Pandas Read CSV

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'.

Example

Load the CSV into a DataFrame:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.to_string())
```

Note-use to_string() to print the entire DataFrame.

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

Example

```
#Print the DataFrame without the to_string() method:

import pandas as pd


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


print(df)
```

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.

Example

```
#Check the number of maximum returned rows:

import pandas as pd

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

In my system the number is 60, which means that if the DataFrame contains more than 60 rows, the print(df) statement will return only the headers and the first and last 5 rows.

You can change the maximum rows number with the same statement.

Example

Increase the maximum number of rows to display the entire DataFrame:

```
import pandas as pd

pd.options.display.max_rows = 9999

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

print(df)
```

Pandas Read JSON

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'.

Example

```
#Load the JSON file into a DataFrame:

import pandas as pd

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

print(df.to_string())
```

Dictionary as JSON

JSON = Python Dictionary

JSON objects have the same format as Python dictionaries.

If your JSON code is not in a file, but in a Python Dictionary, you can load it into a DataFrame directly:

Example

Load a Python Dictionary into a DataFrame:

```
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":479, "2":340,  "3":282, "4":406, "5":300}}

df = pd.DataFrame(data)

print(df)
```

Pandas - Analyzing DataFrames

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.

Example

Get a quick overview by printing the first 10 rows of the DataFrame:

```
import pandas as pd

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

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

Note: if the number of rows is not specified, the head() method will return the top 5 rows.

Example

Print the first 5 rows of the DataFrame:

```
import pandas as pd

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

print(df.head())
```

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

Print the last 5 rows of the DataFrame:

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

Info About the Data

The DataFrames object has a method called info(), that gives you more information about the data set.

Example

Print information about the data:

```
print(df.info())
```

Null Values

The info() method also tells us how many Non-Null values there are present in each column, and in our data set it seems like there are 164 of 169 Non-Null values in the "Calories" column.