Pandas tutorials

What is Pandas?

Pandas is one of the most important libraries of Python. Pandas has data structures for data analysis. The most used of these are Series and DataFrame data structures. Series is one dimensional, that is, it consists of a column. Data frame is two-dimensional, i.e. it consists of rows and columns.

To install Pandas, you can use "pip install pandas"

Jupyter for beginners

Links: https://daily.dev/blog/jupyter-for-beginners#:~:text=Here%27s%20a%20quick%20guide%20to%20get%20you%20started%3A,Dive%20into%20writing%20and%20running%20code.%20More%20items

Pandas First Steps

Install and import

Pandas is an easy package to install. Open up your terminal program (for Mac users) or command line (for PC users) and install it using either of the following commands:

conda install pandas

OR

pip install pandas

Alternatively, if you're currently viewing this article in a Jupyter notebook you can run this cell:

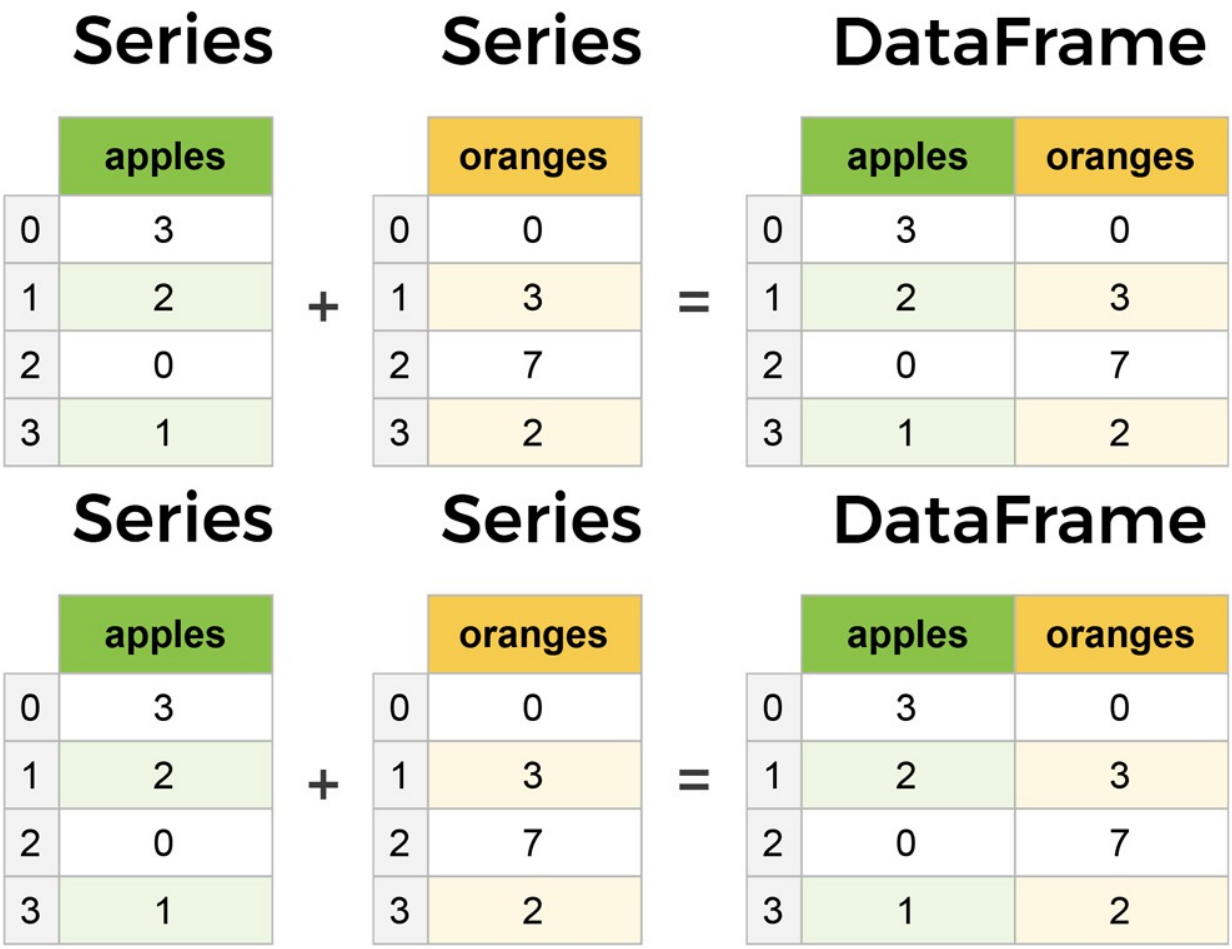
!pip install pandas

Now to the basic components of pandas

Core components of pandas: Series and DataFrames

The primary two components of pandas are the Series and DataFrame.

A Series is essentially a column, and a DataFrame is a multi-dimensional table made up of a collection of Series.



Installation of Pandas

```
!pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\rmnjs\appdata\local\
programs\python\python39\lib\site-packages (2.1.1)
Requirement already satisfied: numpy>=1.22.4 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from pandas)
(1.26.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
rmnjs\appdata\local\programs\python\python39\lib\site-packages (from
pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\rmnjs\appdata\
local\programs\python\python39\lib\site-packages (from pandas)
(2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in c:\users\rmnjs\
appdata\local\programs\python\python39\lib\site-packages (from pandas)
(2023.3)
Requirement already satisfied: six>=1.5 in c:\users\rmnjs\appdata\
```

```
local\programs\python\python39\lib\site-packages (from python-
dateutil>=2.8.2->pandas) (1.16.0)


[notice] A new release of pip is available: 23.3.1 -> 24.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Upgrade the pip package manager

```
pip install --upgrade pip

Requirement already satisfied: pip in c:\users\rmnjs\appdata\local\
programs\python\python39\lib\site-packages (23.3.1)
Collecting pip
  Downloading pip-24.2-py3-none-any.whl.metadata (3.6 kB)
Downloading pip-24.2-py3-none-any.whl (1.8 MB)
   ---------------------------------------- 0.0/1.8 MB ? eta -:--:--
   ------------- -------------------------- 0.6/1.8 MB 13.8 MB/s eta
0:00:01
   ---------------------------------------- 1.8/1.8 MB 23.1 MB/s eta
0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.3.1
    Uninstalling pip-23.3.1:
      Successfully uninstalled pip-23.3.1
Successfully installed pip-24.2
Note: you may need to restart the kernel to use updated packages.
```

Import Pandas:

Once Pandas is installed, import it in your applications by adding the import keyword:

```
import pandas
```

Example

```
mydataset = {
             'cars': ["BMW", "Volvo", "Ford"],
            'passings': [3,7,2]
            }
myvar = pandas.DataFrame(mydataset)
print(myvar)
```

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

Pandas as pd:

Pandas is usually imported under the pd alias.

alias: In Python alias are an alternate name for referring to the same thing.

Create an alias with the as keyword while importing:

```
import pandas as pd
```

Now the Pandas package can be referred to as pd instead of pandas.

Example

```
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
```

Checking Pandas Version:

The version string is stored under **version** attribute.

```
#Example
import pandas as pd
```

```
print(pd.__version__)
```

```
2.1.1
```

What is a Series?

- A Pandas Series is like a column in a table.

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

Example-

Create a simple Pandas Series from a list:

```
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.

```
# Example
# Return the first value of the Series:

print(myvar[0])
```

```
1
```

Create Labels:

With the index argument, you can name your own labels.

```
# Example
# Create your own labels:

import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["A", "B", "C"])

print(myvar)
```

```
A    1
B    7
C    2
dtype: int64
```

When you have created labels, you can access an item by referring to the label.

Example:

Return the value of "y":

```
print(myvar["B"])
```

```
7
```

Key/Value Objects as Series:

You can also use a key/value object, like a dictionary, when creating a Series.

```
# Example-
# Create a simple Pandas Series from a dictionary:

import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

```
day1    420
day2    380
day3    390
dtype: int64
```

Note: The keys of the dictionary become the labels.

To select only some of the items in the dictionary, use the index argument and specify only the items you want to include in the Series.

```
#Example-
# Create a Series using only data from "day1" and "day2":

import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories, index = ["day1", "day2"])

print(myvar)
```

```
day1    420
day2    380
dtype: int64
```

DataFrames:

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.

- Data sets in Pandas are usually multi-dimensional tables, called DataFrames.

- Series is like a column, a DataFrame is the whole table.

Example-

Create a DataFrame from two Series:

```
import pandas as pd
```

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

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

print(myvar)
```

```
   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)

Example-

Return row 0:

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

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

Note: This example returns a Pandas Series.

Example-

Return row 0 and 1:

```
#use a list of indexes:
print(myvar.loc[[0, 1]])
```

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

Note: When using list-[], the result is a Pandas DataFrame.

Named Indexes:

With the index argument, you can name your own indexes.

```
import pandas as pd

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

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

```
      calories  duration
day1       420        50
day2       380        40
day3       390        45
```

Locate Named Indexes:

Use the named index in the loc attribute to return the specified row(s).

Example-

Return "day2":

```
#refer to the named index:
print(df.loc["day2"])
```

```
calories    380
duration     40
Name: day2, dtype: int64
```

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'.

Link:
https://drive.google.com/file/d/1oeJY42VtjD91oBQ8GmK50UZbKap8q2DK/view?usp=sharing

```
# Example
# Load the CSV into a DataFrame:

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     300.0
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
17         45     90       112       NaN
18         60    103       123     323.0
19         45     97       125     243.0
20         60    108       131     364.2
```

21	45	100	119	282.0
22	60	130	101	300.0
23	45	105	132	246.0
24	60	102	126	334.5
25	60	100	120	250.0
26	60	92	118	241.0
27	60	103	132	NaN
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1
32	60	101	124	299.0
33	60	93	113	223.0
34	60	107	136	361.0
35	60	114	140	415.0
36	60	102	127	300.0
37	60	100	120	300.0
38	60	100	120	300.0
39	45	104	129	266.0
40	45	90	112	180.1
41	60	98	126	286.0
42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0
48	45	108	129	298.0
49	60	111	139	397.6
50	60	107	136	380.2
51	80	123	146	643.1
52	60	106	130	263.0
53	60	118	151	486.0
54	30	136	175	238.0
55	60	121	146	450.7
56	60	118	121	413.0
57	45	115	144	305.0
58	20	153	172	226.4
59	45	123	152	321.0
60	210	108	160	1376.0
61	160	110	137	1034.4
62	160	109	135	853.0
63	45	118	141	341.0
64	20	110	130	131.4
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
68	20	106	136	110.4
69	300	108	143	1500.2

70	150	97	129	1115.0
71	60	109	153	387.6
72	90	100	127	700.0
73	150	97	127	953.2
74	45	114	146	304.0
75	90	98	125	563.2
76	45	105	134	251.0
77	45	110	141	300.0
78	120	100	130	500.4
79	270	100	131	1729.0
80	30	159	182	319.2
81	45	149	169	344.0
82	30	103	139	151.1
83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.0
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3
90	180	101	127	600.1
91	45	107	137	NaN
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4
96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	180	90	120	800.3
107	30	90	120	86.2
108	90	90	120	500.3
109	210	137	184	1860.4
110	60	102	124	325.2
111	45	107	124	275.0
112	15	124	139	124.2
113	45	100	120	225.3
114	60	108	131	367.6
115	60	108	151	351.7
116	60	116	141	443.0
117	60	97	122	277.4
118	60	105	125	NaN

119	60	103	124	332.7
120	30	112	137	193.9
121	45	100	120	100.7
122	60	119	169	336.7
123	60	107	127	344.9
124	60	111	151	368.5
125	60	98	122	271.0
126	60	97	124	275.3
127	60	109	127	382.0
128	90	99	125	466.4
129	60	114	151	384.0
130	60	104	134	342.5
131	60	107	138	357.5
132	60	103	133	335.0
133	60	106	132	327.5
134	60	103	136	339.0
135	20	136	156	189.0
136	45	117	143	317.7
137	45	115	137	318.0
138	45	113	138	308.0
139	20	141	162	222.4
140	60	108	135	390.0
141	60	97	127	NaN
142	45	100	120	250.4
143	45	122	149	335.4
144	60	136	170	470.2
145	45	106	126	270.8
146	60	107	136	400.0
147	60	112	146	361.9
148	30	103	127	185.0
149	60	110	150	409.4
150	60	106	134	343.0
151	60	109	129	353.2
152	60	109	138	374.0
153	30	150	167	275.8
154	60	105	128	328.0
155	60	111	151	368.5
156	60	97	131	270.4
157	60	100	120	270.4
158	60	114	150	382.8
159	30	80	120	240.9
160	30	85	120	250.4
161	45	90	130	260.4
162	45	95	130	270.0
163	45	100	140	280.9
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2

```
167         75    120         150      320.4
168         75    125         150      330.4
```

Tip: 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)
```

```
     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     300.0
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.

Example-

Check the number of maximum returned rows:

```
import pandas as pd

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

```
60
```

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)
```

```
     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     300.0
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
17         45     90       112       NaN
18         60    103       123     323.0
19         45     97       125     243.0
20         60    108       131     364.2
21         45    100       119     282.0
22         60    130       101     300.0
23         45    105       132     246.0
24         60    102       126     334.5
25         60    100       120     250.0
```

26	60	92	118	241.0
27	60	103	132	NaN
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1
32	60	101	124	299.0
33	60	93	113	223.0
34	60	107	136	361.0
35	60	114	140	415.0
36	60	102	127	300.0
37	60	100	120	300.0
38	60	100	120	300.0
39	45	104	129	266.0
40	45	90	112	180.1
41	60	98	126	286.0
42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0
48	45	108	129	298.0
49	60	111	139	397.6
50	60	107	136	380.2
51	80	123	146	643.1
52	60	106	130	263.0
53	60	118	151	486.0
54	30	136	175	238.0
55	60	121	146	450.7
56	60	118	121	413.0
57	45	115	144	305.0
58	20	153	172	226.4
59	45	123	152	321.0
60	210	108	160	1376.0
61	160	110	137	1034.4
62	160	109	135	853.0
63	45	118	141	341.0
64	20	110	130	131.4
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
68	20	106	136	110.4
69	300	108	143	1500.2
70	150	97	129	1115.0
71	60	109	153	387.6
72	90	100	127	700.0
73	150	97	127	953.2
74	45	114	146	304.0

75	90	98	125	563.2
76	45	105	134	251.0
77	45	110	141	300.0
78	120	100	130	500.4
79	270	100	131	1729.0
80	30	159	182	319.2
81	45	149	169	344.0
82	30	103	139	151.1
83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.0
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3
90	180	101	127	600.1
91	45	107	137	NaN
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4
96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	180	90	120	800.3
107	30	90	120	86.2
108	90	90	120	500.3
109	210	137	184	1860.4
110	60	102	124	325.2
111	45	107	124	275.0
112	15	124	139	124.2
113	45	100	120	225.3
114	60	108	131	367.6
115	60	108	151	351.7
116	60	116	141	443.0
117	60	97	122	277.4
118	60	105	125	NaN
119	60	103	124	332.7
120	30	112	137	193.9
121	45	100	120	100.7
122	60	119	169	336.7
123	60	107	127	344.9

124	60	111	151	368.5
125	60	98	122	271.0
126	60	97	124	275.3
127	60	109	127	382.0
128	90	99	125	466.4
129	60	114	151	384.0
130	60	104	134	342.5
131	60	107	138	357.5
132	60	103	133	335.0
133	60	106	132	327.5
134	60	103	136	339.0
135	20	136	156	189.0
136	45	117	143	317.7
137	45	115	137	318.0
138	45	113	138	308.0
139	20	141	162	222.4
140	60	108	135	390.0
141	60	97	127	NaN
142	45	100	120	250.4
143	45	122	149	335.4
144	60	136	170	470.2
145	45	106	126	270.8
146	60	107	136	400.0
147	60	112	146	361.9
148	30	103	127	185.0
149	60	110	150	409.4
150	60	106	134	343.0
151	60	109	129	353.2
152	60	109	138	374.0
153	30	150	167	275.8
154	60	105	128	328.0
155	60	111	151	368.5
156	60	97	131	270.4
157	60	100	120	270.4
158	60	114	150	382.8
159	30	80	120	240.9
160	30	85	120	250.4
161	45	90	130	260.4
162	45	95	130	270.0
163	45	100	140	280.9
164	60	105	140	290.8
165	60	110	145	300.0
166	60	115	145	310.2
167	75	120	150	320.4
168	75	125	150	330.4

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 'sample.json'.