

Creating, Reading, Writing

<code>df = pd.DataFrame({"col0": [val0, val1], "col1": [val0, val1]}, index=[0, 1])</code>	Create a dataframe
<code>series = pd.Series(["val0", "val1", "val2"], index=[0, 1, 2, 3], name="name")</code>	Create a series
<code>read = pd.read_csv("../folder/folder/file.csv", index_col=0)</code>	Read a csv
<code>save.to_csv("file.csv")</code>	Save an existing dataframe as a csv

Indexing, Selecting, Assigning

<code>table.head</code>	Show first 5 rows of a dataframe
<code>table["col"]</code>	Select the col from table
<code>table.col.iloc[0]</code>	Select 1 st value of a col from table
<code>table.iloc[0]</code>	Select 1 st row of data from table
<code>table.col.iloc[:10]</code>	Select 1 st 10 values from col in table (index-based select)
<code>table.col.loc[:10]</code>	Select 1 st 10 values from col in table (label-based select)
<code>table.loc[indices, cols]</code>	Select certain rows from certain cols
<code>table[table.col == 'val']</code>	Select cols have a certain val (conditional select)
<code>table.col.isin(['val1', 'val2'])</code>	Select cols have certain vals (conditional select)

Summary Functions & Maps

<code>table.col.describe()</code>	Get high-lvl summary of given col's attributes
<code>table.col.mean()</code>	Get mean of a col with numerical vals
<code>table.col.unique()</code>	Get each unique val of a col w/ no dupes
<code>table.col.value_counts()</code>	Get frequency of each val in col

Summary Functions & Maps (cont)

<code>table.col.map(lambda p: p - s)</code>	Map function to remap a Series of point vals (p) by using a transformation (p-s) -> returns new Series
<code>table.apply(func, axis='col')</code>	Apply function to transform entire df by calling custom method (func taking a row) on each row (umns')

Grouping & Sorting

<code>table.groupby('col').col.count()</code>	Group data w/ same vals in the given col -> count frequency of given col (same as <code>value_counts()</code>)
<code>table.groupby('col').size()</code>	Same as above
<code>table.groupby('col').apply(lambda df: df.title.iloc[0])</code>	Select name (title) of the 1 st thing in col
<code>table.col.idxmax()</code>	Get index of max val in col
<code>table.groupby(['col0', 'col1']).col1.agg([f1, f2, f3])</code>	<code>agg()</code> runs diff. funcs. simultaneously on a df
<code>table.groupby(['col0', 'col1']).col2.agg([len])</code>	Multi-index output has tiered structure. Require 2 levels of labels to retrieve a val
<code>df.reset_index()</code>	Muti-index method used to converting back to regular index
<code>df.sort_values(by='col')</code>	Sort rows of data by vals in col (ascending)
<code>df.sort_values(by='col', ascending=False)</code>	Sort rows of data by vals in col (descending)
<code>df.sort_values(by=['col0', 'col1'])</code>	Sort rows by more than 1 col at a time
<code>df.sort_index()</code>	Sort rows by index (default order; ascending)



By blueberryshortcake

Not published yet.
Last updated 25th June, 2024.
Page 1 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

Data Types & Missing Values

<code>table.col.dtype</code>	Get data type of a col
<code>table.dtypes</code>	Get data types of each col in table
<code>table.col.astype('datatype')</code>	Convert col to datatype if allowed (e.g, int64 -> float64)
<code>table.index.dtype</code>	Number indices are int64
<code>table[pd.isnull(table.col)]</code>	Select NaN entries in a col
<code>table.col.fillna("filler")</code>	Replace all NaN vals in a col with a sentinel val ("Unknown", "Undisclosed", "Invalid") or non-null val
<code>table.col.replace("init_val", "new_val")</code>	Replace, in col, all existing vals with new_vals

Renaming & Combining

<code>table.rename(columns={'init': 'new'})</code>	Rename col or index col names
<code>table.rename(index={0: 'firstEntry', 1: 'secondEntry'})</code>	Rename index or col vals by specifying an index or col param
<code>table.rename_axis("name", axis='rows').rename_axis("name1", axis='columns')</code>	Rename row index &/or col index
<code>pd.concat(list, of, els)</code>	Smush together the list of elements along an axis
<code>left.join(right, lsuffix='strL', rsuffix='strR')</code>	Combine diff df objects that have an index in common. left and right are df.s defined beforehand



By blueberryshortcake

Not published yet.
Last updated 25th June, 2024.
Page 2 of 2.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>