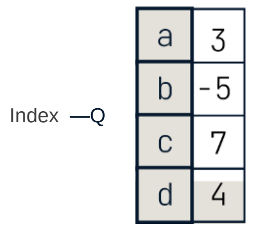
Cheat sheet
of
Pandas

Use **the following import convention:**

```
>>> import pandas as pd
```

Series

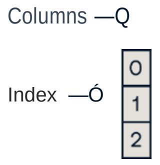
A **one-dimensional** labeled array
capable of holding any data type



```
>>> s = pd.Series([3, - 5,  7, 4], index=['a', 'b', 'c', 'd'])
```

Data frame

A **two-dimensional** labeled data structure
with columns of potentially different types



```
>>> data:  ('Country': ['Belgium', 'India', 'Brazil'],
         'Capital': ['Brussels', 'New  Delhi', 'Brasilia'],
         'Population'  [11190846, 1303171035, 207847528]}
>>> df: pd.DataFrame(data, columns:['Country', 'Capital',
'Population'])
```

Dropping

```
>>> s.drop(['a', ' ']) #Orop values from ross (ax1s——0)
>>> df.drop('Count by', axis=1) #Dnop values from columns (axis=1)
```

Asking for help

```
>>> help(pd.Series.loc)
```

Sort and rank

```
>>> df.sort_index() #SorI by I abel s al ong an ax1s
>>> df.sort_va1ues (by='Count by') #Sort by the values along an ax1s
>>> df.rank() #Ass1gn ranks to entr1es
```

I/O

Read and Write to CSV

```
>» pd.read csv('file.csv', header=None, nrows=5)
>>> df . to _csv('uyData Frase . csv ')
```

Reod ond Write to Excel

```
>>> pd.read_excel( file .xlsx' )
>>> df.to_excel('dir/myDataFrame.xlsx', sheet name:'Sheetl')
```

Read multiple sheets from the some file

```
>> xlsx = pd.ExcelFile('file.xls)
>>› df   pd.read excel(xlsx, 'Sheetl')
```

Reod ond Write to SQL Querg or Dotobase Toble

```
>>> from sqlalchemy import create engine
>>> engine = create engine('sqlite:/// :memory:')
>>> pd.read sql("SELECT * FROM my_tabIe;", engine)
>>› pd.read sql table('my_table', engine)
>>> pd.read sqI query("SELECT * FROM my_table ;", engine)
```

read sql() iso conveniencevvropper oround read sql table() ond read sql query()

```
>>> df.to_sql('myDf', engine)
```

Selection

Getting

```
>>> s['b'] #Oet one eì ement
 -5
>>  df[1:] #Get sUdset oy 0 DOtOFrome
   Country Capital Population
 1 India New Delhi 1303171035
 2 Brazil Brasilia 207847528
```

Selecting, Boolean Indexing & Setting

Bg Position

```
>>> df .il oc[[ò], [ò]] #SeT ect s1ngT e vaT ue bg ron 6 co1u/rn
 'Betg iu u'
>>> df .iat ([ò] , [ò] )
 'Bet g:iu u'
```

Bg Label

```
>>> df .1OC[[ò] , ['CoUnt r'y']] #SeT ect s:ingTe value bg ron 6 co buon TabeTs
 'Betg iU u'
>>> df.at([0], ['CoUntry'])
 'Betgius '
```

Bg Label/Position

```
>>> df .ix [2] #SeT ect s:inghe non of subset oy ross
 Country Brazil
 Capital Brasilia
 Poputation 2ò7847528
>>> df .ix [:, 'Capita T'] #SeT ect a s:ìngT e coT uhm oy subset oy coT u/rns
 0 Brussels
 1 New Delhi
 2 Brasilia
>>> df .ix [1, 'Capita T'] #SeT ect ross and coT uiiins
 'New Delhi'
```

Boolean Indexing

```
>>> s[- (s > 1)] #Sen1es s where voT ue :is not >1
>>> s[(s < -1) | (s > 2)] #s where voT ue :is m1 or >2
>>> df [df ['POpUtation']>1 2òòòòòòòò j #ùse f:i T ter to add ust Oat aFra me
```

Setting

```
>>> s['a'] = ó #Set :index a of Sez:i es s to b
```

Retriving information

Bosic Information

```
>>> df.shape Pro as, coTu/rns)
>>> df.1ndex #0e scr1be 1ndex
>>> df.columns #0e scr1be OataFraue col urns
>>> df.info() #!nfo on OataFraue
>>> df.count(} #ku/rben of non-UA values
```

Summary

```
>>> df.sum() sum of values
>>> df.cumsum() #Cumulative sum of values
>>> df.min() /df.max() #maximum/minimun values
>>> df.idxmin(J /df.idxmax() #min/max :index value
>>> df.describe() #Su/r/rar g staII st1cs
>>> df.mean() #mean oy values
>>> df.median () #med1an of value
```

Applying Functions

```
>>> f = lambda x: xe2
>>> df.apply(f) #Apply function
>>> df.applymap(f) #Apply function on element-wise
```

Data Alignment

Internal Data Alignment

NA values ore introduced in the indices thot don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd']
>>> s + s3
 a 10.Ò
 b Nan
 c 5. 0
 d 7. 0
```

Arithmetic Operations with Fili Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s . add (s3, fill_values=ò)
a 1ò.ò
 b -5. ò
 c 5. ò
 d 7.0
>>> s . sub f s3, fill_values=2)
>>> s . div (s3, fill_values=4)
>>> s . riut (s3, fill_values=3)
```