Cheatsheet
Pandas

1. Introduction

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
```

2. Pandas Data Structures

Series

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

Use the following function to create series

```
s = Series (data, index)
```

Example

```
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = pd.Series(data,index=['b','c','d','a'])
```

DataFrame

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

```
df = DataFrame (data, index, columns)
```

```
data = {'a' : 0., 'b' : 1., 'c' : 2.}
df = pd.DataFrame(data)
```

3. Importing Data

pd.read_csv(filename) - From a CSV file
pd.read_table(filename) - From a delimited text file (like TSV)
pd.read_excel(filename) - From an Excel file
pd.read_sql(query, connection_object) - Read from a SQL table or database
pd.read_json(json_string) - Read from a JSON formatted string, URL or file.
pd.read_html(url) - Parses an html URL, string or file and extracts tables to a list of dataframes
pd.read_clipboard() - Takes the contents of your clipboard and passes it to read_table()
pd.DataFrame(dict) - From a dict, keys for columns names, values for data as lists

4. Exporting Data

df.to_csv(filename) - Write to a CSV file
.to_(filename) - Write to an Excel file
df.to_(table_name, connection_object) - Write to a SQL table
df.to_json(filename) - Write to a file in JSON format
df.to_html(filename) - Save as an HTML table
df.to_clipboard() - Write to the clipboard

5. Observing the Data

df.head(n) - Shows first n rows from the dataset
df.tail(n) - Shows last n rows from the dataset
df.describe() - Generate various summary statistics for every column in the dataset
df.shape - Number of rows and columns
df.apply() - Apply function along a given axis
df.applymap() - Apply function to every element in DataFrame
df.value_counts() - Return a Series containing counts of unique value
df.unique() - Returns Unique elements from a series

6. Selection

df[col] - Return column with label col as Series
df[[col1, col2]] - Return Columns as a new DataFrame
df.iloc[0] - selection by position
df.loc[0] - selection by index
df.loc[:,'x2':'x4'] - Select all columns between x2 and x4 (inclusive)
df.iloc[:,[1,2,5]] - Select columns in positions 1, 2 and 5 (first column is 0).
df.loc[df['a'] > 10, ['a','c']] - Select rows meeting logical condition, and only the specific columns
df.sample(frac=0.5) - Randomly select fraction of rows
df.sample(n=10) - Randomly select n rows
df.iloc[10:20] - Select rows by position
df.nlargest(n, 'value') - Select and order top n entries
df.nsmallest(n, 'value') - Select and order bottom n entries

7. Filter, Sort and Groupby

pd.sort() - Sort index
pd.sort(axis=1) - Sort Columns
df.sort_index() - Sort the index of a DataFrame
df.sort_values('mpg') - Order rows by values of a column (low to high)
df.value_counts(dropna=False) - View unique values and counts
df.groupby() - Split DataFrame by columns. Creates a GroupBy object (gb)
gb.agg() - Apply function (single or list) to a GroupBy object
gb.transform() - Applies function and returns object with same index as one being grouped
gb.filter() - Filter GroupBy object by a given function
gb.groups - Return dict whose keys are the unique groups, and values are axis labels belonging to each group

df[df.Length > 7] - Extract rows that meet logical criteria

df[(df[col] > 5) (df[col] < 7)] - Rows where 7 > col > 5
df.pivot_table(index=col1,values = [col2,col3],aggfunc=max) - Create a pivot table that groups by col1 and calculates the mean of col2 and col3
shift(1) - Copy with values shifted by 1
rank(method='dense') - Ranks with no gaps
rank(method='min') - Ranks. Ties get min rank
rank(pct=True) - Ranks rescaled to interval [0, 1]
rank(method='first') - Ranks. Ties go to first value
shift(-1) - Copy with values lagged by 1

8. Statistics

These can all be applied to a series as .
df.min() - Return minimum of every column
df.max() - Return maximum of every column
df.count() - Returns Series of row counts for every column
df.mean() - Return the mean of all columns
df.corr() - finds the correlation between columns in a DataFrame
df.median() - finds the median of each column
df.std() - finds the standard deviation of each column
df.cumsum() - Cumulative sum
df.cummax() - Cumulative max
df.cummin() - Cumulative min
df.cumprod() - Cumulative product

9. Combine Data Sets

df1.concat(df2) - Merge DataFrame or Series objects
pd.merge(df1, df2,how='left', on='x1') - Join matching rows from df2 to df1
pd.merge(df1, df2,how='right', on='x1') - Join matching rows from df1 to df2

pd.merge(df1, df2,how='inner', on='x1') - Join data. Retain only rows in both sets
pd.merge(df1, df2,how='outer', on='x1') - Join data. Retain all values, all rows
df1.join(df2,on=col1,how='inner') - SQL-style join the columns in df1 with the columns on df2 where the rows for col have identical values. how can be one of 'left', 'right', 'outer', 'inner'

---

10. Data Cleaning

df.rename(columns = 'y':'year') - Rename the columns of a DataFrame
pd.melt(df) - Gather columns into rows
df.pivot(columns='var', values='val') - Spread rows into columns
df.drop(columns=['Length','Height']) - Drop columns from DataFrame
df.dropna() - Drop rows with any column having NA/null data
df.fillna(value) - Replace all NA or null data with value
pd.isnull() - Checks for null Values, Returns Boolean Arrray
pd.notnull() - Opposite of s.isnull()
df.dropna(axis=1) - Drop all columns that contain null values
df.dropna(axis=1,thresh=n) - Drop all rows have have less than n non null values

---