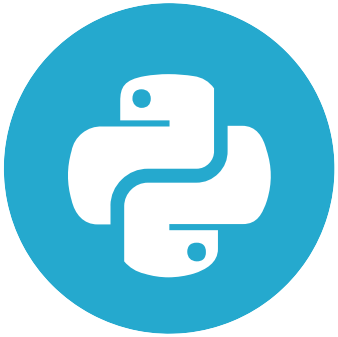


Pandas Cheat Sheet for Data Science in Python

November 2nd, 2016 in Python

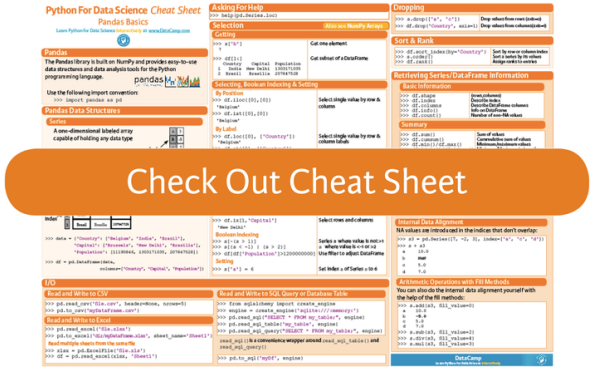
Karlijn Willems

The Pandas library is one of the most preferred tools for data scientists to do data manipulation and analysis, next to matplotlib for data visualization and NumPy, the fundamental library for scientific computing in Python on which Pandas was built.

The fast, flexible, and expressive Pandas data structures are designed to make real-world data ana
might not be immediately the case for those who are just getting started with it. Exactly because th
into this package that the options are overwhelming.

That's where this Pandas cheat sheet might come in handy.

It's a quick guide through the basics of Pandas that you will need to get started on wrangling your d



The Pandas cheat sheet will guide you through the basics of the Pandas library, going from the data structures to I/O, selection, dropping indices or columns, sorting and ranking, retrieving basic information of the data structures you're working with to applying functions and data alignment.

In short, everything that you need to kickstart your data science learning with Python!

Do you want to learn more? Start the Intermediate Python For Data Science course for free now or try out our Pandas DataFrame tutorial!

Also, don't miss out on our Bokeh cheat sheet for data visualization in Python and our Python cheat sheet for data science.

(Click above to download a printable version or read the online version below.)

Python For Data Science Cheat Sheet: Pandas Basics

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'])
```

A	3
B	5
C	7
D	4

DataFrame

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'])
```

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

Asking For Help

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

I/O

Read and Write to CSV

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

Read multiple sheets from the same file

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

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx',  sheet_name='Sheet1')
```

Read and Write to SQL Query or Database Table

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

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

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

Selection

Getting

Get one element

```
>>> s['b']
-5
```

Get subset of a DataFrame

```
>>> df[1:]
Country     Capital   Population
1  India    New Delhi 1303171035
2  Brazil   Brasilia  207847528
```

Selecting', Boolean Indexing and Setting

By Position

Select single value by row and and column

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

```
>>> df.loc([0],  ['Country'])
'Belgium'
>>> df.at([0],  ['Country'])
'Belgium'
```

By Label/Position

Select single row of subset of rows

```
>>> df.ix[2]
Country      Brazil
Capital    Brasilia
Population  207847528
```

Select a single column of subset of columns

```
>>> df.ix[:, 'Capital']
0     Brussels
1    New Delhi
2     Brasilia
```

Select rows and columns

```
>>> df.ix[1, 'Capital']
'New Delhi'
```

Boolean Indexing

Series s where value is not >1

s where value is <-1 or >2

```
>>> s[(s < -1) | (s > 2)]
```

Use filter to adjust DataFrame

```
>>> df[df['Population']>1200000000]
```

Setting

Set index a of Series s to 6

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

Dropping

Drop values from rows (axis=0)

```
>>> s.drop(['a',  'c'])
```

Drop values from columns(axis=1)

```
>>> df.drop('Country', axis=1)
```

Sort and Rank

Sort by labels along an axis

Sort by the values along an axis

```
>>> df.sort_values(by='Country')
```

Assign ranks to entries

```
>>> df.rank()
```

Retrieving Series/DataFrame Information

Basic Information

(rows, columns)

```
>>> df.shape
```

Describe index

```
>>> df.index
```

Describe DataFrame columns

```
>>> df.columns
```

Info on DataFrame

```
>>> df.info()
```

```
>>> df.count()
```

Summary

Sum of values

```
>>> df.sum()
```

Cumulative sum of values

```
>>> df.cumsum()
```

Minimum/maximum values

```
>>> df.min()/df.max()
```

Minimum/Maximum index value

```
>>> df.idxmin()/df.idxmax()
```

Summary statistics

```
>>> df.describe()
```

Mean of values

```
>>> df.mean()
```

```
>>> df.median()
```

Applying Functions

```
>>> f = lambda x: x*2
```

Apply function

```
>>> df.apply(f)
```

Apply function element-wise

```
>>> df.applymap(f)
```

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:

```
c    5.0
d    7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

What do you think?

Python Learning Data Science

Up Next

New Course! Supervised Learning in R: Classification

R Programming 73 views

This beginner-level introduction to machine learning covers four of the most common classification algorithms. You will come away with a b…

New Course: Case Studies in Statistical Thinking!

Python 1,139 views

Hone your applied data science skills in Python by doing a variety of case studies across multiple disciplines. Use data science to solve …

September 20th, 2017 in Blog

Jupyter Notebook Cheat Sheet

Python 19,148 views

This Jupyter Notebook cheat sheet will help you to find your way around the well-known Jupyter Notebook App, a subproject of Project Jupyt..

Comments

mariakatosvich

Nice cheatsheet! make use of DataCamp's Pandas Cheat Sheet when you're considering loading files as Pandas DataFrames.If you have the data in an array, you can inspect it by making use of the following array attributes: shape, ndim, dtype, etc:
Thanks
http://driverwhiz.com/device-drivers

03/28/17 12:34 PM |

jvandenbossche

Nice cheatsheet! Just wanted to point out of few small errors:

- df.idmin()/df.idmax() -> df.idxmin()/df.idxmax() ('x' is missing)

For the "sort & rank" section, there have been recently some deprecations / changes in the API:
- s.order() -> deprecated, better to use s.sort_values()
- df.sort_index(by='Country') -> 'by' is deprecated, better to use df.sort_index() (for sorting index), or df.sort_values(by='Country') for sorting a column

01/03/17 8:29 AM |

karlijn

Hi there, thank you very much for your feedback! I made the necessary changes and uploaded a new version of the cheat sheet, so you should be able to see the changes when you click on the picture above. Thank you once again!

01/18/17 8:47 PM |

davidsantucci

Tnks! That's awesome! :)

12/11/16 12:49 PM |

karlijn

Thanks, David! I'm glad you like it :)

12/14/16 10:25 AM |

cruseakshay

concisely amazing & i love it !

11/13/16 12:51 PM |

karlijn

Thanks!!