# Pandas Basics

## Contents



## Create a DataFrame from scratch

One way is to use a dictionary, key is the column name, value is column data.

```
data = {
    'col1': [1, 2, 3, 4],
    'col2': [5, 6, 7, 8]
}

sampleDf = pd.DataFrame(data)
sampleDf
```

```
  col1  col2
0   1   5
1   2   6
2   3   7
3   4   8
```

Create a DataFrame with row indexes, giving meaningful row names:

```
row_names =['r1', 'r2', 'r3', 'r4']
sampleDf = pd.DataFrame(data, index=row_names)
```

```
  col1  col2
r1   1   5
r2   2   6
r3   3   7
r4   4   8
```

## DataFrame slicing, selecting, extracting

### By column

Similar to access dictionary:

```
data['col1'] => [1, 2, 3, 4]
```

```
sampleDf['col1']
```

| | col1 | col2 |
|---|---|---|
| r1 | 1 | 5 |
| r2 | 2 | 6 |
| r3 | 3 | 7 |
| r4 | 4 | 8 |

```
type(sampleDf['col1']) => Series
```

Use a list of columns will return a DataFrame:

```
type(sampleDf[['col1', 'col2']])
```

### By Row

.loc - locate by name
.iloc- locate by numerical index

```
sampleDf.loc[['r1', 'r2']]
sampleDf.iloc[[0, 1]]

sampleDf.loc['r1':'r3']
sampleDf.iloc[0:3]  # exclude last index

sampleDf.loc[['r1', 'r2'], ['col1']]
```

### Conditional selections

```
condition = (sampleDf['col1']=='3')
or
condition = (sampleDf.col1=='3')
```

```
condition
r1   False
r2   False
r3   True
r4   False

sampleDf[condition]
   col1  col2
r3    3    7

or
sampleDf[sampleDf.col1==3]
```

```
sampleDf[sampleDf['col1'].isin([1,3])]
sampleDf[((sampleDf['col1']==1) | (sampleDf['col1']==3))]
sampleDf[sampleDf['col1']>2]
```

```
sampleDf['doulble'] = sampleDf['col1'].apply(lambda x: x*2)
```

## Pivot table vs Groupby

Both pivot_table and groupby are used to aggregate DataFrame. The difference is only the shape of the result.

```
df = pd.DataFrame({"a": ['a1','a2','a3','a1','a2','a3'],
"b":['b1','b1','b1','b2','b2','b2'], "c":np.random.rand(6)})
```

```
>>>>df
    a   b         c
0  a1  b1  0.578818
1  a2  b1  0.250969
2  a3  b1  0.159156
3  a1  b2  0.328347
4  a2  b2  0.109600
5  a3  b2  0.255304
```

## Pivot table

```
pivotDf = pd.pivot_table(df, index=["a"], columns=["b"], values=["c"],
aggfunc=np.sum)
```

```
>>>>pivotDf
           c
b          b1        b2
a
a1   0.578818  0.328347
a2   0.250969  0.109600
a3   0.159156  0.255304
```

Result: a is on the row axis, b is on the column axis, and the values are the sum of c, based on distinct values of a and b.

## Group By

```
groupbyDf = df.groupby(['a','b'])['c'].sum()

>>>>groupbyDf
a   b
a1  b1    0.578818
    b2    0.328347
a2  b1    0.250969
    b2    0.109600
a3  b1    0.159156
    b2    0.255304
```

Result: rows are created for each combination of distinct values of a and b.

## Stack and Unstack

```
>>>>pivotDf.stack()
          c
a  b
a1 b1  0.578818
   b2  0.328347
a2 b1  0.250969
   b2  0.109600
a3 b1  0.159156
   b2  0.255304
```

Result is similar to groupby.

```
>>>>groupbyDf.unstack()
b          b1        b2
a
a1   0.578818  0.328347
a2   0.250969  0.109600
a3   0.159156  0.255304
```

Result is similar to pivot table.

Conclusion: a pivot table is already unstacked, while a groupby result is stacked.