Karlijn Willems
November 2nd, 2016

PYTHON +2

Pandas Cheat Sheet for Data Science in Python

A quick guide to the basics of the Python data analysis library Pandas, including code samples.

The Pandas library is one of the most

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 analysis significantly easier, but this might not be immediately the case for those who are just getting started with it. Exactly because there is so much functionality built into this package that the options are overwhelming.

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

your data with Python.

As such, you can use it as a handy reference if you are just beginning their data science journey with Pandas or, for those of you who already haven't started yet, you can just use it as a guide to make it easier to learn about and use it.

Python Pandas Cheat Sheet

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

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 Pandas Data Wrangling cheat sheet or our other data science cheat sheets.

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

Sheet: Pandas Basics

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	3
B	5
C	7

DataFrame

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

```
>>> data = {'Country': ['Belgium',  'India

'Capital': ['Brussels',  'New Delhi',  'Br

'Population': [11190846, 1303171035, 20784
>>> df = pd.DataFrame(data,columns=['Count
```

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

Columns.

Asking For Help

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

I/O

Read and Write to CSV

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

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx,  'Sheet1')
>>> pd.read_excel('file.xlsx')
>>> df.to_excel('dir/myDataFrame.xlsx',  s
```

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:///:mem
>>> pd.read_sql(SELECT * FROM my_table;, e
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query(SELECT * FROM my_tab
```

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

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

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

By Label

Select single value by row and column labels

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

By Label/Position

Select single row of subset of rows

```
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

```
>>> s[~(s > 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
```

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

```
>>> df.sort_index()
```

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
>>> df.columns
```

Info on DataFrame

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

Number of non-NA values

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

Summary

Sum of values

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

Minimum/maximum values

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

Minimum/Maximum index value

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

Summary statistics

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

Mean of values

Median of values

```
>>> 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'
>>> 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_value=0)
a     10.0
```

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

▲ 54

11

COMMENTS

Yuting Liao
17/06/2018 05:13 AM

Thanks for this good summary.

There’s an error in the I/O section:

>>> pd.to_csv('myDataFrame.csv')

`'pandas' has no attribute 'to_csv'"`.

`In fact,` `to_csv` is a method of a `DataFrame` object, not of the `pandas` module.

So the correct one should be

```
>>> df.to_csv('myDataFrame.csv')
```

▲ 12

saikrishna thatikonda
22/08/2018 08:37 AM
Yes you are right Liao.

▲ 1

Uttam Mali
24/01/2019 06:48 AM
I want you be friends with me

▲ 5

Arwin Edra
23/01/2019 01:51 AM

Hiii

▲ 1

Uttam Mali
24/01/2019 06:41 AM

I want to know more about

▲ 1

Michael Kendagor
20/02/2019 07:47 AM

Thanks for the preview

▲ 2

Thanks guys. Hoping to learn more!

▲ 1

vaishu barathi
28/06/2019 08:31 AM

Have you been thinking about the power sources and the tiles whom use blocks I wanted to thank you for this great read!! I definitely enjoyed every little bit of it and I have you bookmarked to check out the new stuff you post

MSBI online training

▲ 1

Renuka Peshwani
23/07/2019 11:41 AM

Thanks @karlijn, for such a informative Python Pandas cheat sheet. Pandas is popularly used Python Library, which has a code than python. If any one can get a proper guidance for Pandas, s/he can achieve their dream job.

▲ 1

Ta Nam
25/09/2019 05:32 AM

I read it in your cheatsheet however I cannot replicate the final code related to df.select. Pls give me some instructions about this. Thanks
Indexing With isin
>>> df[(df.Country.isin(df2.Type))]
>>> df3.filter(items=”a”,”b”])
>>> df.select(lambda x: not x%5)

▲ 3

Aditi Mishra

Python package for data science. Not only does it give you lots of methods and functions that make working with data easier, but it has been optimized for speed which gives you a significant advantage compared with working with smartwatchesforwomens numeric data using Python’s built-in functions.

▲ 1