

PANDAS

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

Importing Data

| | |
|------------------------------------|-----------------|
| <code>pd.read_csv(filename)</code> | From a CSV file |
|------------------------------------|-----------------|

| | |
|--------------------------------------|---------------------------------------|
| <code>pd.read_table(filename)</code> | From a delimited text file (like TSV) |
|--------------------------------------|---------------------------------------|

| | |
|--------------------------------------|--------------------|
| <code>pd.read_excel(filename)</code> | From an Excel file |
|--------------------------------------|--------------------|

| | |
|--|--------------------------------|
| <code>pd.read_sql(query, connection_object)</code> | Read from a SQL table/database |
|--|--------------------------------|

| | |
|--|---|
| <code>pd.read_json(json_string)</code> | Read from a JSON formatted string, URL or file. |
|--|---|

| | |
|--------------------------------|--|
| <code>pd.read_html(url)</code> | Parses an html URL, string or file and extracts tables to a list of dataframes |
|--------------------------------|--|

| | |
|----------------------------------|---|
| <code>pd.read_clipboard()</code> | Takes the contents of your clipboard and passes it to <code>read_table()</code> |
|----------------------------------|---|

| | |
|---------------------------------|---|
| <code>pd.DataFrame(dict)</code> | From a dict, keys for columns names, values for data as lists |
|---------------------------------|---|

Exporting Data

| | |
|----------------------------------|---------------------|
| <code>df.to_csv(filename)</code> | Write to a CSV file |
|----------------------------------|---------------------|

| | |
|------------------------------------|------------------------|
| <code>df.to_excel(filename)</code> | Write to an Excel file |
|------------------------------------|------------------------|

| | |
|---|----------------------|
| <code>df.to_sql(table_name, connection_object)</code> | Write to a SQL table |
|---|----------------------|

| | |
|-----------------------------------|--------------------------------|
| <code>df.to_json(filename)</code> | Write to a file in JSON format |
|-----------------------------------|--------------------------------|

Create Test Objects

Useful for testing code segments

| | |
|---|--|
| <code>pd.DataFrame(np.random.rand(20,5))</code> | 5 columns and 20 rows of random floats |
|---|--|

| | |
|---------------------------------|---|
| <code>pd.Series(my_list)</code> | Create a series from an iterable <code>my_list</code> |
|---------------------------------|---|

| | |
|---|------------------|
| <code>df.index = pd.date_range('1900/1/30', periods=df.shape[0])</code> | Add a date index |
|---|------------------|

Viewing/Inspecting Data

| | |
|-------------------------|-------------------------------|
| <code>df.head(n)</code> | First n rows of the DataFrame |
|-------------------------|-------------------------------|

| | |
|-------------------------|------------------------------|
| <code>df.tail(n)</code> | Last n rows of the DataFrame |
|-------------------------|------------------------------|

| | |
|-------------------------|----------------------------|
| <code>df.shape()</code> | Number of rows and columns |
|-------------------------|----------------------------|

| | |
|------------------------|--|
| <code>df.info()</code> | Index, Datatype and Memory information |
|------------------------|--|

| | |
|----------------------------|--|
| <code>df.describe()</code> | Summary statistics for numerical columns |
|----------------------------|--|

| | |
|---|-------------------------------|
| <code>s.value_counts(dropna=False)</code> | View unique values and counts |
|---|-------------------------------|

| | |
|---|--|
| <code>df.apply(pd.Series.value_counts)</code> | Unique values and counts for all columns |
|---|--|

Selection

| | |
|----------------------|---|
| <code>df[col]</code> | Returns column with label col as Series |
|----------------------|---|

| | |
|-------------------------------|------------------------------------|
| <code>df[[col1, col2]]</code> | Returns columns as a new DataFrame |
|-------------------------------|------------------------------------|

| | |
|------------------------|-----------------------|
| <code>s.iloc[0]</code> | Selection by position |
|------------------------|-----------------------|

| | |
|---------------------------------|--------------------|
| <code>s.loc['index_one']</code> | Selection by index |
|---------------------------------|--------------------|

| | |
|---------------------------|-----------|
| <code>df.iloc[0,:]</code> | First row |
|---------------------------|-----------|

| | |
|---------------------------|-------------------------------|
| <code>df.iloc[0,0]</code> | First element of first column |
|---------------------------|-------------------------------|

Data Cleaning

| | |
|---|----------------|
| <code>df.columns = ['a','b','c']</code> | Rename columns |
|---|----------------|

| | |
|--------------------------|---|
| <code>pd.isnull()</code> | Checks for null Values, Returns Boolean Array |
|--------------------------|---|

| | |
|---------------------------|--------------------------------------|
| <code>pd.notnull()</code> | Opposite of <code>pd.isnull()</code> |
|---------------------------|--------------------------------------|

| | |
|--------------------------|--|
| <code>df.dropna()</code> | Drop all rows that contain null values |
|--------------------------|--|

| | |
|--------------------------------|---|
| <code>df.dropna(axis=1)</code> | Drop all columns that contain null values |
|--------------------------------|---|

| | |
|--|---|
| <code>df.dropna(axis=1, thresh=n)</code> | Drop all columns have less than n non-null values |
|--|---|

| | |
|---------------------------|--------------------------------|
| <code>df.fillna(x)</code> | Replace all null values with x |
|---------------------------|--------------------------------|

| | |
|---------------------------------|---|
| <code>s.fillna(s.mean())</code> | Replace all null values with the mean (mean can be replaced with almost any function from the statistics section) |
|---------------------------------|---|

| | |
|------------------------------|---|
| <code>s.astype(float)</code> | Convert the datatype of the series to float |
|------------------------------|---|

| | |
|---------------------------------|--|
| <code>s.replace(1,'one')</code> | Replace all values equal to 1 with 'one' |
|---------------------------------|--|

| | |
|---|---|
| <code>s.replace([1,3],['one','three'])</code> | Replace all 1 with 'one' and 3 with 'three' |
|---|---|

| | |
|---|--------------------------|
| <code>df.rename(columns=lambda x: x + 1)</code> | Mass renaming of columns |
|---|--------------------------|

| | |
|--|--------------------|
| <code>df.rename(columns={'old_name': 'new_name'})</code> | Selective renaming |
|--|--------------------|

| | |
|---|------------------|
| <code>df.set_index('column_one')</code> | Change the index |
|---|------------------|

| | |
|---|------------------------|
| <code>df.rename(index=lambda x: x + 1)</code> | Mass renaming of index |
|---|------------------------|

Filter, Sort & Groupby

| | |
|-----------------------------------|--|
| <code>df[df[col] > 0.5]</code> | Rows where the column <code>col</code> is greater than 0.5 |
|-----------------------------------|--|

| | |
|--|---|
| <code>df[(df[col] > 0.5) & (df[col] < 0.7)]</code> | Rows where <code>0.7 > col > 0.5</code> |
|--|---|

| | |
|-----------------------------------|---|
| <code>df.sort_values(col1)</code> | Sort values by <code>col1</code> in ascending order |
|-----------------------------------|---|

| | |
|---|--|
| <code>df.sort_values(col2,ascending=False)</code> | Sort values by <code>col2</code> in descending order |
|---|--|

| | |
|---|--|
| <code>df.sort_values([col1,col2],ascending=[True,False])</code> | Sort values by col1 in ascending order then col2 in descending order |
|---|--|

| | |
|------------------------------|---|
| <code>df.groupby(col)</code> | Returns a groupby object for values from one column |
|------------------------------|---|

| | |
|--------------------------------------|---|
| <code>df.groupby([col1,col2])</code> | Returns groupby object for values from multiple columns |
|--------------------------------------|---|

| | |
|-------------------------------------|--|
| <code>df.groupby(col1)[col2]</code> | Returns the mean of the values in col2 , grouped by the values in col1 (mean can be replaced with almost any function from the statistics section) |
|-------------------------------------|--|

| | |
|---|--|
| <code>df.pivot_table(index=col1,values=[col2,col3],aggfunc=mean)</code> | Create a pivot table that groups by col1 and calculates the mean of col2 and col3 |
|---|--|

| | |
|--|--|
| <code>df.groupby(col1).agg(np.mean)</code> | Find the average across all columns for every unique col1 group |
|--|--|

| | |
|--------------------------------|--|
| <code>df.apply(np.mean)</code> | Apply the function np.mean() across each column |
|--------------------------------|--|

| | |
|--------------------------------------|--|
| <code>nf.apply(np.max,axis=1)</code> | Apply the function np.max() across each row |
|--------------------------------------|--|

Join/Combine

| | |
|------------------------------|---|
| <code>df1.append(df2)</code> | Add the rows in df1 to the end of df2 (columns should be identical) |
|------------------------------|---|

| | |
|---|---|
| <code>df.concat([df1, df2],axis=1)</code> | Add the columns in df1 to the end of df2 (rows should be identical) |
|---|---|

| | |
|---|--|
| <code>df1.join(df2,on =col1,how='inner')</code> | 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' |
|---|--|

Statistics

These can all be applied to a series as well.

| | |
|----------------------------|--|
| <code>df.describe()</code> | Summary statistics for numerical columns |
|----------------------------|--|

| | |
|------------------------|---------------------------------|
| <code>df.mean()</code> | Returns the mean of all columns |
|------------------------|---------------------------------|

| | |
|------------------------|--|
| <code>df.corr()</code> | Returns the correlation between columns in a DataFrame |
|------------------------|--|

| | |
|-------------------------|--|
| <code>df.count()</code> | Returns the number of non-null values in each DataFrame column |
|-------------------------|--|

| | |
|-----------------------|--|
| <code>df.max()</code> | Returns the highest value in each column |
|-----------------------|--|

| | |
|-----------------------|---|
| <code>df.min()</code> | Returns the lowest value in each column |
|-----------------------|---|

| | |
|--------------------------|-----------------------------------|
| <code>df.median()</code> | Returns the median of each column |
|--------------------------|-----------------------------------|

| | |
|-----------------------|---|
| <code>df.std()</code> | Returns the standard deviation of each column |
|-----------------------|---|

