Pandas: 3rd lesson – Renaming & Combining

Renaming:

* The first function we'll introduce here is rename(), which lets you change index names and/or column names. For example, to change the points column in our dataset to score, we would do:

import pandas as pd

pd.set\_option('display.max\_rows', 5)

reviews = pd.read\_csv("../input/wine-reviews/winemag-data-130k-v2.csv", index\_col=0)

reviews.rename(columns={'points': 'score'})

* rename() lets you rename index or column values by specifying a index or column keyword parameter, respectively. It supports a variety of input formats, but usually a Python dictionary is the most convenient. Here is an example using it to rename some elements of the index:

reviews.rename(index={0: 'firstEntry', 1: 'secondEntry'})

* You'll probably rename columns very often, but rename index values very rarely. For that, set\_index() is usually more convenient. Both the row index and the column index can have their own name attribute. The complimentary rename\_axis() method may be used to change these names. For example:

reviews.rename\_axis("wines", axis='rows').rename\_axis("fields", axis='columns')

Combining:

* When performing operations on a dataset, we will sometimes need to combine different DataFrames and/or Series in non-trivial ways. Pandas has three core methods for doing this. In order of increasing complexity, these are concat(), join(), and merge(). Most of what merge() can do can also be done more simply with join(), so we will omit it and focus on the first two functions here. The simplest combining method is concat(). Given a list of elements, this function will smush those elements together along an axis.
* This is useful when we have data in different DataFrame or Series objects but having the same fields (columns). One example: the YouTube Videos dataset, which splits the data up based on country of origin (e.g. Canada and the UK, in this example). If we want to study multiple countries simultaneously, we can use concat() to smush them together:

canadian\_youtube = pd.read\_csv("../input/youtube-new/CAvideos.csv")

british\_youtube = pd.read\_csv("../input/youtube-new/GBvideos.csv")

pd.concat([canadian\_youtube, british\_youtube])

* The middlemost combiner in terms of complexity is join(). join() lets you combine different DataFrame objects which have an index in common. For example, to pull down videos that happened to be trending on the same day in both Canada and the UK, we could do the following:

left = canadian\_youtube.set\_index(['title', 'trending\_date'])

right = british\_youtube.set\_index(['title', 'trending\_date'])

left.join(right, lsuffix='\_CAN', rsuffix='\_UK')

* The lsuffix and rsuffix parameters are necessary here because the data has the same column names in both British and Canadian datasets. If this wasn't true (because, say, we'd renamed them beforehand) we wouldn't need them.