Python for Data Science
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:

```
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

```
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->

index->

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

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

2 Dropping

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

3 Asking For Help

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

4 Sort & Rank

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

5 I/O

Read and Write to CSV

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

Read and Write to Excel

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

Read and Write to SQL Query or Database Table

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///:memory:')
pd.read_sql("SELECT * FROM my_table;", engine)
pd.read_sql_table('my_table', engine)
pd.read_sql_query("SELECT * FROM my_table;", engine)

# read_sql() is a convenience wrapper around read_sql_table() and read_sql_query()

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

6 Selection

Getting

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

Selecting, Boolean Indexing & Setting

By Position

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

By Label

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

Boolean Indexing

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

Setting

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

7 Retrieving Series/DataFrame Information

Basic Information

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

Summary

```
df.sum()   #Sum of values
df.cumsum()   #Cummulative sum of values
df.min()   #Minimum values
df.max()   #Maximum values
df.idxmin()   #Minimum index value
df.idxmax()   #Maximum index value
df.mean()   #Mean of values
df.median()   #Median of values
df.describe()   #Summary statistics
```

8 Data Alignment

Internal Data Alignment

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

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

Arithmetic Operations with Fill Methods

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

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