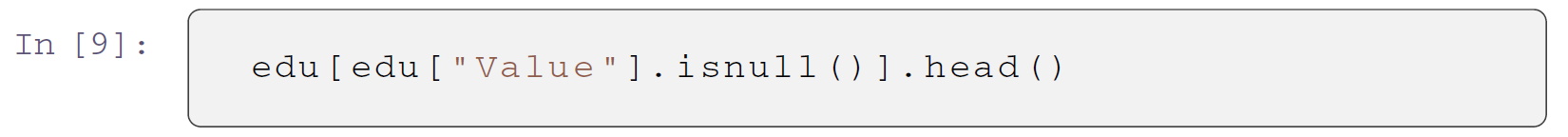
**Pandas**

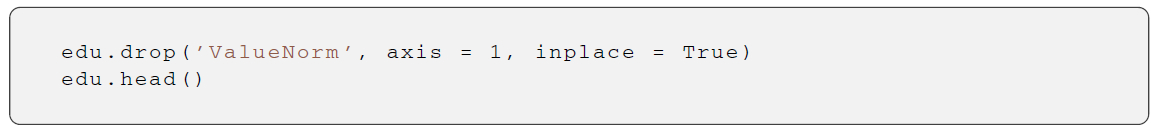
**FilteringMissing Values**

A subtle feature of NaN values is that two NaN are never equal. Because of this, the only safe way to tell whether a value is missing in a DataFrame is by using the isnull()

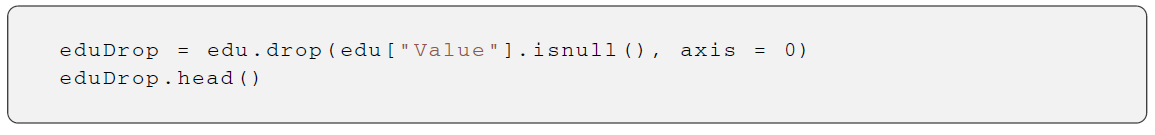


**Manipulating Data**

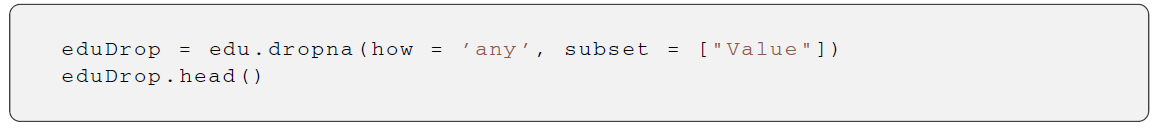
In Pandas, all the functions that change the contents of a DataFrame, such as the drop function, will normally return a copy of the modified data, instead of overwriting the DataFrame. Therefore, the original DataFrame is kept. If you do not want to keep the old values, you can set the keyword inplace to True.



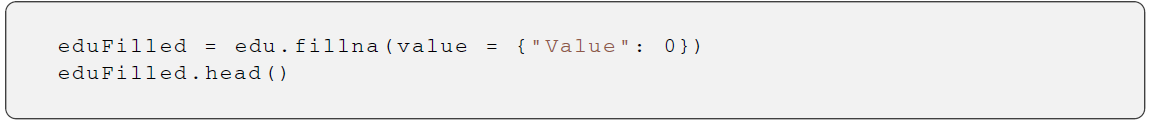
The drop() function is also used to remove missing values by applying it over the result of the isnull() function.



To remove NaN values, instead of the generic drop function, we can use the specific dropna() function. If we want to erase any row that contains an NaN value, we have to set the how keyword to any. To restrict it to a subset of columns, we can specify it using the subset keyword. As we can see below, the result will be the same as using the drop function

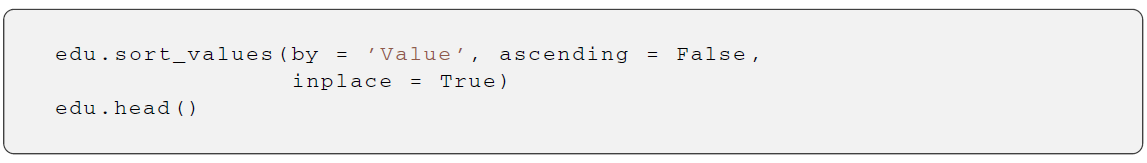


If, instead of removing the rows containing NaN, we want to fill them with another value, then we can use the fillna() method, specifying which value has to be used. If we want to fill only some specific columns, we have to set as argument to the fillna() function a dictionary with the name of the columns as the key and which character to be used for filling as the value

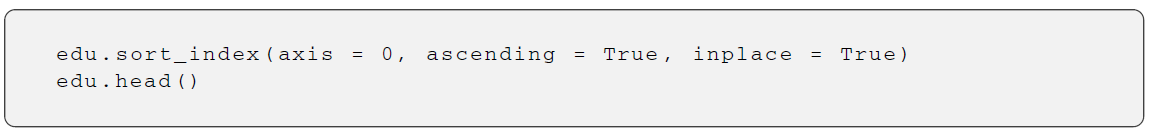


**Sorting**

We can sort a DataFrame using any column, using the sort function. If we want to see the first five rows of data sorted in descending order (i.e., from the largest to the smallest values) and using the Value column, then we just need to do this:



If we want to return to the original order, we can sort by an index using the sort\_index function and specifying axis=0:



**Grouping Data**