Pandas Cheat Sheet

Pandas Basic commands:

Imports the following commands to start:

```
import pandas as pd
import numpy as np
```

Pandas version:

```
import pandas as pd
print(pd.__version__)
```

Key and Imports	
df	pandas DataFrame object
s	pandas Series object

Create Dataframe:

```
import pandas as pd

df = pd.DataFrame({'X':[78,85,96,80,86],
'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]});

print(df)
```

Copy

Sample Output:

```
    X   Y   Z
0  78  84  86
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

Create DataSeries:

```
import pandas as pd

s = pd.Series([2, 4, 6, 8, 10])

print(s)
```

Copy

Sample Output:

```
0      2
1      4
2      6
3      8
4     10
dtype: int64
```

Create Test Objects

pd.DataFrame(np.random.rand(20,5))	5 columns and 20 rows of random floats
pd.Series(my_list)	Create a series from an iterable my_list
df.index = pd.date_range('1900/1/30', periods=df.shape[0])	Add a date index

Viewing/Inspecting Data

df.head(n)	First n rows of the DataFrame
df.tail(n)	Last n rows of the DataFrame
df.shape	Number of rows and columns
df.info()	Index, Datatype and Memory information
df.describe()	Summary statistics for numerical columns
s.value_counts(dropna=False)	View unique values and counts
df.apply(pd.Series.value_counts)	Unique values and counts for all columns

Selection

df[col]	Returns column with label col as Series

df[[col1, col2]]	Returns columns as a new DataFrame
s.iloc[0]	Selection by position
s.loc['index_one']	Selection by index
df.iloc[0,:]	First row
df.iloc[0,0]	First element of first column

Data Cleaning

df.columns = ['a','b','c']	Rename columns
pd.isnull()	Checks for null Values, Returns Boolean Arrray
pd.notnull()	Opposite of pd.isnull()
df.dropna()	Drop all rows that contain null values
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
df.fillna(x)	Replace all null values with x
s.fillna(s.mean())	Replace all null values with the mean
s.astype(float)	Convert the datatype of the series to float
s.replace(1,'one')	Replace all values equal to 1 with 'one'

s.replace([2,3],['two', 'three'])	Replace all 2 with 'two' and 3 with 'three'
df.rename(columns=lambda x: x + 1)	Mass renaming of columns
df.rename(columns={'old_name': 'new_name'})	Selective renaming
df.set_index('column_one')	Change the index
df.rename(index=lambda x: x + 1)	Mass renaming of index

Filter, Sort, and Groupby

df[df[col] > 0.6]	Rows where the column col is greater than 0.6
df[(df[col] > 0.6) & (df[col] < 0.8)]	Rows where 0.8 > col > 0.6
df.sort_values(col1)	Sort values by col1 in ascending order
df.sort_values(col2,ascending=False)	Sort values by col2 in descending order.5
df.sort_values([col1,col2],ascending=[True,False])	Sort values by col1 in ascending order then col2 in descending order
df.groupby(col)	Returns a groupby object for values from one column
df.groupby([col1,col2])	Returns groupby object for values from multiple columns

df.groupby(col1)[col2]	Returns the mean of the values in col2, grouped by the values in col1
df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean)	Create a pivot table that groups by col1 and calculates the mean of col2 and col3
df.groupby(col1).agg(np.mean)	Find the average across all columns for every unique col1 group
df.apply(np.mean)	Apply the function np.mean() across each column
nf.apply(np.max,axis=1)	Apply the function np.max() across each row

Join/Combine

df1.append(df2)	Add the rows in df1 to the end of df2 (columns should be identical)
pd.concat([df1, df2],axis=1)	Add the columns in df1 to the end of df2 (rows should be identical)
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. The 'how' can be 'left', 'right', 'outer' or 'inner'

Statistics

df.describe()	Summary statistics for numerical columns
df.mean()	Returns the mean of all columns
df.corr()	Returns the correlation between columns in a DataFrame

df.count()	Returns the number of non-null values in each DataFrame column
df.max()	Returns the highest value in each column
df.min()	Returns the lowest value in each column
df.median()	Returns the median of each column
df.std()	Returns the standard deviation of each column

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/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

Exporting Data

df.to_csv(filename)	Write to a CSV file
df.to_excel(filename)	Write to an Excel file
df.to_sql(table_name, connection_object)	Write to a SQL table
df.to_json(filename)	Write to a file in JSON format