a dataframe

df = pd.read\_csv('data.csv')

df.to\_string() to print the entire datframe

Load JSON into a dataframe

-use head(# of rows) to get a quick overview

- use info(), that gives you more information about the data se

**Data cleaning**

-Data cleaning means fixing bad data

* Empty cells
* Data in wrong format
* Wrong data
* Duplicates

\*empty cells : Empty cells can potentially give you a wrong result when you analyze data.

+One way to deal with empty cells is to remove rows that contain empty cells.

This is usually OK, since data sets can be very big, and removing a few rows will not have a big impact on the result.

* Dropna() method will remove all rows with nulls and return a new dataframe, inplace=true will affect the original.

+Another way of dealing with empty cells is to insert a new value instead.

This way you do not have to delete entire rows just because of some empty cells.

* Df.Fillna(value) or Df[‘column’].Fillna(value)
* Replace using mean, mode, median:
* x = df["Calories"].mean()  
    
  df["Calories"].fillna(x, inplace = True)

\*Wrong Format

Cells with data of wrong format can make it difficult, or even impossible, to analyze data.

To fix it, you have two options: remove the rows, or convert all cells in the columns into the same format

df['Date'] = pd.to\_datetime(df['Date'])  
  
print(df.to\_string())

\*wrong data

"Wrong data" does not have to be "empty cells" or "wrong format", it can just be wrong, like if someone registered "199" instead of "1.99".

Sometimes you can spot wrong data by looking at the data set, because you have an expectation of what it should be.

In our example, it is most likely a typo,

Isolate & replace:

df.loc[7, 'Duration'] = 45

create rules, iterate, isolate, replace:

for x in df.index:  
  if df.loc[x, "Duration"] > 120:  
    df.loc[x, "Duration"] = 120

Another way of handling wrong data is to remove the rows that contains wrong data.

This way you do not have to find out what to replace them with, and there is a good chance you do not need them to do your analyses.

for x in df.index:  
  if df.loc[x, "Duration"] > 120:  
    df.drop(x, inplace = True)

\* Removing Duplicates

print(df.duplicated()): returns ture if a row is duplicate

df.drop\_duplicates(inplace = True)

We can access these three main components of a DataFrame with the columns, index, and values attributes.

**Correlation between columns**

df.corr()

**pandas plotting**

df.plot(kind = 'scatter', x = 'Duration', y = 'Maxpulse')  
  
plt.show()