The Pandas library is one of the most powerful libraries in Python. It is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

Check out the sections below to learn the various functions and tools Pandas offers.

> ***Sections:***

Pandas Data Structures

There are two main types of data structures that the Pandas library is centered around. The first is a one-dimensional array called a **Series**, and the second is a two-dimensional table called a **Data Frame**.

- Series — One dimensional labeled array

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

- Data Frame — A two dimensional labeled data structure

```
>>> data = {'Country':['Belgium','India','Brazil'], 'Capital':
['Brussels','New Delhi','Brasilia'], 'Population':
['111907','1303021','208476']}
```

```
>>> df = pd.DataFrame(data, columns =
['Country','Capital','Population'])

   Country    Capital Population
0  Belgium   Brussels     111907
1    India  New Delhi    1303021
2   Brazil   Brasilia     208476
```

Dropping

In this section, you'll learn how to remove specific values from a Series, and how to remove columns or rows from a Data Frame.

s and **df** in the code below are used as examples of a Series and Data Frame throughout this section.

```
>>> s

a    6
b   -5
c    7
d    4

>>> df

   Country    Capital  Population
0  Belgium   Brussels      111907
1    India  New Delhi     1303021
2   Brazil   Brasilia      208476
```

- Drop values from rows (axis = 0)

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

 b   -5
 d    4
```

- Drop values from columns (axis = 1)

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

    Capital Population
0   Brussels     111907
1  New Delhi    1303021
2   Brasilia     208476
```

Sort & Rank

In this section, you'll learn how to sort Data Frames by an index, or column, along with learning how to rank column values.

df in the code below is used as an example Data Frame throughout this section.

```
>>> df

   Country    Capital  Population
0  Belgium   Brussels      111907
1    India  New Delhi     1303021
2   Brazil   Brasilia      208476
```

- Sort by labels along an axis

```
>>> df.sort_index()

   Country    Capital Population
0  Belgium   Brussels     111907
1    India  New Delhi    1303021
2   Brazil   Brasilia     208476
```

- Sort by values along an axis

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

   Country    Capital Population
0  Belgium   Brussels     111907
2   Brazil   Brasilia     208476
1    India  New Delhi    1303021
```

- Assign ranks to entries

```
>>> df.rank()

   Country  Capital  Population
0      1.0      2.0         1.0
1      3.0      3.0         2.0
2      2.0      1.0         3.0
```

Retrieving Series/DataFrame Information

In this section, you'll learn how to retrieve info from a Data Frame that includes the dimensions, column names column types, and index range.

df in the code below is used as an example Data Frame throughout this section.

```
>>> df

   Country    Capital  Population
0  Belgium   Brussels      111907
1    India  New Delhi     1303021
2   Brazil   Brasilia      208476
```

- (rows, columns)

```
>>> df.shape
 (3, 3)
```

- Describe index

```
>>> df.index
 RangeIndex(start=0, stop=3, step=1)
```

- Describe DataFrame columns

```
>>> df.columns
 Index(['Country', 'Capital', 'Population'], dtype='object')
```

- Info on DataFrame

```
>>> df.info()

 <class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 3 columns):
Country       3 non-null object
Capital       3 non-null object
Population    3 non-null object
dtypes: object(3)
memory usage: 152.0+ bytes
```

- Number of non-NA values

```
>>> df.count()

Country       3
Capital       3
Population    3
```

DataFrame Summary

In this section, you'll learn how to retrieve summary statistics of a Data Frame which include the sum of each column, min/max values of each column, mean values of each column, and others.

df in the code below is used as an example of a Data Frame throughout this section.

```
>>> df

   Even  Odd
0     2    1
1     4    3
2     6    5
```

- Sum of values

```
>>> df.sum()
Even    12
Odd      9
```

- Cumulative sum of values

```
>>> df.cumsum()

   Even  Odd
0     2    1
1     6    4
2    12    9
```

- Minimum value

```
>>> df.min()
Even    2
Odd     1
```

- Maximum value

```
>>> df.max()
Even    6
Odd     5
```

- Summary statistics

```
>>> df.describe()

       Even  Odd
count   3.0  3.0
mean    4.0  3.0
std     2.0  2.0
min     2.0  1.0
25%     3.0  2.0
50%     4.0  3.0
75%     5.0  4.0
max     6.0  5.0
```

- Mean of values

```
>>> df.mean()
Even    4.0
Odd     3.0
```

- Median of values

```
>>> df.median()
Even    4.0
Odd     3.0
```

Selection

In this section, you’ll learn how to retrieve specific values from a Series and Data Frame.

s and **df** in the code below are used as examples of a Series and Data Frame throughout this section.

```
>>> s

a    6
b   -5
c    7
d    4

>>> df

   Country    Capital  Population
0  Belgium   Brussels      111907
1    India  New Delhi     1303021
2   Brazil   Brasilia      208476
```

- Get one element

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

- Get subset of a DataFrame

```
>>> df[1:]

  Country    Capital Population
1   India  New Delhi    1303021
2  Brazil   Brasilia     208476
```

- Select single value by row & column

```
>>> df.iloc[0,0]
 'Belgium'
```

- Select single value by row and column labels

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

- Select single row of subset rows

```
>>> df.ix[2]

Country         Brazil
Capital       Brasilia
Population      208476
```

- 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'
```

- Use filter to adjust DataFrame

```
>>> df[df['Population'] > 120000]

  Country    Capital  Population
1   India  New Delhi    1303021
2  Brazil   Brasilia     208476
```

- Set index a of Series s to 6

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

 a    6
 b   -5
 c    7
 d    4
```

Applying Functions

In this section, you'll learn how to apply a function to all values of a Data Frame or a specific column.

df in the code below is used as an example of a Data Frame throughout this section.

```
>>> df

   Even  Odd
0     2    1
1     4    3
2     6    5
```

- Apply function

```
>>> df.apply(lambda x: x*2)

   Even  Odd
0     4    2
1     8    6
2    12   10
```

Data Alignment

In this section, you'll learn how to add, subtract, and divide two series that have different indexes from one another.

s and **s3**in the code below are used as examples of Series throughout this section.

```
>>> s

a    6
b   −5
c    7
d    4

>>> s3

a    7
c   −2
d    3
```

- Internal Data Alignment

```
>>> s + s3

a    13.0
b     NaN
c     5.0
d     7.0

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

- Arithmetic Operations with Fill Methods

```
>>> s.add(s3, fill_value = 0)

a    13.0
b    −5.0
c     5.0
d     7.0

>>> s.sub(s3, fill_value = 2)
```

```
a   -1.0
b   -7.0
c    9.0
d    1.0

>>> s.div(s3, fill_value = 4)

a    0.857143
b   -1.250000
c   -3.500000
d    1.333333
```

In/Out

In this section, you'll learn how to read a CSV file, Excel file, and SQL Query into Python using Pandas. You will also learn how to export a Data Frame from Pandas into a CSV file, Excel file, and SQL Query.

- Read CSV file

```
>>> pd.read_csv('file.csv')
```

- Write to CSV file

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

- Read Excel file

```
>>> pd.read_excel('file.xlsx')
```

- Write to Excel file

```
>>> pd.to_excel('dir/'myDataFrame.xlsx')
```

- Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')

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

- Read SQL Query

```
>>> 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)
```

- Write to SQL Query

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

Python is the top dog when it comes to data science for now and in the foreseeable future. Knowledge of Pandas, one of its most powerful libraries is often a requirement for Data Scientists today.

Use this cheat sheet as a guide in the beginning and come back to it when needed, and you'll be well on your way to mastering the Pandas library.