Python Pandas Cheat Sheet

Loading/exporting a data set

path_to_file: string indicating the path to the file, e.g., 'data/results.csv'

`df = pd.read_csv(path_to_file)` — read a CSV file

`df = pd.read_excel(path_to_file)` — read an Excel file

`df = pd.read_html(path_to_file)` — parses HTML to find all tables

`df.to_csv(path_to_file)` — creates CSV of the data frame

Examining the data

`df.head(n)` — returns first n rows

`df.tail(n)` — returns last n rows

`df.describe()` — returns summary statistics for each numerical column

`df['State'].unique()` — returns unique values for the column

`df.columns` — returns column names

`df.shape` — returns the number of rows and columns

Selecting and filtering

SELECTING COLUMNS

`df['State']` — selects 'State' column
`df[['State', 'Population']]` — selects 'State' and 'Population' column

SELECTING BY LABEL

`df.loc['a']` — selects row by index label
`df.loc['a', 'State']` — selects single value of row 'a' and column 'State'

SELECTING BY POSITION

`df.iloc[0]` — selects rows in position 0
`df.iloc[0, 0]` — selects single value by position at row 0 and column 0

FILTERING

`df[df['Population'] > 20000000]]` — filter out rows not meeting the condition

`df.query("Population > 20000000")` — filter out rows not meeting the condition

Statistical operations

can be applied to both data frames and series/column

`df['Population'].sum()` — sum of all values of a column

`df.sum()` — sum for all numerical columns

`df.mean()` — mean

`df.std()` — standard deviation

`df.min()` — minimum value

`df.count()` — count of values, excludes missing values

`df.max()` — maximum value

`df['Population'].apply(func)` — apply func to each value of column

Data cleaning and modifications

`df['State'].isnull()` — returns True/False for rows with missing values

`df.dropna(axis=0)` — drop rows containing missing values

`df.dropna(axis=1)` — drop columns containing missing values

`df.fillna(0)` — fill in missing values, here filled with 0

`df.sort_values('Population', ascending=True)` — sort rows by a column's values

`df.set_index('State')` — changes index to a specified column

`df.reset_index()` — makes the current index a column

`df.rename(columns={'Population'='Pop.'})` — renames columns

Grouping and aggregation

`grouped = df.groupby(by='col1')` — create grouped by object

`grouped['col2'].mean()` — mean value of 'col2' for each group

`grouped.agg({'col2': np.mean, 'col3': [np.mean, np.std]})` — apply different functions to different columns

`grouped.apply(func)` — apply func to each group

Merging data frames

There are several ways to merge two data frames, depending on the value of `how`. The resulting indices are integers starting with zero.

```
df1.merge(df2, how=method, on='State')
```

	State	Capital	Population
a	Texas	Austin	28700000
b	New York	Albany	19540000
c	Washington	Olympia	7536000

Data frame `df1`

	State	Highest Point
x	Washington	Mount Rainier
y	New York	Mount Marcy
z	Nebraska	Panorama Point

Data frame `df2`

`how='left'`

	State	Capital	Population	Highest Point
0	Texas	Austin	28700000	NaN
1	New York	Albany	19540000	Mount Marcy
2	Washington	Olympia	7536000	Mount Rainier

`how='right'`

	State	Capital	Population	Highest Point
0	New York	Albany	19540000	Mount Marcy
1	Washington	Olympia	7536000	Mount Rainier
2	Nebraska	NaN	NaN	Panorama Point

`how='inner'`

	State	Capital	Population	Highest Point
0	New York	Albany	19540000	Mount Marcy
1	Washington	Olympia	7536000	Mount Rainier

`how='outer'`

	State	Capital	Population	Highest Point
0	Texas	Austin	28700000	NaN
1	New York	Albany	19540000	Mount Marcy
2	Washington	Olympia	7536000	Mount Rainier
3	Nebraska	NaN	NaN	Panorama Point