# 6.1 - Columns

July 12, 2017

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

In [2]: df = pd.DataFrame(index=range(2))

In [3]: def fill_df1(df):
            mydf = df.copy()
            for i in range(10000):
                mydf[i] = [i, -i]
            return mydf

In [4]: fill_df1(df)
```

Out[4]:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 9990 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 9990 |
| 1 | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9 | ... | -9990 |

| | 9991 | 9992 | 9993 | 9994 | 9995 | 9996 | 9997 | 9998 | 9999 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 9991 | 9992 | 9993 | 9994 | 9995 | 9996 | 9997 | 9998 | 9999 |
| 1 | -9991 | -9992 | -9993 | -9994 | -9995 | -9996 | -9997 | -9998 | -9999 |

[2 rows x 10000 columns]

```
In [5]: %timeit fill_df1(df)

1 loop, best of 3: 3.53 s per loop


In [6]: df2 = pd.DataFrame(index=range(2), columns=range(10000))

In [7]: %timeit fill_df1(df2)

1 loop, best of 3: 3.95 s per loop


In [8]: df3 = pd.DataFrame(-1, index=range(2), columns=range(10000))

In [9]: df3.dtypes.head()
```

```
Out[9]: 0    int64
        1    int64
        2    int64
        3    int64
        4    int64
        dtype: object

In [10]: %timeit fill_df1(df3)

1 loop, best of 3: 803 ms per loop


In [11]: df4 = pd.DataFrame(-1.0, index=range(2), columns=range(10000))

In [12]: %timeit fill_df1(df4)

1 loop, best of 3: 3.51 s per loop


In [13]: df5 = pd.DataFrame(-1.0, index=range(2), columns=range(10000))

In [14]: df5._data.blocks

Out[14]: (FloatBlock: slice(0, 10000, 1), 10000 x 2, dtype: float64,)

In [15]: df5[0] = [0, 0]

In [16]: df5._data.blocks

Out[16]: (FloatBlock: slice(1, 10000, 1), 9999 x 2, dtype: float64,
          IntBlock: slice(0, 1, 1), 1 x 2, dtype: int64)

In [17]: df5[1] = [1, -1]

In [18]: df5._data.blocks

Out[18]: (FloatBlock: slice(2, 10000, 1), 9998 x 2, dtype: float64,
          IntBlock: slice(0, 1, 1), 1 x 2, dtype: int64,
          IntBlock: slice(1, 2, 1), 1 x 2, dtype: int64)

In [19]: df5[2] = [2, -2]

In [20]: df5._data.blocks

Out[20]: (FloatBlock: slice(3, 10000, 1), 9997 x 2, dtype: float64,
          IntBlock: slice(0, 1, 1), 1 x 2, dtype: int64,
          IntBlock: slice(1, 2, 1), 1 x 2, dtype: int64,
          IntBlock: slice(2, 3, 1), 1 x 2, dtype: int64)

In [21]: df5._data.is_consolidated()
```

```
Out[21]: False

In [22]: m = df5.max()

In [23]: m.head()

Out[23]: 0    0.0
         1    1.0
         2    2.0
         3   -1.0
         4   -1.0
         dtype: float64

In [24]: df5._data.blocks

Out[24]: (FloatBlock: slice(3, 10000, 1), 9997 x 2, dtype: float64,
          IntBlock: slice(0, 3, 1), 3 x 2, dtype: int64)

In [25]: df5._data.is_consolidated()

Out[25]: True

In [28]: def fill_df2(df):
             mydf = df.copy()
             for i in range(1000):
                 mydf[i] = [i, -i]
                 mydf.max()
             return mydf

In [29]: def fill_df3(df):
             mydf = df.copy()
             for i in range(1000):
                 mydf[i] = [i, -i]
             for i in range(1000):
                 mydf.max()
             return mydf

In [30]: df6 = pd.DataFrame(-1.0, index=range(2), columns=range(1000))

In [31]: %timeit fill_df2(df6)

1 loop, best of 3: 573 ms per loop


In [32]: %timeit fill_df3(df6)

1 loop, best of 3: 341 ms per loop


In [33]: df7 = pd.DataFrame(-1, index=range(2), columns=range(1000))
```

```
In [34]: %timeit fill_df2(df7)

1 loop, best of 3: 258 ms per loop

In [35]: %timeit fill_df3(df7)

1 loop, best of 3: 216 ms per loop
```