**Loading Olympic edition DataFrame**

1. In this chapter, you'll be using [**The Guardian's Olympic medal dataset**](https://www.theguardian.com/sport/datablog/2012/jun/25/olympic-medal-winner-list-data).
2. Your first task here is to prepare a DataFrame editions from a *tab-separated values* (TSV) file.
3. Initially, editions has 26 rows (one for each Olympic *edition*, i.e., a year in which the Olympics was held) and 7 columns: 'Edition', 'Bronze', 'Gold', 'Silver', 'Grand Total', 'City', and 'Country'.For the analysis that follows, you won't need the overall medal counts, so you want to keep only the useful columns from editions: 'Edition', 'Grand Total', City, and Country.

**Input Sample:**

**Edition** **Bronze** **Gold** **Silver** **Grand Total**  **City**  **Country**

**0** | 1896 | 40 | 64 | 47 | 151 | Athens | Greece

**1** | 1900 | 142 | 178 | 192 | 512 | Paris | France

**2** | 1904 | 123 | 188 | 159 | 470 | St. Louis |UnitedStates

**3** | 1908 | 211 | 311 | 282 | 804 | London UnitedKingdom

**4** | 1912 | 284 | 301 | 300 | 885 | Stockholm | Sweden

##### INSTRUCTIONS

* Read file\_path into a DataFrame called editions. The identifier file\_path has been pre-defined with the filename 'SOM\_EDITIONS.tsv'. You'll have to use the option sep='\t' because the file uses tabs to delimit fields (pd.read\_csv() expects commas by default).
* Select only the columns 'Edition', 'Grand Total', 'City', and 'Country' from editions.
* Save the modified dataset (i.e editions ) in following directory:   **"/code/ex1a\_eval.csv"**

**Sample Output:**

**Edition** **Grand Total**  **City** **Country**

**0** | 1896 | 151 | Athens | Greece

**1** | 1900 | 512 | Paris | France

**2** | 1904 | 470 | St. Louis | United States

**3** | 1908 | 804 | London | United Kingdom

**4** | 1912 | 885 | Stockholm | Sweden

**5** | 1920 | 1298 | Antwerp | Belgium

# Loading IOC codes DataFrame

1. Your task here is to prepare a DataFrame ioc\_codes from a comma-separated values (CSV) file.
2. Initially, ioc\_codes has 200 rows (one for each country) and columns: 'Country', 'NOC', & 'ISO code'.
3. For the analysis that follows, you want to keep only the useful columns from ioc\_codes: 'Country' and 'NOC' (the column 'NOC'contains three-letter codes representing each country).

**Input sample:**

**Country**  **NOC** **ISO code**

**0** | Afghanistan | AFG | AF

**1** | Albania | ALB | AL

**2** | Algeria | ALG | DZ

**3** | American Samoa\* | ASA | AS

**4** | Andorra | AND | AD

##### INSTRUCTIONS

* Read file\_path into a DataFrame called ioc\_codes. The identifier file\_path has been pre-defined with the filename  IOC\_COUNTRY\_CODES.csv'.
* Select only the columns 'Country' and 'NOC' from ioc\_codes.
* Print the modified ioc\_codes DataFrame

# Building medals DataFrame

1. Here, you'll start with the DataFrame editions from the previous exercise.
2. You have a sequence of files summer\_1896.csv, summer\_1900.csv, ..., summer\_2008.csv, one for each Olympic edition (year).You will build up a dictionary medals\_dict with the Olympic editions (years) as keys and DataFrames as values.
3. The dictionary is built up inside a loop over the year of each Olympic edition (from the Index of editions).Once the dictionary of DataFrames is built up, you will combine the DataFrames using pd.concat()

##### INSTRUCTIONS

* Within the for loop:
  + Create the file path
  + Read file\_path into a DataFrame. Assign the result to the yearkey of medals\_dict.
  + Select only the columns 'Athlete', 'NOC', and 'Medal' from medals\_dict[year].
  + Create a *new* column called 'Edition' in the DataFrame medals\_dict[year] whose entries are *all* year.
* Concatenate the dictionary of DataFrames medals\_dict into a DataFame called medals. Specify the keyword argument ignore\_index=True to prevent repeated integer indices.

Print the first and last 5 rows of medals.

# Counting medals by country/edition in a pivot table

1. Here, you'll start with the concatenated DataFrame medals from the previous exercise. You can construct a pivot table to see the number of medals each country won in each year. The result is a new DataFrame with the Olympic edition on the Index and with 138 country NOC codes as columns.

##### INSTRUCTIONS

* Construct a pivot table from the DataFrame medals, aggregating by count (by specifying the aggfunc parameter). Use 'Edition' as the Index, 'Athlete' for the values, and 'NOC' for the columns.

Save the output by using following command in ex1d\_eval.csv file **medal\_counts.to\_csv("/code/ex1d\_eval.csv")**

# Computing fraction of medals per Olympic edition

1. In this exercise, you'll start with the DataFrames editions, medals, & medal\_counts from prior exercises.
2. You can extract a Series with the total number of medals awarded in each Olympic edition.The DataFrame medal\_counts can be divided row-wise by the total number of medals awarded each edition; the method .divide()performs the broadcast as you require.

##### INSTRUCTIONS

* Set the index of the DataFrame editions to be 'Edition' (using the method .set\_index()). Save the result as totals.
* Extract the 'Grand Total' column from totals and assign the result back to totals.
* Divide the DataFrame medal\_counts by totals along each row. You will have to use the .divide() method with the option axis='rows'. Assign the result to fractions.

# Computing percentage change in fraction of medals won

1. Here, you'll start with the DataFrames editions, medals, medal\_counts, & fractions from prior exercises.To see if there is a host country advantage, you first want to see how the fraction of medals won changes from edition to edition.
2. The expanding mean provides a way to see this down each column. It is the value of the mean with all the data available up to that point in time.

##### INSTRUCTIONS

* Create mean\_fractions by chaining the methods .expanding().mean() to fractions.
* Compute the percentage change in mean\_fractions down each column by applying .pct\_change() and multiplying by 100. Assign the result to fractions\_change.
* Reset the index of fractions\_change using the .reset\_index()method. This will make 'Edition' an ordinary column.
* Save the output by using following command in ex1f\_eval.csv file **fractions\_change.to\_csv("/code/ex1f\_eval.csv")**

# Building hosts DataFrame

1. Your task here is to prepare a DataFrame hosts by left joining editions and ioc\_codes. Once created, you will subset the Edition and NOC columns and set Edition as the Index. There are some missing NOC values; you will set those explicitly.Finally, you'll reset the Index & print the final DataFrame.

##### INSTRUCTIONS

* Create the DataFrame hosts by doing a *left join* on DataFrames editions and ioc\_codes (using pd.merge()).
* Clean up hosts by subsetting and setting the Index.
  + Extract the columns 'Edition' and 'NOC'.
  + Set 'Edition' column as the Index.
* Use the .loc[] accessor to find and assign the missing values to the 'NOC' column in hosts. This has been done for you.
* Reset the index of hosts using .reset\_index

Save the output by using following command in **ex1g\_eval.csv** file

**hosts.to\_csv("/code/ex1g\_eval.csv")**

# Reshaping for analysis

1. This exercise starts off with fractions\_change and hosts .Your task here is to reshape the fractions\_change DataFrame for later analysis. Initially, fractions\_change is a wide DataFrame of 26 rows (one for each Olympic edition) and 139 columns (one for the edition and 138 for the competing countries).

On reshaping with pd.melt(), as you will see, the result is a tall DataFrame with 3588 rows and 3 columns that summarizes the fractional change in the expanding mean of the percentage of medals won for each country in blocks.

##### INSTRUCTIONS

* Create a DataFrame reshaped by reshaping the DataFrame fractions\_change with pd.melt().
* You'll need to use the keyword argument id\_vars='Edition' to set the identifier variable.
* You'll also need to use the keyword argument value\_name='Change' to set the measured variables.
* Print the shape of the DataFrames reshaped and fractions\_change. Create a DataFrame chn by extracting all the rows from reshaped in which the three letter code for each country ('NOC') is 'CHN'.
* Print the last 5 rows of the DataFrame chn using the .tail()method

# Merging to compute influence

1. This exercise starts off with the DataFrames reshaped and hosts . Your task is to merge the two DataFrames and tidy the result. The end result is a DataFrame summarizing the fractional change in the expanding mean of the percentage of medals won for the host country in each Olympic edition.

##### INSTRUCTIONS

* Merge reshaped and hosts using an inner join. Remember, how='inner' is the default behavior for pd.merge().
* Print the first 5 rows of the DataFrame merged. This has been done for you. You should see that the rows are jumbled chronologically.
* Set the index of merged to be 'Edition' and sort the index.
* Save the output by using following command in **ex1i\_eval.csv**  **influence.to\_csv("/code/ex1i\_eval.csv")**

.

# Plotting influence of host country

This final exercise starts off with the DataFrames influence and editions.Your job is to plot the influence of being a host country.

##### INSTRUCTIONS

* Create a Series called change by extracting the 'Change' column from influence.
* Create a bar plot of change using the .plot() method with kind='bar'. Save the result as ax to permit further customization.
* Customize the bar plot of change to improve readability:
* Apply the method .set\_ylabel("% Change of Host Country Medal Count") toax.
* Apply the method .set\_title("Is there a Host Country Advantage?") to ax.
* Apply the method .set\_xticklabels(editions['City']) to ax.
* Reveal the final plot using plt.show().
* Save the following image using below command **plt.savefig('/code/output/host\_plot.png')**