**Filtering and grouping with .map()**

You have seen how to group by a column, or by multiple columns. Sometimes, you may instead want to group by a function/transformation of a column. The key here is that the Series is indexed the same way as the DataFrame. You can also mix and match column grouping with Series grouping.

In this exercise your job is to investigate survival rates of passengers on the Titanic by 'age' and 'pclass'. In particular, the goal is to find out what fraction of children under 10 survived in each 'pclass'. You'll do this by first creating a boolean array where True is passengers under 10 years old and False is passengers over 10. You'll use .map() to change these values to strings.

Finally, you'll group by the under 10 series and the 'pclass' column and aggregate the 'survived' column. The 'survived'column has the value 1 if the passenger survived and 0 otherwise. The mean of the 'survived' column is the fraction of passengers who lived.

The DataFrame has been pre-loaded for you as titanic.

**INSTRUCTIONS**

* Create a Boolean Series of titanic['age'] < 10 and call .map with {True:'under 10', False:'over 10'}.
* Group titanic by the under10 Series and then compute and print the mean of the 'survived' column.
* Group titanic by the under10 Series as well as the 'pclass' column and then compute and print the mean of the 'survived' column.

# Create the Boolean Series: under10

under10 = (titanic['age'] < 10).map({True:'under 10', False:'over 10'})

# Group by under10 and compute the survival rate

survived\_mean\_1 = titanic.groupby(under10)['survived'].mean()

print(survived\_mean\_1)

# Group by under10 and pclass and compute the survival rate

survived\_mean\_2 = titanic.groupby([under10, 'pclass'])['survived'].mean()

print(survived\_mean\_2)