

# Pandas Basics Cheat Sheet

f616 adapted from datacamp.com

## Introductory Note

This document is an adaption of the original datacamp.org cheat sheet.

- <https://www.datacamp.com/resources/cheat-sheets/pandas-cheat-sheet-for-data-science-in-python>
- <https://github.com/f616/Python-Pandas-Basics-Cheat-Sheet>

## Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

### Use the following import convention:

```
1 import pandas as pd
```

## 1 Pandas Data Structures

### Series

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

index->	a	3
	b	-5
	c	7
	d	4

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

### Dataframe

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

columns->		Country	Capital	Population
index->	0	Belgium	Brussels	11190846
	1	India	New Delhi	1303171035
	2	Brazil	Brasilia	207847528

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

## 2 Dropping

```
1 s.drop(['a', 'c']) #Drop values from rows (axis=0)  
2 df.drop('Country', axis=1) #Drop values from columns (axis=1)
```

## 3 Asking For Help

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

## 4 Sort & Rank

```
1 df.sort_index() #Sort by labels along an axis  
2 df.sort_values(by='Country') #Sort by the values along an  
  axis  
3 df.rank() #Assign ranks to entries
```

## 5 I/O

### Read and Write to CSV

```
1 pd.read_csv('file.csv', header=None, nrows=5)  
2 df.to_csv('myDataFrame.csv')
```

### Read and Write to Excel

```
1 xlsx = pd.ExcelFile('file.xlsx')  
2 df = pd.read_excel(xlsx, 'Sheet1') #Read from xlsx file  
  Sheet1  
3 df.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')  
  #Save to xlsx file
```

### Read and Write to SQL Query or Database Table

```
1 from sqlalchemy import create_engine  
2 engine = create_engine('sqlite:///memory:')  
3 pd.read_sql("SELECT * FROM my_table;", engine)  
4 pd.read_sql_table('my_table', engine)  
5 pd.read_sql_query("SELECT * FROM my_table;", engine)  
6  
7 # read_sql() is a convenience wrapper around read_sql_table()  
  and read_sql_query()  
8  
9 df.to_sql('myDf', engine)
```

## 6 Selection

### Getting

```
1 s['b'] #Get one element  
2 -5  
3 df[1:] #Get subset of a DataFrame  
4 Country Capital Population  
5 1 India New Delhi 1303171035  
6 2 Brazil Brasilia 207847528
```

### Selecting, Boolean Indexing & Setting

#### By Position

```
1 df.iloc[[0],[0]] #Select single value by row & column  
2 'Belgium'  
3 df.iat[0,0]  
4 'Belgium'
```

#### By Label

```
1 df.loc[[0], ['Country']] #Select single value by row &  
  column labels  
2 'Belgium'  
3 df.at[0, 'Country']  
4 'Belgium'
```

#### Boolean Indexing

```
1 s[~(s > 1)] #Series s where value is not >1  
2 s[(s < -1) | (s > 2)] #s where value is <-1 or >2  
3 df[df['Population']>1200000000] #Use filter to adjust  
  DataFrame
```

#### Setting

```
1 s['a'] = 6 #Set index a of Series s to 6
```

## 7 Retrieving Series/DataFrame Information

### Basic Information

```
1 df.shape # (rows, columns)
2 df.index # Describe index
3 df.columns # Describe DataFrame columns
4 df.info() # Info on DataFrame
5 df.count() # Number of non-NA values
```

### Summary

```
1 df.sum() # Sum of values
2 df.cumsum() # Cumulative sum of values
3 df.min() # Minimum values
4 df.max() # Maximum values
5 df.idxmin() # Minimum index value
6 df.idxmax() # Maximum index value
7 df.mean() # Mean of values
8 df.median() # Median of values
9 df.describe() # Summary statistics
```

## 8 Data Alignment

### Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
1 s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
2 s + s3
3 a 10.0
4 b NaN
5 c 5.0
6 d 7.0
```

### Arithmetic Operations with Fill Methods

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

```
1 s.add(s3, fill_values=0)
2 a 10.0
3 b -5.0
4 c 5.0
5 d 7.0
6 s.sub(s3, fill_value=2)
7 s.div(s3, fill_value=4)
8 s.mul(s3, fill_value=3)
```