**Restoring the index order**

Continuing from the previous exercise, you will now use .swaplevel(0, 1) to flip the index levels. Note they won't be sorted. To sort them, you will have to follow up with a .sort\_index(). You will then obtain the original DataFrame. Note that an unsorted index leads to slicing failures.

To begin, print both users and bycity in the IPython Shell. The goal here is to convert bycity back to something that looks like users.

**INSTRUCTIONS**

* Define a DataFrame newusers with the 'city' level stacked back into the index of bycity.
* Swap the levels of the index of newusers.
* Print newusers and verify that the index is not sorted.
* Sort the index of newusers.
* Print newusers and verify that the index is now sorted.
* Assert that newusers equals users.

# Stack 'city' back into the index of bycity: newusers

newusers = bycity.stack(level='city')

# Swap the levels of the index of newusers: newusers

newusers = newusers.swaplevel(0,1)

# Print newusers and verify that the index is not sorted

print(newusers)

# Sort the index of newusers: newusers

newusers = newusers.sort\_index()

# Print newusers and verify that the index is now sorted

print(newusers)

# Verify that the new DataFrame is equal to the original

print(newusers.equals(users))