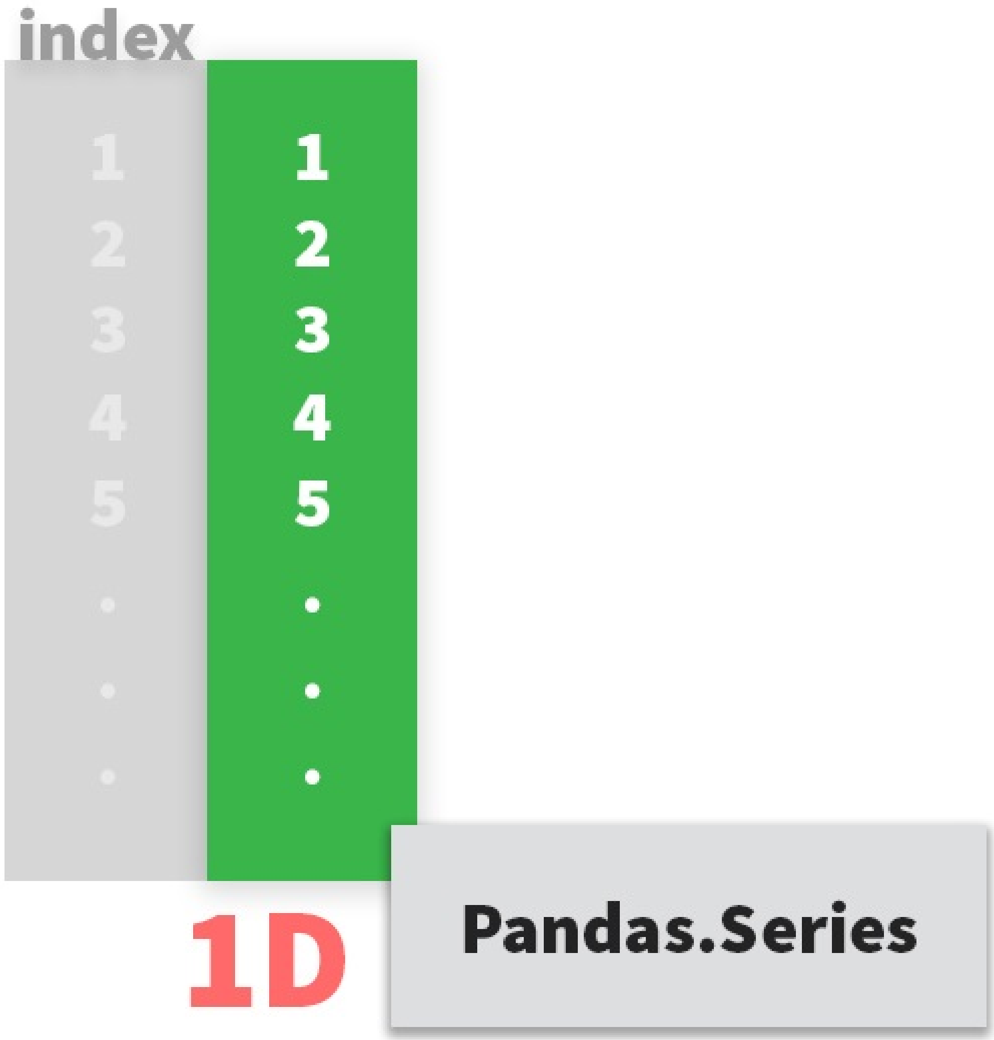
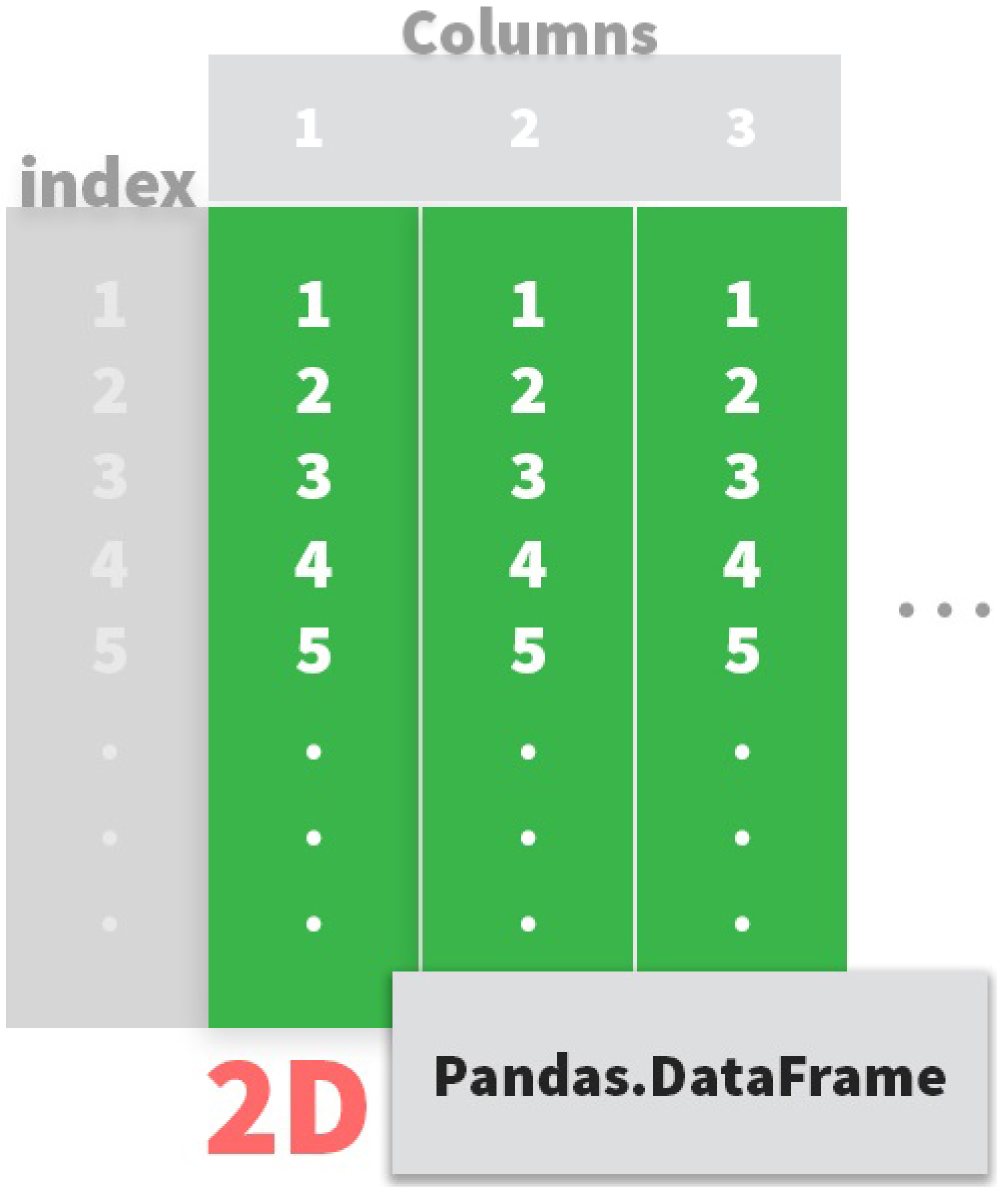
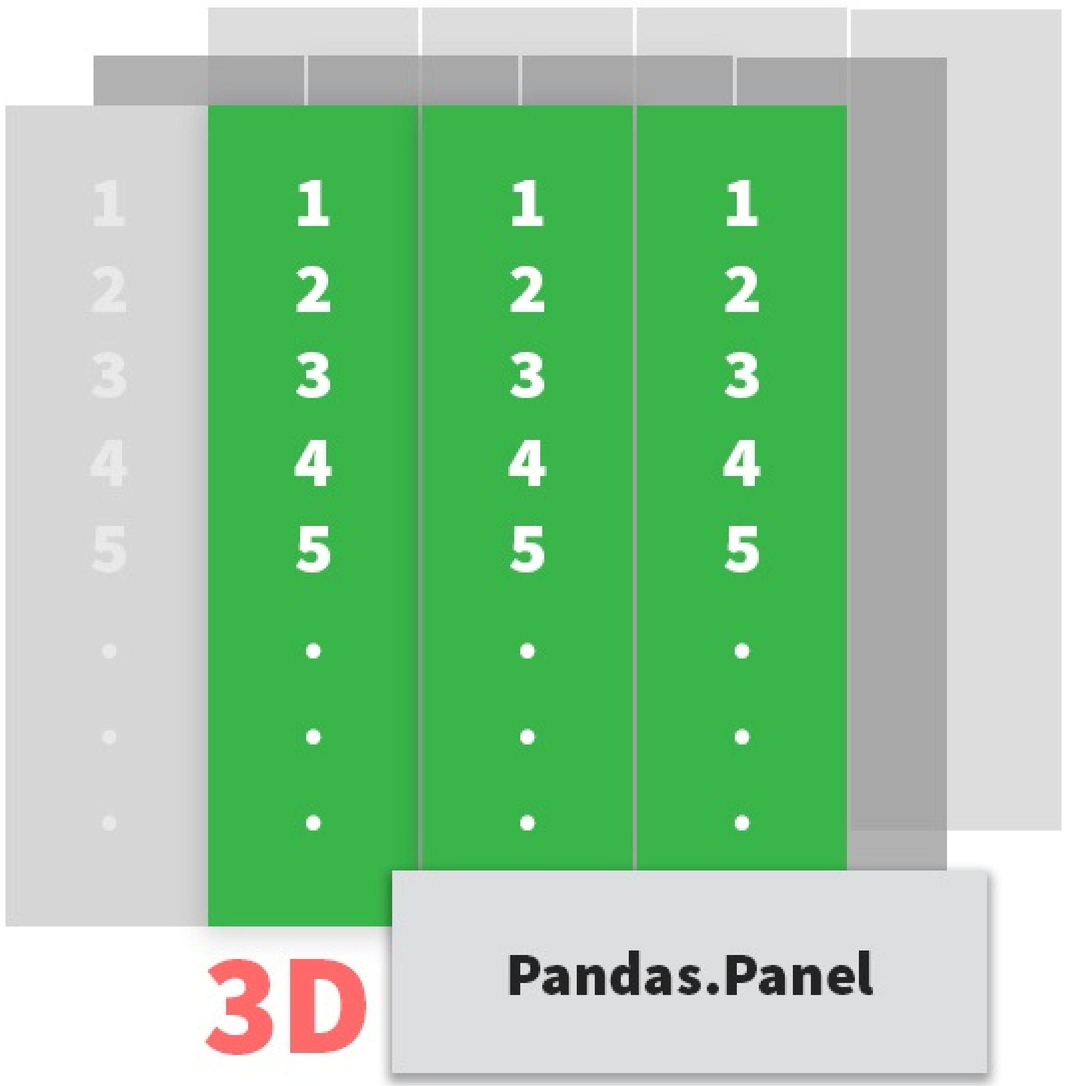
Pandas

> pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.




Columns
1
2
3
index
1
2
3
4
5
2D
Pandas.DataFrame



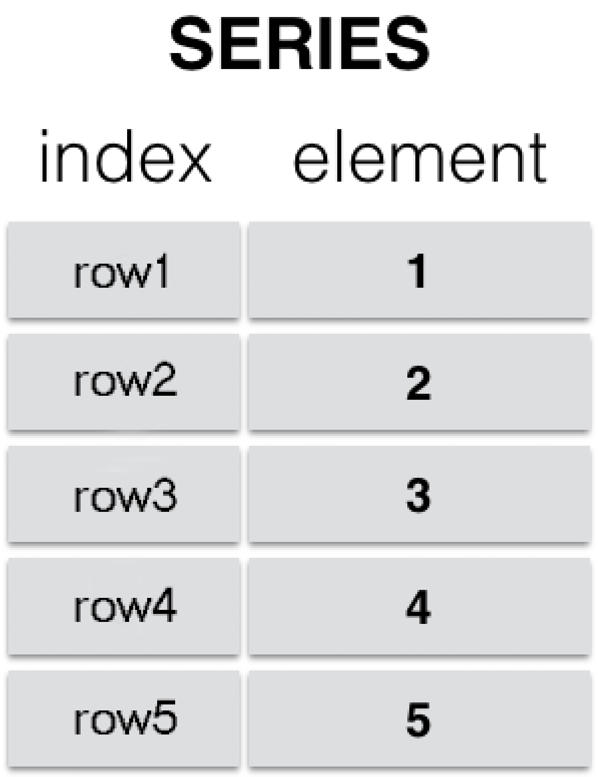
Data Structure

> **Pandas docs :** In 0.20.0, Panel is deprecated and will be removed in a future version. The 3-D structure of a Panel is much less common for many types of data analysis, than the 1-D of the Series or the 2-D of the DataFrame.

1 - Series

`pandas.Series()`

> **Pandas docs :** One-dimensional ndarray with axis labels (including time series).



SERIES

index	element
row1	1
row2	2
row3	3
row4	4
row5	5

In [3]:

```
import pandas as pd
my_series = pd.Series([1, 2, 3,4,5],index=['row1','row2','row3','row4','row5'])
my_series
```

Out[3]:

```
row1    1
row2    2
row3    3
row4    4
row5    5
dtype: int64
```

Show Values

In [4]:

```
my_series.values
```

Out[4]:

```
array([1, 2, 3, 4, 5], dtype=int64)
```

Show index

In [5]:

```
my_series.index
```

Out[5]:

```
Index(['row1', 'row2', 'row3', 'row4', 'row5'], dtype='object')
```

Select index

In [6]:

```
my_series.row2
```

Out[6]:

```
2
```

In [7]:

```
my_series['row2']
```

Out[7]:

```
2
```

Boolean indexing

In [8]:

```
my_series[my_series>3]
```

Out[8]:

```
row4    4
row5    5
dtype: int64
```

Example : Set alphabet label as new index

In [9]:

```
my_series.index =  ['A','B','C','D','E']
my_series
```

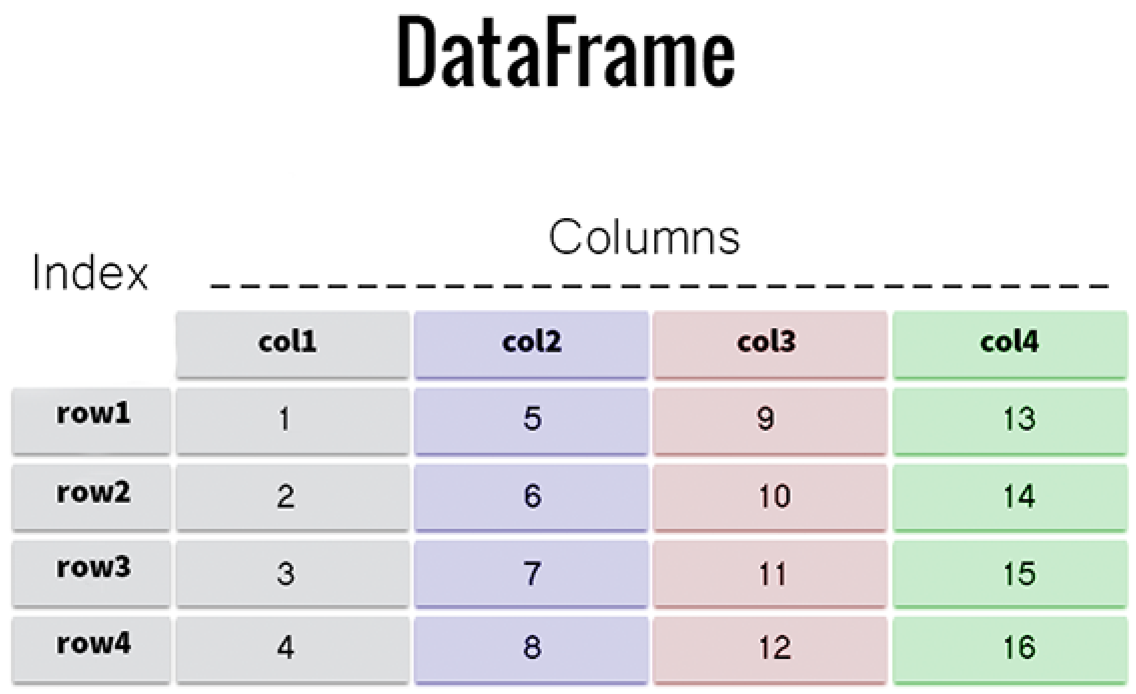
Out[9]:

```
A    1
B    2
C    3
D    4
E    5
dtype: int64
```

2 - DataFrame

`pandas.DataFrame()`

> **Pandas docs :** Two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure



Create DataFrame with Array

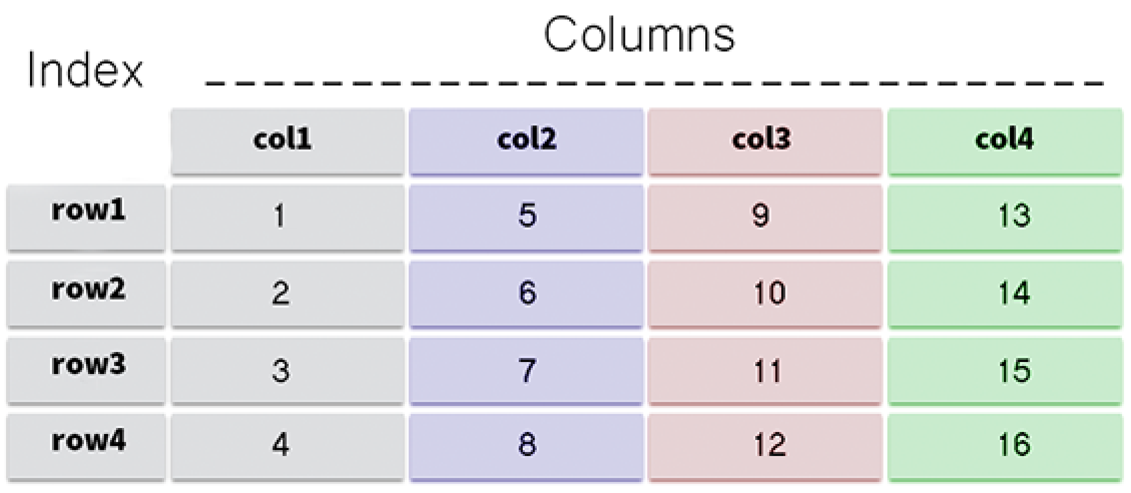
In [200]:

```
import numpy as np
my_array = np.array([[1 ,5 ,9 ,13],[2 ,6 ,10 ,14],[3 ,7 ,11 ,15],[4 ,8 ,12 ,16]])
my_df = pd.DataFrame(my_array,index=['row1' ,'row2' ,'row3' ,'row4'],columns=['col1' ,'col2' ,'c
ol3' ,'col4'])
my_df
```

Out[200]:

	col1	col2	col3	col4
row1	1	5	9	13
row2	2	6	10	14
row3	3	7	11	15
row4	4	8	12	16

DataFrame


Columns
Index

	col1	col2	col3	col4
row1	1	5	9	13
row2	2	6	10	14
row3	3	7	11	15
row4	4	8	12	16

Create DataFrame with Dictionary

In [36]:

```
my_dict = {'col1':[1,2,3,4],'col2':[5,6,7,8],'col3':[9,10,11,12],'col4':[13,14,15,19]}
my_df = pd.DataFrame(my_dict, index=['row1','row2','row3','row4'])
my_df
```

Out[36]:

	col1	col2	col3	col4
row1	1	5	9	13
row2	2	6	10	14
row3	3	7	11	15
row4	4	8	12	19

Show index

In [37]:

```
my_df.index
```

Out[37]:

```
Index(['row1', 'row2', 'row3', 'row4'], dtype='object')
```

Show Columns

In [38]:

```
my_df.columns
```

Out[38]:

```
Index(['col1', 'col2', 'col3', 'col4'], dtype='object')
```

Show Value

In [39]:

```
my_df.values
```

Out[39]:

```
array([[ 1,  5,  9, 13],
       [ 2,  6, 10, 14],
       [ 3,  7, 11, 15],
       [ 4,  8, 12, 19]], dtype=int64)
```

Selecting

In [40]:

```
my_df
```

Out[40]:

	col1	**col2**	**col3**	**col4**
row1	1	5	9	13
row2	2	6	10	14
row3	3	7	11	15
row4	4	8	12	19

In [41]:

```
my_df.loc['row1'][:]
```

Out[41]:

```
col1     1
col2     5
col3     9
col4    13
Name: row1, dtype: int64
```

In [42]:

```
my_df.iloc[0][:]
```

Out[42]:

```
col1     1
col2     5
col3     9
col4    13
Name: row1, dtype: int64
```

Edit a DataFrame

In [44]:

```
my_df['col5'] = [20 ,21 ,22 ,23]
my_df
```

Out[44]:

	col1	col2	col3	col4	col5
row1	1	5	9	13	20
row2	0	6	10	14	21
row3	3	7	11	15	22
row4	4	8	12	19	23

In [47]:

```
my_df.loc[['row1','row2'],'col1'] = 0
my_df
```

Out[47]:

	col1	col2	col3	col4	col5
row1	0	5	9	13	20
row2	0	6	10	14	21
row3	3	7	11	15	22
row4	4	8	12	19	23

Reset index

In [48]:

```
my_df.reset_index(drop=True)
```

Out[48]:

	col1	col2	col3	col4	col5
0	0	5	9	13	20
1	0	6	10	14	21
2	3	7	11	15	22
3	4	8	12	19	23

Deleting

In [49]:

```
my_df.drop('col5',axis=1)
```

Out[49]:

	col1	col2	col3	col4
row1	0	5	9	13
row2	0	6	10	14
row3	3	7	11	15
row4	4	8	12	19

Renaming

In [50]:

```
my_df.rename(columns={'col4':'col_four'})
```

Out[50]:

	col1	col2	col3	col_four	col5
row1	0	5	9	13	20
row2	0	6	10	14	21
row3	3	7	11	15	22
row4	4	8	12	19	23

Replacing

In [51]:

```
my_df.replace({0:1},regex=True)
```

Out[51]:

	col1	col2	col3	col4	col5
row1	1	5	9	13	20
row2	1	6	10	14	21
row3	3	7	11	15	22
row4	4	8	12	19	23

Apply function on index

In [214]:

```
my_df.col1 = ['{:3.2f}'.format(x) for x in my_df.iloc[:,0] ]
my_df
```

Out[214]:

	col1	col2	col3	col4	col5
row1	0.00	5	9	13	20
row2	2.00	6	10	14	21
row3	3.00	7	11	15	22
row4	4.00	8	12	19	23

In [215]:

```
my_df['col2'] = my_df['col2'].apply(lambda x:'{0:3.2f}'.format(x))
my_df
```

Out[215]:

	col1	col2	col3	col4	col5
row1	0.00	5.00	9	13	20
row2	2.00	6.00	10	14	21
row3	3.00	7.00	11	15	22
row4	4.00	8.00	12	19	23

Sorting

- sort index

In [216]:

```
my_df.sort_index(axis=1,ascending=False)
```

Out[216]:

	col5	col4	col3	col2	col1
row1	20	13	9	5.00	0.00
row2	21	14	10	6.00	2.00
row3	22	15	11	7.00	3.00
row4	23	19	12	8.00	4.00

- sort values

In [217]:

```
my_df.sort_values(by='col1',ascending=False)
```

Out[217]:

	col1	col2	col3	col4	col5
row4	4.00	8.00	12	19	23
row3	3.00	7.00	11	15	22
row2	2.00	6.00	10	14	21
row1	0.00	5.00	9	13	20

Methods

In [218]:

```
my_df.head()
```

Out[218]:

	col1	col2	col3	col4	col5
row1	0.00	5.00	9	13	20
row2	2.00	6.00	10	14	21
row3	3.00	7.00	11	15	22
row4	4.00	8.00	12	19	23