Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com



Pandas

Use the following import convention:

>>> import pandas as pd

Pandas Data Structures

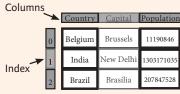
Series

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



>>> 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'])

Asking For Help

>>> help(pd.Series.loc)

Selection

Also see NumPy Arrays

Select single value by row &

Set index a of Series s to 6

column

Getting

```
>>> s['b']
-5
>>> df[1:]
Country Capital Population
1 India New Delhi 1303171035
2 Brazil Brasilia 207847528

Get one element

Get subset of a DataFrame
```

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]]
    'Belgium'
>>> df.iat([0],[0])
    'Belgium'
```

By Label

>>> df.loc[[0], ['Country']]	Select single value by row & column labels
>>> df.at([0], ['Country'])	
'Belgium'	

By Label/Position

>>> df.ix[2]	Select single row of
Country Brazil	subset of rows
Capital Brasília	
Population 207847528	
>>> df.ix[:,'Capital']	Select a single column of
0 Brussels	subset of columns
1 New Delhi	
2 Brasília	
>>> df.ix[1,'Capital']	Select rows and columns

Boolean Indexing

'New Delhi'

>>> 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
Caustin II	

Setting

	>>>	s [a']	=	6	
--	-----	-----	----	---	---	---	--

Read and Write to SQL Query or Database Table

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

Read and Write to Excel

Read and Write to CSV

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
Read multiple sheets from the same file
```

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

>>> 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()

>>> pd.to sql('myDf', engine)

Dropping

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

Sort & Rank

>>>	<pre>df.sort_values(by='Country')</pre>	Sort by labels along an axis Sort by the values along an axis Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

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

Summary

<pre>>>> df.idxmin()/df.idxmax() >>> df.describe() >>> df.mean()</pre>	Summary statistics Mean of values
>>> df.mean() >>> df.median()	Mean of values Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f) Apply function
>>> df.applymap(f) Apply function element-wise
```

Data Alignment

Internal Data Alignment

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

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_value=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)
```

DataCamp

