In [1]:

```
# import lib

import numpy as np
import pandas as pd
```

In [ ]:

```
# Join
# Join method is used to combine the columns of two potential diff - indexes

# syntax
# DataFrame.join()
```

In [2]:

```
data1 = pd.DataFrame({
                        'clo1':[10,20,30,40],
                        'col2':[50,60,70,80]
})
data1
```

Out[2]:

	clo1	col2
0	10	50
1	20	60
2	30	70
3	40	80

In [3]:

```
data2 = pd.DataFrame({
                        'clo3':[100,200,300,400],
                        'col4':[500,600,700,800]
})
data2
```

Out[3]:

	clo3	col4
0	100	500
1	200	600
2	300	700
3	400	800

In [4]:

```
display(data1,data2)
```

	clo1	col2
0	10	50
1	20	60
2	30	70
3	40	80

	clo3	col4
0	100	500
1	200	600
2	300	700
3	400	800

In [ ]:

```
# data1.join(data2)
# Signature: data1.join(other, on=None, how='left', lsuffix='', rsuffix='', sort=False)
```

In [5]:

```
data1.join(data2)
```

Out[5]:

	clo1	col2	clo3	col4
0	10	50	100	500
1	20	60	200	600
2	30	70	300	700
3	40	80	400	800

In [ ]:

```

```

In [ ]:

```
# Question if both dataset have same column name then ?
# we should use concat method
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

In [7]:

```
data3 = pd.DataFrame({
                        'clo1':[10,20,30,40],
                        'col2':[50,60,70,80],
},index=['a','b','c','d'])
data3
```

Out[7]:

	clo1	col2
a	10	50
b	20	60
c	30	70
d	40	80

In [8]:

```
data4 = pd.DataFrame({
                        'clo3':[100,200,300,400],
                        'col4':[500,600,700,800]
})
data4
```

Out[8]:

	clo3	col4
0	100	500
1	200	600
2	300	700
3	400	800

In [9]:

```
data3.join(data4)
```

Out[9]:

	clo1	col2	clo3	col4
a	10	50	NaN	NaN
b	20	60	NaN	NaN
c	30	70	NaN	NaN
d	40	80	NaN	NaN

In [10]:

```
data4.join(data3)
```

Out[10]:

	clo3	col4	clo1	col2
0	100	500	NaN	NaN
1	200	600	NaN	NaN
2	300	700	NaN	NaN
3	400	800	NaN	NaN

In [ ]:

```

```

In [ ]:

```

```

In [11]:

```
data5 = pd.DataFrame({
                        'clo1':[10,20,30,40],
                        'col2':[50,60,70,80],
},index=['a','b','c','d'])
data5
```

Out[11]:

	clo1	col2
a	10	50
b	20	60
c	30	70
d	40	80

In [12]:

```
data6 = pd.DataFrame({
                        'clo3':[100,200,300,400],
                        'col4':[500,600,700,800]
},index=['a','b','c','d'])
data6
```

Out[12]:

	clo3	col4
a	100	500
b	200	600
c	300	700
d	400	800

In [13]:

```
data5.join(data6)
```

Out[13]:

	clo1	col2	clo3	col4
a	10	50	100	500
b	20	60	200	600
c	30	70	300	700
d	40	80	400	800

In [ ]:

```

```

In [ ]:

```

```

In [14]:

```
# IF both data index column values are diff , what will be the result ?

data7 = pd.DataFrame({
                        'clo1':[10,20,30,40],
                        'col2':[50,60,70,80],
},index=['a','b','c','d'])
data7
```

Out[14]:

	clo1	col2
a	10	50
b	20	60
c	30	70
d	40	80

In [17]:

```
data8 = pd.DataFrame({
                        'clo3':[10,20],
                        'col4':[50,60],
},index=['a','b'])
data8
```

Out[17]:

	clo3	col4
a	10	50
b	20	60

In [18]:

```
data7.join(data8)
```

Out[18]:

	clo1	col2	clo3	col4
a	10	50	10.0	50.0
b	20	60	20.0	60.0
c	30	70	NaN	NaN
d	40	80	NaN	NaN

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
```

In [ ]:

```
# how :{left,right,outer,inner}
```

In [19]:

```
data10 = pd.DataFrame({
                        'clo1':[10,20,30,40],
                        'col2':[50,60,70,80],
},index=['a','b','c','d'])
data10
```

Out[19]:

	clo1	col2
a	10	50
b	20	60
c	30	70
d	40	80

In [24]:

```
data11 = pd.DataFrame({
                        'clo3':[100,200,4000],
                        'col4':[500,600,8000]
},index=['a','b','e'])
data11
```

Out[24]:

	clo3	col4
a	100	500
b	200	600
e	4000	8000

In [25]:

```
data10.join(data11,how='right')
```

Out[25]:

	clo1	col2	clo3	col4
a	10.0	50.0	100	500
b	20.0	60.0	200	600
e	NaN	NaN	4000	8000

In [26]:

```
data10.join(data11,how='left')
```

Out[26]:

	clo1	col2	clo3	col4
a	10	50	100.0	500.0
b	20	60	200.0	600.0
c	30	70	NaN	NaN
d	40	80	NaN	NaN

In [27]:

```
data10.join(data11,how='outer')
```

Out[27]:

	clo1	col2	clo3	col4
a	10.0	50.0	100.0	500.0
b	20.0	60.0	200.0	600.0
c	30.0	70.0	NaN	NaN
d	40.0	80.0	NaN	NaN
e	NaN	NaN	4000.0	8000.0

In [28]:

```
data10.join(data11,how='inner')
```

Out[28]:

	clo1	col2	clo3	col4
a	10	50	100	500
b	20	60	200	600

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
# lsuffix
# rsuffix
```

In [38]:

```
data13 = pd.DataFrame({
                        'clo1':[10,20,30],
                        'col2':[50,60,70],
},index=['a','b','c'])
data13
```

Out[38]:

	clo1	col2
a	10	50
b	20	60
c	30	70

In [41]:

```
data14 = pd.DataFrame({
                        'clo1':[100,200,4000],
                        'col3':[500,600,8000]
},index=['a','b','c'])
data14
```

Out[41]:

	clo1	col3
a	100	500
b	200	600
c	4000	8000

In [ ]:

```
'''
lsuffix : str, default ''
    Suffix to use from left frame's overlapping columns.
rsuffix : str, default ''
    Suffix to use from right frame's overlapping columns.

'''
```

In [44]:

```
data13.join(data14)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-44-704e54c6f489> in <module>
----> 1 data13.join(data14)

~\Anaconda3\lib\site-packages\pandas\core\frame.py in join(self, other, on,
 how, lsuffix, rsuffix, sort)
   6813         # For SparseDataFrame's benefit
   6814         return self._join_compat(other, on=on, how=how, lsuffix=lsuf
fix,
-> 6815                                  rsuffix=rsuffix, sort=sort)
   6816
   6817     def _join_compat(self, other, on=None, how='left', lsuffix='', r
suffix='',

~\Anaconda3\lib\site-packages\pandas\core\frame.py in _join_compat(self, oth
er, on, how, lsuffix, rsuffix, sort)
   6828             return merge(self, other, left_on=on, how=how,
   6829                          left_index=on is None, right_index=True,
-> 6830                          suffixes=(lsuffix, rsuffix), sort=sort)
   6831         else:
   6832             if on is not None:

~\Anaconda3\lib\site-packages\pandas\core\reshape\merge.py in merge(left, ri
ght, how, on, left_on, right_on, left_index, right_index, sort, suffixes, co
py, indicator, validate)
     46                          copy=copy, indicator=indicator,
     47                          validate=validate)
---> 48     return op.get_result()
     49
     50

~\Anaconda3\lib\site-packages\pandas\core\reshape\merge.py in get_result(sel
f)
    550
    551         llabels, rlabels = items_overlap_with_suffix(ldata.items, ls
uf,
--> 552                                                      rdata.items, rs
uf)
    553
    554         lindexers = {1: left_indexer} if left_indexer is not None el
se {}

~\Anaconda3\lib\site-packages\pandas\core\internals\managers.py in items_ove
rlap_with_suffix(left, lsuffix, right, rsuffix)
   1970         if not lsuffix and not rsuffix:
   1971             raise ValueError('columns overlap but no suffix specifie
d: '
-> 1972                              '{rename}'.format(rename=to_rename))
   1973
   1974         def lrenamer(x):

ValueError: columns overlap but no suffix specified: Index(['clo1'], dtype
='object')
```

In [45]:

```
data13.join(data14, lsuffix='_data-1')
```

Out[45]:

	clo1_data-1	col2	clo1	col3
a	10	50	100	500
b	20	60	200	600
c	30	70	4000	8000

In [46]:

```
data13.join(data14, rsuffix='_data-2')
```

Out[46]:

	clo1	col2	clo1_data-2	col3
a	10	50	100	500
b	20	60	200	600
c	30	70	4000	8000