In [1]:
```
import pandas as pd
import numpy as np
```

In [3]:
```
def make_df(cols, ind):
    """Quickly make a DataFrame"""
    data = {c: [str(c) + str(i) for i in ind]
            for c in cols}
    return pd.DataFrame(data, ind)

class display(object):
    """Display HTML representation of multiple objects"""
    template = """<div style="float: left; padding: 10px;">
    <p style='font-family:"Courier New", Courier, monospace'>{0}</p>{1}
    </div>"""
    def __init__(self, *args):
        self.args = args

    def _repr_html_(self):
        return '\n'.join(self.template.format(a, eval(a)._repr_html_())
                         for a in self.args)

    def __repr__(self):
        return '\n\n'.join(a + '\n' + repr(eval(a))
                           for a in self.args)
```

In [18]:
```
#simple concatenation
print("Series object 1=")
ser_1=pd.Series(["A","B","C"],index=[0,1,2])
print(ser_1)
print("Series object 2=")
ser_2=pd.Series(["D","E","F"],index=[3,4,5])
print(ser_2)
ser_12=pd.concat([ser_1,ser_2])#combining two series objects
print("Combining Series objects 1 and 2=")
print(ser_12)
df1 = make_df('AB', [1, 2])
df2 = make_df('AB', [3, 4])
display("df1","df2","pd.concat([df1, df2])")#concatenation of dataframes
df3 = make_df('AB', [0, 1])
df4 = make_df('CD', [0, 1])
print("Concatenation along row=")
display('df3', 'df4', "pd.concat([df3, df4])")#concatenation along rows
```

```
Series object 1=
0    A
1    B
2    C
dtype: object
Series object 2=
3    D
4    E
5    F
dtype: object
Combining Series objects 1 and 2=
0    A
1    B
2    C
3    D
4    E
5    F
dtype: object
Concatenation along row=
```

Out[18]:

df3

	A	B
0	A0	B0
1	A1	B1

df4

	C	D
0	C0	D0
1	C1	D1

pd.concat([df3, df4])

	A	B	C	D
0	A0	B0	NaN	NaN
1	A1	B1	NaN	NaN
0	NaN	NaN	C0	D0
1	NaN	NaN	C1	D1

In [17]:
```
display('df3', 'df4', "pd.concat([df3, df4],axis=1)")#concatenation along columns
```

Out[17]:

df3

	A	B
0	A0	B0
1	A1	B1

df4

	C	D
0	C0	D0
1	C1	D1

pd.concat([df3, df4],axis=1)

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1

In [24]:
```
#dealing with duplicate indexes
#throwing error for duplicate index using verify_integrity
df1 = make_df('AB', [1, 2])
df2 = make_df('AB', [2, 3])
try:
    display(pd.concat([df1, df2],verify_integrity=True))#try block to check for errors
except ValueError as e:
    print(e)#catchinbg error thrown if index are repeated
```

```
Indexes have overlapping values: Int64Index([2], dtype='int64')
```

In [26]:
```
#ignore index will create new integer index if indexes are repeated
display("df1","df2","pd.concat([df1,df2],ignore_index=True)")
```

Out[26]:

df1

	A	B
1	A1	B1
2	A2	B2

df2

	A	B
2	A2	B2
3	A3	B3

pd.concat([df1,df2],ignore_index=True)

	A	B
0	A1	B1
1	A2	B2
2	A2	B2
3	A3	B3

In [33]:
```
#adding multiIndex keys
display("df1","df2","pd.concat([df1,df2],keys=['a','b'])")
```

Out[33]:

df1

	A	B
1	A1	B1
2	A2	B2

df2

	A	B
2	A2	B2
3	A3	B3

pd.concat([df1,df2],keys=['a','b'])

		A	B
a	**1**	A1	B1
	2	A2	B2
b	**2**	A2	B2
	3	A3	B3

In [39]:
```
#concatenation with joins
df5 = make_df("ABC", [1, 2])
df6 = make_df("BCD", [3, 4])
#for union of 2 dataframes use join="outer"
#for intersection use join="inner
```

Out[39]:

df5

	A	B	C
1	A1	B1	C1
2	A2	B2	C2

df6

	B	C	D
3	B3	C3	D3
4	B4	C4	D4

pd.concat([df5, df6],join='inner')

	B	C
1	B1	C1
2	B2	C2
3	B3	C3
4	B4	C4

In [52]:
```
#we can directly specify the columns of resulting concatenation using join_axes
#however form internet it is observed that join axes is outdated
#using reindex is more optimal
df5 = make_df("ABC", [1, 2])
df6 = make_df("BCD", [3, 4])
df7=pd.concat([df5,df6.reindex(columns=df5.columns)],ignore_index=True)
print(df7)
df8=pd.concat([df5.reindex(columns=df6.columns),df6],ignore_index=True)
print(df8)
#append method-not efficient method and deprecated in pandas now
print(df5.append(df6))
```

```
     A   B   C
0   A1  B1  C1
1   A2  B2  C2
2  NaN  B3  C3
3  NaN  B4  C4
    B   C    D
0  B1  C1  NaN
1  B2  C2  NaN
2  B3  C3   D3
3  B4  C4   D4
     A   B   C    D
1   A1  B1  C1  NaN
2   A2  B2  C2  NaN
3  NaN  B3  C3   D3
4  NaN  B4  C4   D4

C:\Users\kdmag\AppData\Local\Temp\ipykernel_26004\1871981053.py:11: FutureWarning: The frame.append method is deprecated and wi
ll be removed from pandas in a future version. Use pandas.concat instead.
  print(df5.append(df6))
```