

✓ 1) Pandas Series

✓ Create a Series

```
import numpy as np
import pandas as pd
```

```
# 1) creating pandas series from a list
```

```
my_data = [10,20,30,40,50]
labels = ['a','b','c','d','e']
```

```
pd.Series(data=my_data)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

```
pd.Series(data=my_data, index=labels)
```

```
a    10
b    20
c    30
d    40
e    50
dtype: int64
```

```
pd.Series(my_data, labels)
```

```
a    10
b    20
c    30
d    40
e    50
dtype: int64
```

```
# 2) Pandas series using numpy array
```

```
arr = np.array(my_data)
arr
```

```
array([10, 20, 30, 40, 50])
```

```
pd.Series(data=arr)
```

```
0    10
1    20
2    30
3    40
4    50
dtype: int64
```

```
pd.Series(data=arr,index=labels)
```

```
a    10
b    20
c    30
d    40
e    50
dtype: int64
```

```
# 3) pandas series using a dictionary
```

```
d = {'a':100, 'b':200, 'c':300, 'd':400}
```

```
d
```

```
{'a': 100, 'b': 200, 'c': 300, 'd': 400}
```

```
pd.Series(d)
```

```
a    100
b    200
c    300
d    400
dtype: int64
```

```
pd.Series(data=labels)
```

```
0    a
1    b
2    c
3    d
4    e
dtype: object
```

```
pd.Series(data=[sum, len])
```

```
0    <built-in function sum>
1    <built-in function len>
dtype: object
```

▼ Grab Info from Pandas Series

```
ser1 = pd.Series(data=[1,2,3,4], index=['Red','Green','Blue','Orange'])
```

```
ser1
```

```
Red      1
Green     2
Blue      3
Orange    4
dtype: int64
```

```
ser2 = pd.Series(data=[1,2,5,4], index=['Red','Green','Yellow','Orange'])
```

```
ser2
```

```
Red      1
Green     2
Yellow    5
Orange    4
dtype: int64
```

```
ser2['Green']
```

```
2
```

```
ser1['Blue']
```

```
3
```

```
ser3 = pd.Series(data=['a','b','c','d','e'])
```

```
ser3
```

```
0    a
1    b
2    c
3    d
4    e
dtype: object
```

```
ser3[1]
```

```
'b'
```

```
ser3[0:3]
```

```
0    a
1    b
2    c
dtype: object
```

```
ser1
Red      1
Green    2
Blue     3
Orange   4
dtype: int64
```

```
ser2
Red      1
Green    2
Yellow   5
Orange   4
dtype: int64
```

```
ser1 + ser2
Blue     NaN
Green    4.0
Orange   8.0
Red      2.0
Yellow   NaN
dtype: float64
```

2) DataFrames

```
import numpy as np
import pandas as pd
from numpy.random import randn
np.random.seed(101)

df = pd.DataFrame(data=randn(5,4), index=['a','b','c','d','e'], columns=['w','x','y','z'])
```

df

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

1) Indexing and Selection

```
df['w']
a      2.706850
b      0.651118
c     -2.018168
d      0.188695
e      0.190794
Name: w, dtype: float64
```

```
type(df['w'])
pandas.core.series.Series
```

```
type(df)
pandas.core.frame.DataFrame
```

df

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df[['x', 'y']]
```

	x	y
a	0.628133	0.907969
b	-0.319318	-0.848077
c	0.740122	0.528813
d	-0.758872	-0.933237
e	1.978757	2.605967

```
type(df[['x', 'y']])

pandas.core.frame.DataFrame
```

```
# 2) add a new column
```

```
df
```

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df['new'] = df['w'] + df['x']
```

```
df
```

	w	x	y	z	new
a	2.706850	0.628133	0.907969	0.503826	3.334983
b	0.651118	-0.319318	-0.848077	0.605965	0.331800
c	-2.018168	0.740122	0.528813	-0.589001	-1.278046
d	0.188695	-0.758872	-0.933237	0.955057	-0.570177
e	0.190794	1.978757	2.605967	0.683509	2.169552

```
# 3) removing columns and rows
```

```
df.drop(labels='new',axis=1) #axis=1 is for columns and axis=0 is for rows.
```

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df.drop(labels='new',axis=1,inplace=True) #permanent deletion
```

df

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df.drop('b',axis=0) #axis=0 ---> rows
```

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
# 4) selecting the rows
```

df

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df.loc['c']  
  
w    -2.018168  
x     0.740122  
y     0.528813  
z    -0.589001  
Name: c, dtype: float64
```

```
type(df.loc['c'])  
  
pandas.core.series.Series
```

```
# selecting the subset of rows and columns
```

df

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df.loc['c','x']  
  
0.7401220570561068
```

```
df.loc[['d','e'],['y','z']]
```

	y	z
d	-0.933237	0.955057
e	2.605967	0.683509

```
import numpy as np
import pandas as pd
from numpy.random import randn
np.random.seed(101)

df=pd.DataFrame(randn(5,4),["a","b","c","d","e"],["w","x","y","z"])
```

df

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
# 1) conditional selection
```

df > 0

	w	x	y	z
a	True	True	True	True
b	True	False	False	True
c	False	True	True	False
d	True	False	False	True
e	True	True	True	True

```
b = df > 0
```

b

	w	x	y	z
a	True	True	True	True
b	True	False	False	True
c	False	True	True	False
d	True	False	False	True
e	True	True	True	True

df[b]

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	NaN	NaN	0.605965
c	NaN	0.740122	0.528813	NaN
d	0.188695	NaN	NaN	0.955057
e	0.190794	1.978757	2.605967	0.683509

df[df > 0]

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	NaN	NaN	0.605965
c	NaN	0.740122	0.528813	NaN
d	0.188695	NaN	NaN	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df['w'] > 0
```

a	True
b	True
c	False
d	True
e	True
Name: w, dtype: bool	

```
d = df['w'] > 0
```

d

a	True
b	True
c	False
d	True
e	True
Name: w, dtype: bool	

```
df[d]
```

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df[df['w'] > 0]
```

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
# 2) multiple conditions
```



df

	w	x	y	z
a	2.706850	0.628133	0.907969	0.503826
b	0.651118	-0.319318	-0.848077	0.605965
c	-2.018168	0.740122	0.528813	-0.589001
d	0.188695	-0.758872	-0.933237	0.955057
e	0.190794	1.978757	2.605967	0.683509

```
df[(df['w']>0) & (df['y']>1)] # and operator
```




	w	x	y	z
e	0.190794	1.978757	2.605967	0.683509

```
df[(df['w']>0) | (df['y']>1)] # or operator
```



	w	x	y	z	
a	2.706850	0.628133	0.907969	0.503826	
b	0.651118	-0.319318	-0.848077	0.605965	
d	0.188695	-0.758872	-0.933237	0.955057	
e	0.190794	1.978757	2.605967	0.683509	

3) index

df

	w	x	y	z	
a	2.706850	0.628133	0.907969	0.503826	
b	0.651118	-0.319318	-0.848077	0.605965	
c	-2.018168	0.740122	0.528813	-0.589001	
d	0.188695	-0.758872	-0.933237	0.955057	
e	0.190794	1.978757	2.605967	0.683509	

df.reset_index()

	index	w	x	y	z	
0	a	2.706850	0.628133	0.907969	0.503826	
1	b	0.651118	-0.319318	-0.848077	0.605965	
2	c	-2.018168	0.740122	0.528813	-0.589001	
3	d	0.188695	-0.758872	-0.933237	0.955057	
4	e	0.190794	1.978757	2.605967	0.683509	




new_index = ['R','G','B','O','Y']

new_index



['R', 'G', 'B', 'O', 'Y']

df['Colors'] = new_index

df

	w	x	y	z	Colors	
a	2.706850	0.628133	0.907969	0.503826	R	
b	0.651118	-0.319318	-0.848077	0.605965	G	
c	-2.018168	0.740122	0.528813	-0.589001	B	
d	0.188695	-0.758872	-0.933237	0.955057	O	
e	0.190794	1.978757	2.605967	0.683509	Y	

df.set_index('Colors')

	w	x	y	z	
Colors					
R	2.706850	0.628133	0.907969	0.503826	
G	0.651118	-0.319318	-0.848077	0.605965	
B	-2.018168	0.740122	0.528813	-0.589001	
O	0.188695	-0.758872	-0.933237	0.955057	
Y	0.190794	1.978757	2.605967	0.683509	

Multi-index and index hierarchy


```
import numpy as np
import pandas as pd
from numpy.random import randn
np.random.seed(101)
```

```
outside = ['Red','Red','Red','Green','Green','Green']
inside = [1,2,3,1,2,3]
```

```
hier_index = list(zip(outside,inside))
```

```
hier_index
```

```
[('Red', 1), ('Red', 2), ('Red', 3), ('Green', 1), ('Green', 2), ('Green', 3)]
```

```
multi_index = pd.MultiIndex.from_tuples(hier_index, names=['Colors','Numbers'])
```

```
multi_index
```

```
MultiIndex([( 'Red', 1),
              ( 'Red', 2),
              ( 'Red', 3),
              ('Green', 1),
              ('Green', 2),
              ('Green', 3)],
            names=['Colors', 'Numbers'])
```

```
df = pd.DataFrame(data=randn(6,2), index=multi_index, columns=['A','B'])
```

```
df
```

		A	B
Colors	Numbers		
Red	1	2.706850	0.628133
	2	0.907969	0.503826
	3	0.651118	-0.319318
Green	1	-0.848077	0.605965
	2	-2.018168	0.740122
	3	0.528813	-0.589001

```
df.loc['Red'].loc[1]['A']
```

```
2.706849839399938
```

```
df.loc['Green'].loc[2]['B']
```

```
0.7401220570561068
```

```
df.index.names
```

```
FrozenList(['Colors', 'Numbers'])
```

```
df.index.names = ['Col','Num']
```

```
df
```

		A	B
Col	Num		
Red	1	2.706850	0.628133
	2	0.907969	0.503826
	3	0.651118	-0.319318
Green	1	-0.848077	0.605965
	2	-2.018168	0.740122
	3	0.528813	-0.589001

```
# cross section
```

df

		A	B
Col	Num		
Red	1	2.706850	0.628133
	2	0.907969	0.503826
	3	0.651118	-0.319318
Green	1	-0.848077	0.605965
	2	-2.018168	0.740122
	3	0.528813	-0.589001

```
df.loc['Red']
```

	A	B
Num		
1	2.706850	0.628133
2	0.907969	0.503826
3	0.651118	-0.319318

```
df.xs('Red')
```

	A	B
Num		
1	2.706850	0.628133
2	0.907969	0.503826
3	0.651118	-0.319318

```
df.xs(1, level='Num')
```

	A	B
Col		
Red	2.706850	0.628133
Green	-0.848077	0.605965

Missing Data

```
import numpy as np
import pandas as pd

d = {'A':[1,2,np.nan], 'B':[3,np.nan,np.nan], 'C':[4,5,6]}

d

{'A': [1, 2, nan], 'B': [3, nan, nan], 'C': [4, 5, 6]}

df = pd.DataFrame(d)

df
```

	A	B	C
0	1.0	3.0	4
1	2.0	NaN	5
2	NaN	NaN	6

```
# dropna method
```

```
df.dropna()
```

	A	B	C
0	1.0	3.0	4

```
df.dropna(axis=1)
```

	C
0	4
1	5
2	6

```
df.dropna(thresh=2, axis=0)
```

	A	B	C
0	1.0	3.0	4
1	2.0	NaN	5

```
df.dropna(thresh=1, axis=1)
```

	A	B	C
0	1.0	3.0	4
1	2.0	NaN	5
2	NaN	NaN	6

```
# fillna method
```

```
df
```

	A	B	C
0	1.0	3.0	4
1	2.0	NaN	5
2	NaN	NaN	6

```
df.fillna(value='New Value')
```

	A	B	C
0	1.0	3.0	4
1	2.0	New Value	5
2	New Value	New Value	6

```
df['A'].fillna(value=df['A'].mean())
```

0	1.0
1	2.0
2	1.5

Name: A, dtype: float64

Groupby Method

```
import numpy as np
import pandas as pd
```

```
team_data = {'Company': ['Apple', 'Apple', 'FB', 'FB', 'Google', 'Google'],
             'Person': ['Mark', 'Tom', 'John', 'Sara', 'Mia', 'Emma'],
             'Sales': [200, 150, 350, 125, 260, 180]}
```

team_data

```
{'Company': ['Apple', 'Apple', 'FB', 'FB', 'Google', 'Google'],
 'Person': ['Mark', 'Tom', 'John', 'Sara', 'Mia', 'Emma'],
 'Sales': [200, 150, 350, 125, 260, 180]}
```

```
df = pd.DataFrame(team_data)
```

df

	Company	Person	Sales	
0	Apple	Mark	200	
1	Apple	Tom	150	
2	FB	John	350	
3	FB	Sara	125	
4	Google	Mia	260	
5	Google	Emma	180	

```
df.groupby(by='Company')
```

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7f7023e03cd0>

```
b = df.groupby(by='Company')
```

```
b.mean()
```

<ipython-input-123-781b9fa94bde>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.mean is deprecated. In a fut
b.mean()

	Sales	
Company		
Apple	175.0	
FB	237.5	
Google	220.0	

```
b.sum()
```

<ipython-input-124-5625bf30d0a3>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a fut
b.sum()

	Sales	
Company		
Apple	350	
FB	475	
Google	440	

```
b.std()
```

<ipython-input-125-be35c3b18507>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.std is deprecated. In a fut
b.std()

	Sales	
Company		
Apple	35.355339	
FB	159.099026	
Google	56.568542	

```
b.sum().loc['Apple']
```

<ipython-input-126-b28e268a74b0>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a fut
b.sum().loc['Apple']
Sales 350
Name: Apple, dtype: int64

```
df.groupby(by='Company').sum().loc['FB']

<ipython-input-127-b4cc53764e09>:1: FutureWarning: The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a fut
df.groupby(by='Company').sum().loc['FB']
Sales      475
Name: FB, dtype: int64
```

```
b.count()
```

	Person	Sales
Company		
Apple	2	2
FB	2	2
Google	2	2

```
b.max()
```

	Person	Sales
Company		
Apple	Tom	200
FB	Sara	350
Google	Mia	260

```
b.min()
```

	Person	Sales
Company		
Apple	Mark	150
FB	John	125
Google	Emma	180

```
# groupby method with describe function
```

```
df.groupby(by='Company').describe()
```

	Sales							
	count	mean	std	min	25%	50%	75%	max
Company								
Apple	2.0	175.0	35.355339	150.0	162.50	175.0	187.50	200.0
FB	2.0	237.5	159.099026	125.0	181.25	237.5	293.75	350.0
Google	2.0	220.0	56.568542	180.0	200.00	220.0	240.00	260.0

```
df.groupby(by='Company').describe().transpose()
```

	Company	Apple	FB	Google
Sales	count	2.000000	2.000000	2.000000
	mean	175.000000	237.500000	220.000000
	std	35.355339	159.099026	56.568542
	min	150.000000	125.000000	180.000000
	25%	162.500000	181.250000	200.000000
	50%	175.000000	237.500000	220.000000
	75%	187.500000	293.750000	240.000000
	max	200.000000	350.000000	260.000000

```
df.groupby(by='Company').describe().loc['FB']
```

Sales	count	2.000000
	mean	237.500000
	std	159.099026
	min	125.000000
	25%	181.250000
	50%	237.500000
	75%	293.750000
	max	350.000000

Name: FB, dtype: float64

✖ Merging, Joining and Concatenating

```
import pandas as pd
```



✖ DataFrames Concatenation

```
df1=pd.DataFrame({'A':['A0','A1','A2','A3'],
                  'B':['B0','B1','B2','B3'],
                  'C':['C0','C1','C2','C3'],
                  'D': ['D0','D1','D2','D3']},
                  index = [0, 1, 2, 3])

df2=pd.DataFrame({'A':['A4','A5','A6','A7'],
                  'B':['B4','B5','B6','B7'],
                  'C':['C4','C5','C6','C7'],
                  'D':['D4','D5','D6','D7']},
                  index = [4, 5, 6, 7])

df3=pd.DataFrame({'A':['A8','A9','A10','A11'],
                  'B':['B8','B9','B10','B11'],
                  'C':['C8','C9','C10','C11'],
                  'D':['D8','D9','D10','D11']},
                  index = [8, 9, 10, 11])
```




df1

	A	B	C	D	
0	A0	B0	C0	D0	
1	A1	B1	C1	D1	
2	A2	B2	C2	D2	
3	A3	B3	C3	D3	



df2

	A	B	C	D	
4	A4	B4	C4	D4	
5	A5	B5	C5	D5	
6	A6	B6	C6	D6	
7	A7	B7	C7	D7	



df3

	A	B	C	D	
8	A8	B8	C8	D8	
9	A9	B9	C9	D9	
10	A10	B10	C10	D10	
11	A11	B11	C11	D11	

```
pd.concat(objs=[df1,df2,df3])
```

	A	B	C	D	
0	A0	B0	C0	D0	
1	A1	B1	C1	D1	
2	A2	B2	C2	D2	
3	A3	B3	C3	D3	
4	A4	B4	C4	D4	
5	A5	B5	C5	D5	
6	A6	B6	C6	D6	
7	A7	B7	C7	D7	
8	A8	B8	C8	D8	
9	A9	B9	C9	D9	
10	A10	B10	C10	D10	
11	A11	B11	C11	D11	

```
pd.concat(objs=[df1,df2,df3],axis=1)
```




	A	B	C	D	A	B	C	D	A	B	C	D	
0	A0	B0	C0	D0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	A1	B1	C1	D1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	A2	B2	C2	D2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	A3	B3	C3	D3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	NaN	NaN	A4	B4	C4	D4	NaN	NaN	NaN	NaN	
5	NaN	NaN	NaN	NaN	A5	B5	C5	D5	NaN	NaN	NaN	NaN	
6	NaN	NaN	NaN	NaN	A6	B6	C6	D6	NaN	NaN	NaN	NaN	
7	NaN	NaN	NaN	NaN	A7	B7	C7	D7	NaN	NaN	NaN	NaN	
8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A8	B8	C8	D8	
9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A9	B9	C9	D9	
10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A10	B10	C10	D10	
11	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	A11	B11	C11	D11	

DataFrames Merging




```
df4=pd.DataFrame({'Key':['K0','K1','K2','K3'],
                  'A':['A0','A1','A2','A3'],
                  'B':['B0','B1','B2','B3']})

df5=pd.DataFrame({'Key':['K0','K1','K2','K3'],
                  'C':['C0','C1','C2','C3'],
                  'D':['D0','D1','D2','D3']})
```


df4

	Key	A	B	
0	K0	A0	B0	
1	K1	A1	B1	
2	K2	A2	B2	
3	K3	A3	B3	

df5

	Key	C	D	
0	K0	C0	D0	
1	K1	C1	D1	
2	K2	C2	D2	
3	K3	C3	D3	




```
pd.merge(left=df4, right=df5, on='Key')
```

	Key	A	B	C	D	
0	K0	A0	B0	C0	D0	
1	K1	A1	B1	C1	D1	
2	K2	A2	B2	C2	D2	
3	K3	A3	B3	C3	D3	




```
df6 = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                    'key2': ['K0', 'K1', 'K0', 'K1'],
                    'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3']})

df7 = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                    'key2': ['K0', 'K0', 'K0', 'K0'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']})
```



df6

	key1	key2	A	B	
0	K0	K0	A0	B0	
1	K0	K1	A1	B1	
2	K1	K0	A2	B2	
3	K2	K1	A3	B3	

df7

	key1	key2	C	D	
0	K0	K0	C0	D0	
1	K1	K0	C1	D1	
2	K1	K0	C2	D2	
3	K2	K0	C3	D3	

```
pd.merge(left=df6, right=df7, on=['key1', 'key2'])
```

	key1	key2	A	B	C	D	
0	K0	K0	A0	B0	C0	D0	
1	K1	K0	A2	B2	C1	D1	
2	K1	K0	A2	B2	C2	D2	

▼ DataFrames Join

```
df8 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    index= ['K0', 'K1', 'K2', 'K3']})

df9 = pd.DataFrame({'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3'],
                    index= ['K0', 'K1', 'K2', 'K3']})
```

df8

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2
K3	A3	B3

df9

	C	D
K0	C0	D0
K1	C1	D1
K2	C2	D2
K3	C3	D3

df8.join(df9)

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	C1	D1
K2	A2	B2	C2	D2
K3	A3	B3	C3	D3

#OPERATIONS

```
d = {'col1':[1,2,3,4],
      'col2':[45,55,65,45],
      'col3':['asd','jkl','qwe','xyz']}
```

```
df=pd.DataFrame(d)
df
```

	col1	col2	col3
0	1	45	asd
1	2	55	jkl
2	3	65	qwe
3	4	45	xyz

```
df['col2'].unique()
array([45, 55, 65])

len(df['col2'].unique())
3
```

```
df['col2'].nunique()
3
```

```
df['col2'].value_counts()
45    2
55    1
65    1
Name: col2, dtype: int64
```

#apply method

```
def times5(x):
    return x*5
```




```
times5(10)
```

50

```
df['col1'].sum()
```

10

```
df
```

	col1	col2	col3	
0	1	45	asd	
1	2	55	jkl	
2	3	65	qwe	
3	4	45	xyz	


```
df['col1'].apply(times5)
```

```
0      5
1     10
2     15
3     20
Name: col1, dtype: int64
```

```
df['col3'].apply(len)
```

```
0      3
1      3
2      3
3      3
Name: col3, dtype: int64
```

```
df.drop('col1',axis=1)
```

	col2	col3	
0	45	asd	
1	55	jkl	
2	65	qwe	
3	45	xyz	


```
df.columns
```

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

```
df.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

```
df.sort_values('col2')
```

	col1	col2	col3	
0	1	45	asd	
3	4	45	xyz	
1	2	55	jkl	