Pandas

Link: https://pandas.pydata.org/

Setup, Load/Save Data, Filtering, Sorting, Grouping, Indexing, Preprocessing, Clean-Up

Installing/Setting Up Panda: https://pandas.pydata.org/docs/getting_started/install.html

Pandas Introduction:

Pandas is used to analyze data.

Pandas is a Python library used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008**.**

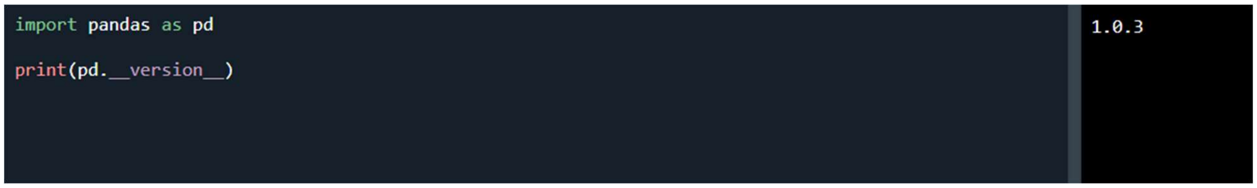
Why Use Pandas?

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets, and make them readable and relevant.

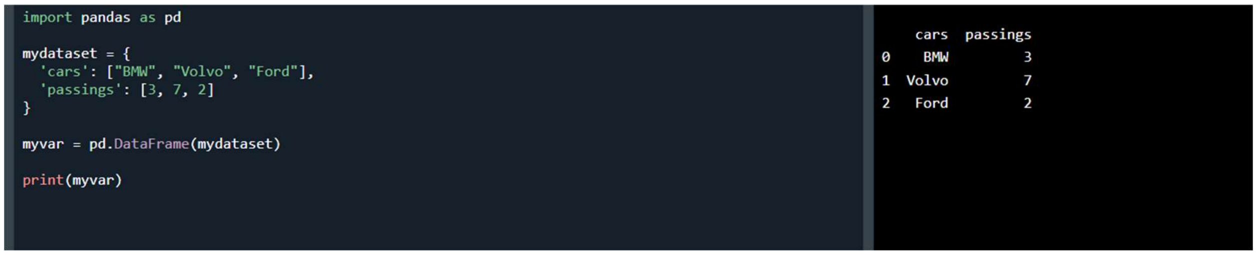
Relevant data is very important in data science.

Version:



```python
import pandas as pd

print(pd.__version__)
```

```
1.0.3
```



```python
import pandas as pd

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```

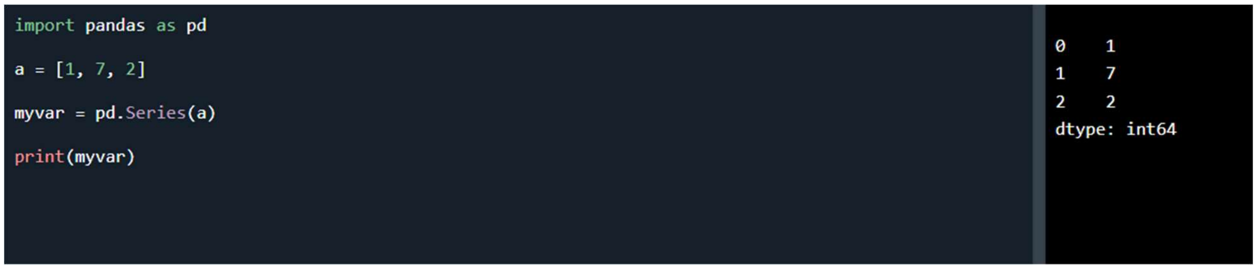
```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

Series:

A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

Pandas



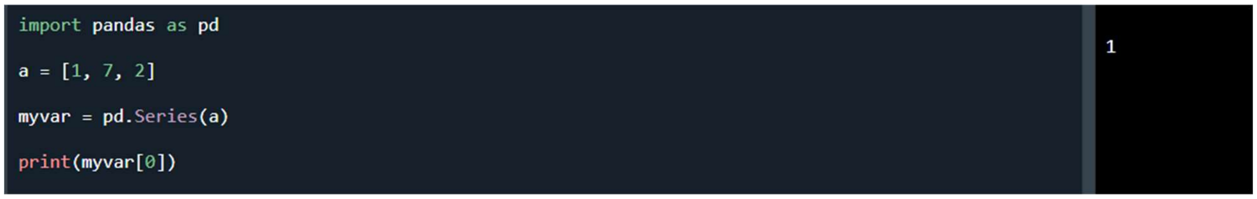
```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

```
0    1
1    7
2    2
dtype: int64
```

Labels

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc.

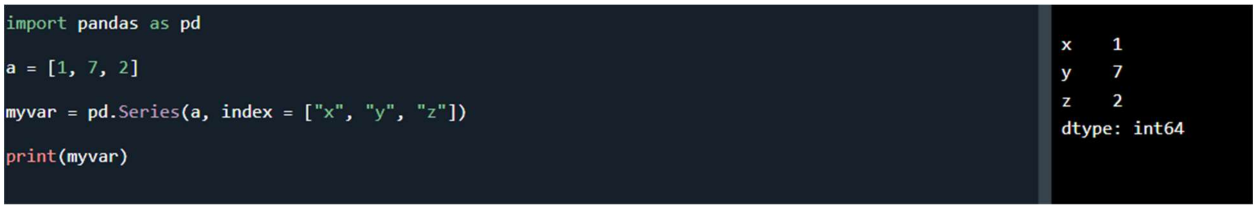
This label can be used to access a specified value.



```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar[0])
```

```
1
```

Create Labels:

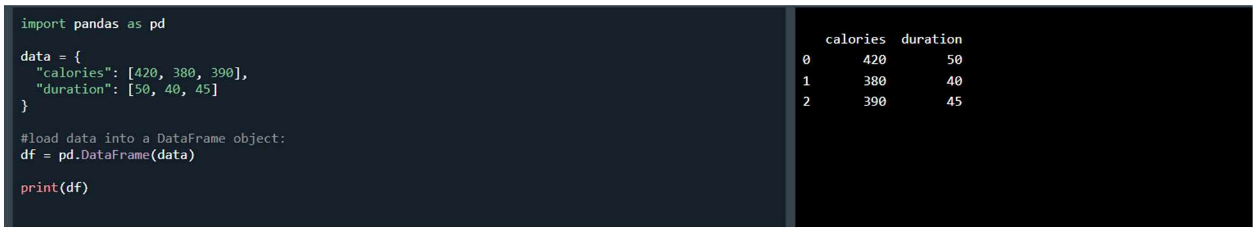


```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

```
x    1
y    7
z    2
dtype: int64
```

What is a DataFrame?

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.



```
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df)
```

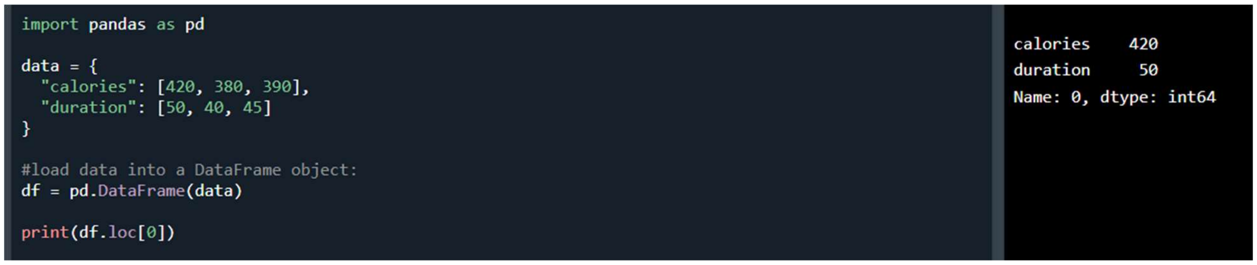
```
   calories  duration
0       420        50
1       380        40
2       390        45
```

Locate Row:

As you can see from the result above, the DataFrame is like a table with rows and columns.

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

Pandas



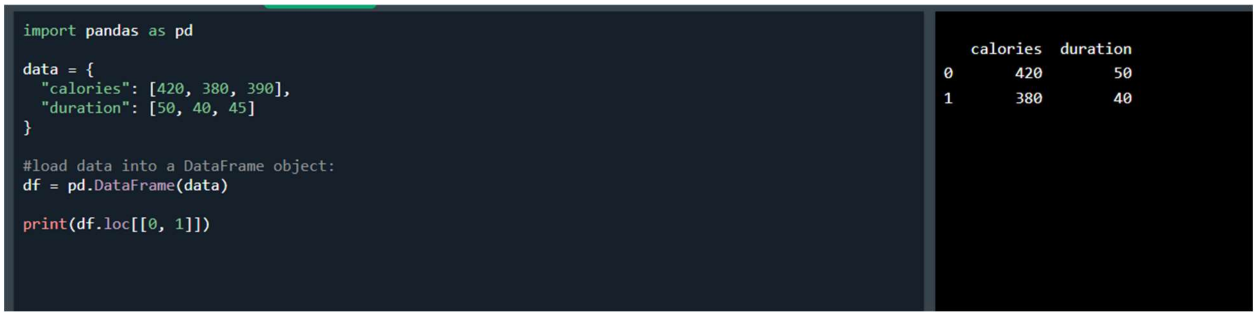
```python
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df.loc[0])
```

```
calories    420
duration     50
Name: 0, dtype: int64
```

Example

Return row 0:

#refer to the row index:
print(df.loc[0])



```python
import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

#load data into a DataFrame object:
df = pd.DataFrame(data)

print(df.loc[[0, 1]])
```

```
   calories  duration
0       420        50
1       380        40
```

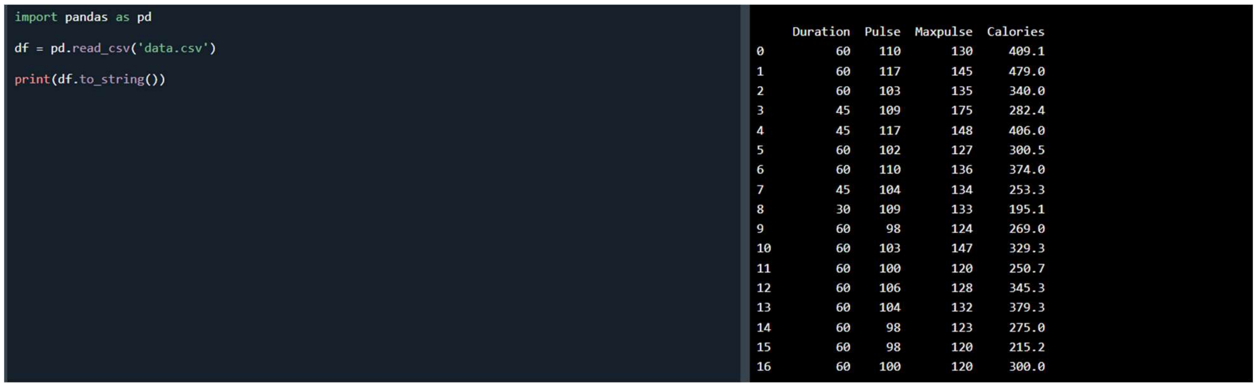
Load Data:

Read CSV Files

A simple way to store big data sets is to use CSV files (comma separated files).

CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

In our examples we will be using a CSV file called 'data.csv'.

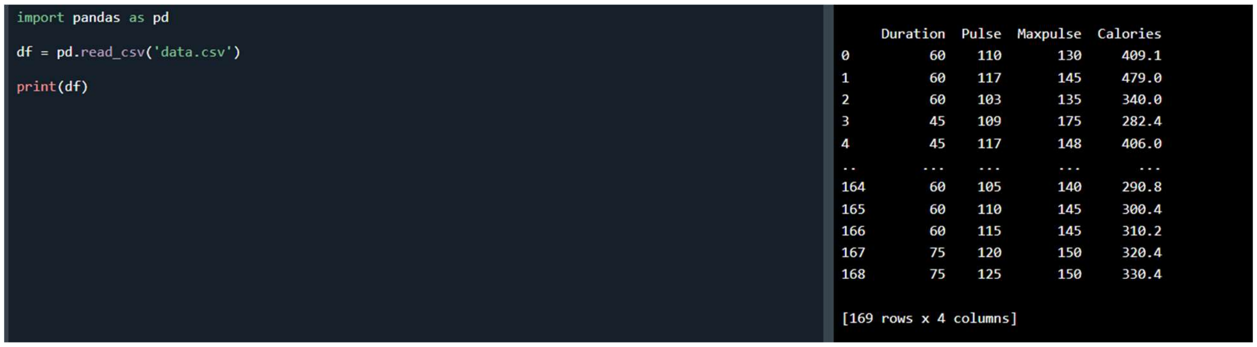


```python
import pandas as pd

df = pd.read_csv('data.csv')

print(df.to_string())
```

```
    Duration  Pulse  Maxpulse  Calories
0         60    110       130     409.1
1         60    117       145     479.0
2         60    103       135     340.0
3         45    109       175     282.4
4         45    117       148     406.0
5         60    102       127     300.5
6         60    110       136     374.0
7         45    104       134     253.3
8         30    109       133     195.1
9         60     98       124     269.0
10        60    103       147     329.3
11        60    100       120     250.7
12        60    106       128     345.3
13        60    104       132     379.3
14        60     98       123     275.0
15        60     98       120     215.2
16        60    100       120     300.0
```

If you have a large DataFrame with many rows, Pandas will only return the first 5 rows, and the last 5 rows

Pandas



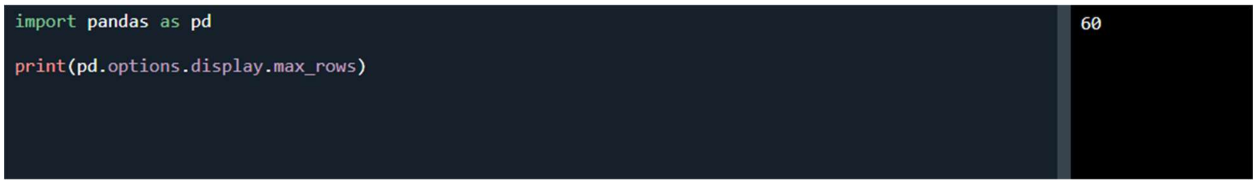
```python
import pandas as pd

df = pd.read_csv('data.csv')

print(df)
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.4
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
```

max_rows

The number of rows returned is defined in Pandas option settings.

You can check your system's maximum rows with the pd.options.display.max_rows statement.



```python
import pandas as pd

print(pd.options.display.max_rows)
```

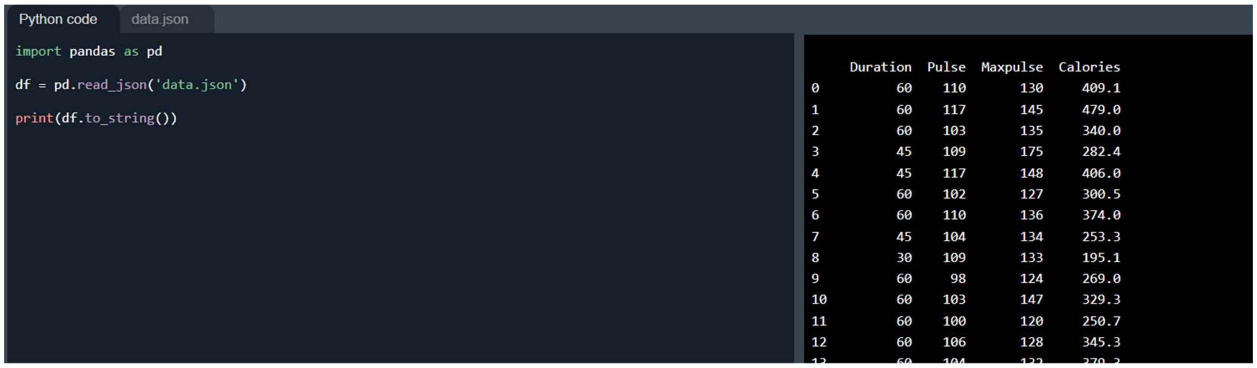
```
60
```

Read JSON

Big data sets are often stored, or extracted as JSON.

JSON is plain text, but has the format of an object, and is well known in the world of programming, including Pandas.

In our examples we will be using a JSON file called 'data.json'.

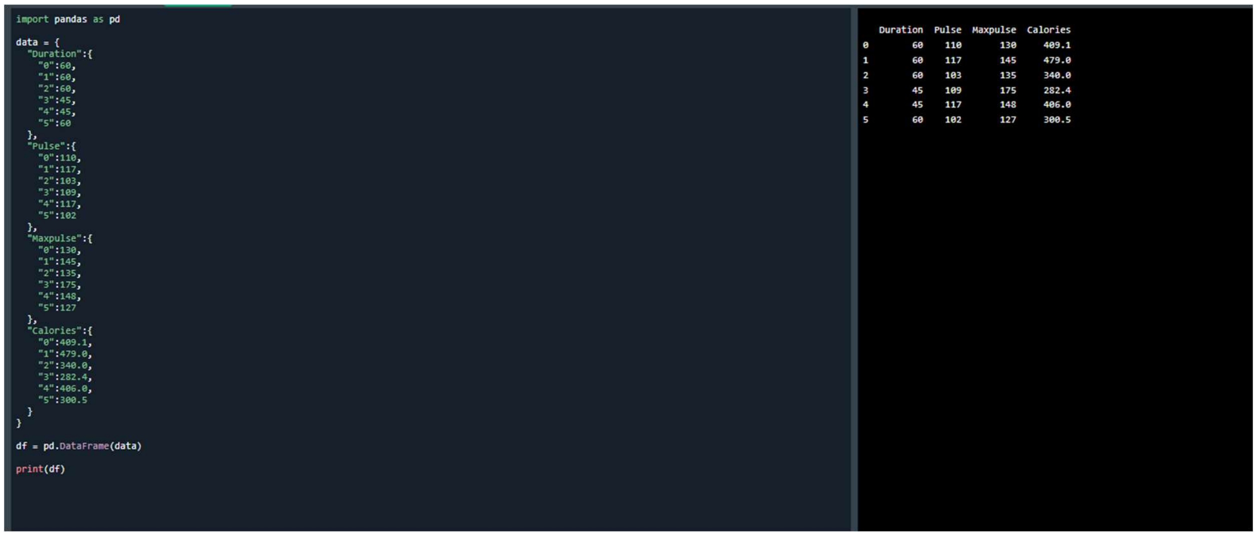


```python
import pandas as pd

df = pd.read_json('data.json')

print(df.to_string())
```

```
    Duration  Pulse  Maxpulse  Calories
0         60    110       130     409.1
1         60    117       145     479.0
2         60    103       135     340.0
3         45    109       175     282.4
4         45    117       148     406.0
5         60    102       127     300.5
6         60    110       136     374.0
7         45    104       134     253.3
8         30    109       133     195.1
9         60     98       124     269.0
10        60    103       147     329.3
11        60    100       120     250.7
12        60    106       128     345.3
```

Dict to JSON:

Pandas



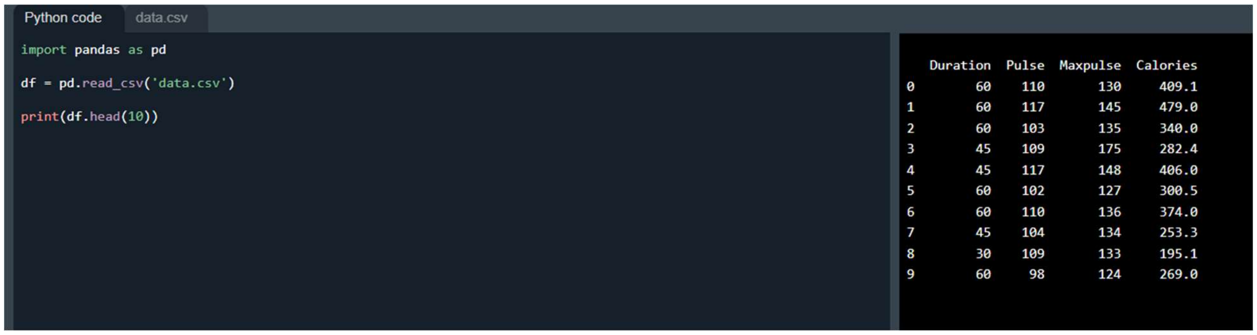
```
import pandas as pd

data = {
  "Duration":{
    "0":60,
    "1":60,
    "2":60,
    "3":45,
    "4":45,
    "5":60
  },
  "Pulse":{
    "0":110,
    "1":117,
    "2":103,
    "3":109,
    "4":117,
    "5":102
  },
  "Maxpulse":{
    "0":130,
    "1":145,
    "2":135,
    "3":175,
    "4":148,
    "5":127
  },
  "Calories":{
    "0":409.1,
    "1":479.0,
    "2":340.0,
    "3":282.4,
    "4":406.0,
    "5":300.5
  }
}

df = pd.DataFrame(data)

print(df)
```

```
   Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
2        60    103       135     340.0
3        45    109       175     282.4
4        45    117       148     406.0
5        60    102       127     300.5
```

Viewing the Data

One of the most used method for getting a quick overview of the DataFrame, is the head() method.

The head() method returns the headers and a specified number of rows, starting from the top.



```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.head(10))
```

```
   Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
2        60    103       135     340.0
3        45    109       175     282.4
4        45    117       148     406.0
5        60    102       127     300.5
6        60    110       136     374.0
7        45    104       134     253.3
8        30    109       133     195.1
9        60     98       124     269.0
```

Print the first 5 rows of the DataFrame:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.head())
```

Try it Yourself »

There is also a tail() method for viewing the *last* rows of the DataFrame.

The tail() method returns the headers and a specified number of rows, starting from the bottom.

Example

Pandas

Print the last 5 rows of the DataFrame:

```
print(df.tail())
```

Thank you