**Grouping and aggregating**

The Olympic medal data for the following exercises comes from [**The Guardian**](https://www.theguardian.com/sport/datablog/2012/jun/25/olympic-medal-winner-list-data). It comprises records of all events held at the Olympic games between 1896 and 2012.

Use Olympic\_medals.csv file & load it into a dataframe named medals.You now want to find the total number of medals awarded to the USA per edition. To do this, filter the 'USA' rows and use the groupby() function to put the 'Edition' column on the index:

USA\_edition\_grouped = medals.loc[medals.NOC == 'USA'].groupby('Edition')

Given the goal of finding the total number of USA medals awarded per edition, what column should you select and which aggregation method should you use?

##### INSTRUCTIONS

USA\_edition\_grouped['City'].mean()

USA\_edition\_grouped['Athlete'].sum()

USA\_edition\_grouped['Medal'].count()

USA\_edition\_grouped['Gender'].first()

# Using .value\_counts() for ranking

For this exercise, you will use the pandas Series method .value\_counts() to determine the top 15 countries ranked by total number of medals.

Notice that .value\_counts() sorts by values by default. The result is returned as a Series of counts indexed by unique entries from the original Series with values (counts) ranked in descending order.

Use the DataFrame medals.

##### INSTRUCTIONS

* Extract the 'NOC' column from the DataFrame medals and assign the result to country\_names. Notice that this Series has repeated entries for every medal (of *any* type) a country has won in any Edition of the Olympics.
* Create a Series medal\_counts by applying .value\_counts()to the Series country\_names.
* Print the top 15 countries ranked by total number of medals won.

# Using .pivot\_table() to count medals by type

Rather than ranking countries by total medals won and showing that list, you may want to see a bit more detail. You can use a pivot table to compute how many separate bronze, silver and gold medals each country won. That pivot table can then be used to repeat the previous computation to rank by total medals won.

In this exercise, you will use .pivot\_table() first to aggregate the total medals by type. Then, you can use .sum() along the columns of the pivot table to produce a new column. When the modified pivot table is sorted by the total medals column, you can display the results from the last exercise with a bit more detail.

##### INSTRUCTIONS

* Construct a pivot table counted from the DataFrame medals aggregating by count. Use 'NOC' as the index, 'Athlete' for the values, and 'Medal' for the columns.
* Modify the DataFrame counted by adding a column counted['totals']. The new column 'totals' should contain the result of taking the sum along the columns (i.e., use .sum(axis='columns')).
* Overwrite the DataFrame counted by sorting it with the .sort\_values() method. Specify the keyword argument ascending=False.
* Print the first 15 rows of counted using .head(15).

# Applying .drop\_duplicates()

What could be the difference between the 'Event\_gender' and 'Gender' columns? You should be able to evaluate your guess by looking at the unique values of the pairs (Event\_gender, Gender) in the data. In particular, you should not see something like (Event\_gender='M', Gender='Women'). However, you will see that, strangely enough, there is an observation with (Event\_gender='W', Gender='Men').

The duplicates can be dropped using the .drop\_duplicates()method, leaving behind the unique observations. The DataFrame has been loaded as medals.

##### INSTRUCTIONS

* Select the columns 'Event\_gender' and 'Gender'.
* Create a dataframe ev\_gen\_uniques containing the unique pairs contained in ev\_gen.
* Print ev\_gen\_uniques.

# Finding possible errors with .groupby()

You will now use .groupby() to continue your exploration. Your job is to group by 'Event\_gender' and 'Gender' and count the rows.

You will see that there is only one suspicious row: This is likely a data error.

Use the DataFrame  medals.

##### INSTRUCTIONS

* Group medals by 'Event\_gender' and 'Gender'.
* Create a medal\_count\_by\_gender DataFrame with a group count using the .count() method.
* Print medal\_count\_by\_gender.

# Locating suspicious data

You will now inspect the suspect record by locating the offending row.

You will see that, according to the data, Joyce Chepchumba was a man that won a medal in a women's event. That is a data error as you can confirm with a web search.

##### INSTRUCTIONS

* Create a Boolean Series with a condition that captures the only row that has medals.Event\_gender == 'W' and medals.Gender == 'Men'. Be sure to use the & operator.
* Use the Boolean Series to create a DataFrame called suspectwith the suspicious row.
* Print suspect.

# Using .nunique() to rank by distinct sports

You may want to know which countries won medals in the most distinct sports. The .nunique() method is the principal aggregation here. Given a categorical Series S, S.nunique()returns the number of distinct categories.

##### INSTRUCTIONS

* Group medals by 'NOC'.
* Compute the number of distinct sports in which each country won medals. To do this, select the 'Sport' column from country\_grouped and apply .nunique().
* Sort Nsports in descending order with .sort\_values() and ascending=False.
* Print the first 15 rows of Nsports.

# Counting USA vs. USSR Cold War Olympic Sports

The Olympic competitions between 1952 and 1988 took place during the height of the Cold War between the United States of America (USA) & the Union of Soviet Socialist Republics (USSR). Your goal in this exercise is to aggregate the number of distinct sports in which the USA and the USSR won medals during the Cold War years.

The construction is mostly the same as in the preceding exercise. There is an additional filtering stage beforehand in which you reduce the original DataFrame medals by extracting data from the Cold War period that applies only to the US or to the USSR. Select country codes 'USA' & 'URS' in the DataFrame medals

##### INSTRUCTIONS

* Create a Boolean Series called during\_cold\_war by extracting all rows from medals for which the 'Edition' is >= 1952 and <= 1988.
* Create a Boolean Series called is\_usa\_urs by extracting rows from medals for which 'NOC' is either 'USA' or 'URS'.
* Filter the medals DataFrame using during\_cold\_war and is\_usa\_urs to create a new DataFrame called cold\_war\_medals.
* Group cold\_war\_medals by 'NOC'.
* Create a Series Nsports from country\_grouped using indexing & chained methods:
  + Extract the column 'Sport'.
  + Use .nunique() to get the number of unique elements in each group;
  + Apply .sort\_values(ascending=False) to rearrange the Series.
* Print the final Series Nsports.

# Counting USA vs. USSR Cold War Olympic Medals

For this exercise, you want to see which country, the USA or the USSR, won the most medals consistently over the Cold War period.

There are several steps involved in carrying out this computation.

* You'll need a pivot table with years ('Edition') on the index and countries ('NOC') on the columns. The entries will be the total number of medals each country won that year. If the country won no medals in a given edition, expect a NaN in that entry of the pivot table.
* You'll need to slice the Cold War period and subset the 'USA' and 'URS' columns.
* You'll need to make a Series from this slice of the pivot table that tells which country won the most medals in that edition using .idxmax(axis='columns'). If .max() returns the maximum value of Series or 1D array, .idxmax() returns the index of the maximizing element. The argument axis=columns or axis=1 is required because, by default, this aggregation would be done along columns for a DataFrame.
* The final Series contains either 'USA' or 'URS'according to which country won the most medals in each Olympic edition. You can use .value\_counts() to count the number of occurrences of each.

##### INSTRUCTIONS

* Construct medals\_won\_by\_country using medals.pivot\_table().
  + The index should the years ('Edition') & the columns should be country ('NOC')
  + the values should be 'Athlete' (which captures every medal regardless of kind) & the aggregation method should be 'count' (which captures the total number of medals won).
* Create cold\_war\_usa\_usr\_medals by slicing the pivot table medals\_won\_by\_country. Your slice should contain the editions from years 1952:1988 and only the columns 'USA'& 'URS' from the pivot table.
* Create the Series most\_medals by applying the .idxmax()method to cold\_war\_usa\_usr\_medals. Be sure to use axis='columns'.
* Print the result of applying .value\_counts() to most\_medals. The result reported gives the number of times each of the USA or the USSR won more Olympic medals in total than the other between 1952 and 1988.

# Visualizing USA Medal Counts by Edition: Line Plot

Your job in this exercise is to visualize the medal counts by 'Edition' for the USA. The DataFrame has been pre-loaded for you as medals.

##### INSTRUCTIONS

* Create a DataFrame usa with data only for the USA.
* Group usa such that ['Edition', 'Medal'] is the index. Aggregate the count over 'Athlete'.
* Use .unstack() with level='Medal' to reshape the DataFrame usa\_medals\_by\_year.
* Construct a line plot from the final DataFrame usa\_medals\_by\_year.

# Edition: Area Plot

As in the previous exercise, your job in this exercise is to visualize the medal counts by 'Edition' for the USA. This time, you will use an area plot to see the breakdown better.

##### INSTRUCTIONS

Use the usa\_medals\_by\_year you crested in the previous exercise.Then plot it using plot.area().

# Visualizing USA Medal Counts by Edition: Area Plot with Ordered Medals

You may have noticed that the medals are ordered according to a lexicographic (dictionary) ordering: Bronze < Gold < Silver. However, you would prefer an ordering consistent with the Olympic rules: Bronze < Silver < Gold.

You can achieve this using Categorical types. In this final exercise, after redefining the 'Medal' column of the DataFrame medals, you will repeat the area plot from the previous exercise to see the new ordering.

##### INSTRUCTIONS

* Redefine the 'Medal' column of the DataFrame medals as an *ordered categorical*. To do this, use pd.Categorical() with three keyword arguments:
  + values = medals.Medal.
  + categories=['Bronze', 'Silver', 'Gold'].
  + ordered=True.
  + After this, you can verify that the type has changed using medals.info().
* Plot the final DataFrame usa\_medals\_by\_year as an *area plot*. This has been done for you, so hit 'Submit Answer' to see how the plot has changed!