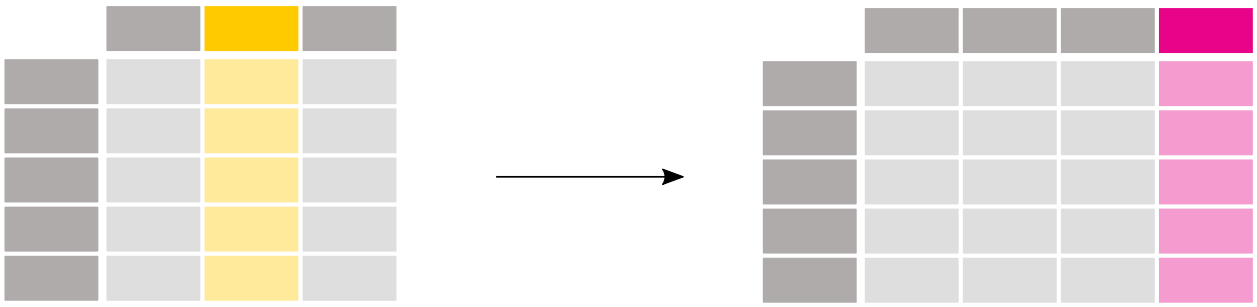
```
In [1]: import pandas as pd
```

Data used for this tutorial:

Air quality data

```
In [2]: air_quality = pd.read_csv("data/air_quality_no2.csv", index_col=0, parse_dates=True)

In [3]: air_quality.head()
Out[3]:
                     station_antwerp  station_paris  station_london
datetime
2019-05-07 02:00:00              NaN            NaN            23.0
2019-05-07 03:00:00             50.5           25.0            19.0
2019-05-07 04:00:00             45.0           27.7            19.0
2019-05-07 05:00:00              NaN           50.4            16.0
2019-05-07 06:00:00              NaN           61.9             NaN
```

How to create new columns derived from existing columns



I want to express the NO_2 concentration of the station in London in mg/m^3.

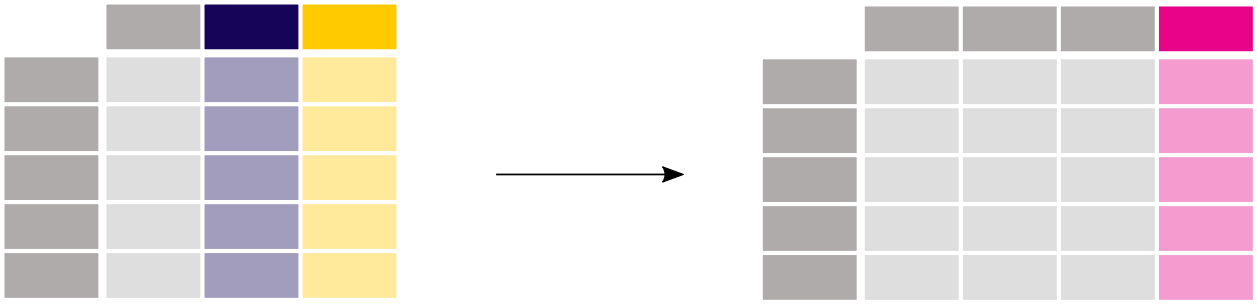
(*If we assume temperature of 25 degrees Celsius and pressure of 1013 hPa, the conversion factor is 1.882*)

```
In [4]: air_quality["london_mg_per_cubic"] = air_quality["station_london"] * 1.882

In [5]: air_quality.head()
Out[5]:
                     station_antwerp  ...  london_mg_per_cubic
datetime                              ...
2019-05-07 02:00:00              NaN  ...               43.286
2019-05-07 03:00:00             50.5  ...               35.758
2019-05-07 04:00:00             45.0  ...               35.758
2019-05-07 05:00:00              NaN  ...               30.112
2019-05-07 06:00:00              NaN  ...                  NaN

[5 rows x 4 columns]
```

To create a new column, use the `[]` brackets with the new column name at the left side of the assignment.

Note

The calculation of the values is done **element-wise**. This means all values in the given column are multiplied by the value 1.882 at once. You do not need to use a loop to iterate each of the rows!



I want to check the ratio of the values in Paris versus Antwerp and save the result in a new column.

```
In [6]: air_quality["ratio_paris_antwerp"] = (
   ...:     air_quality["station_paris"] / air_quality["station_antwerp"]
   ...: )
   ...:

In [7]: air_quality.head()
Out[7]:
                     station_antwerp  ...  ratio_paris_antwerp
datetime                              ...
2019-05-07 02:00:00              NaN  ...                  NaN
2019-05-07 03:00:00             50.5  ...             0.495050
2019-05-07 04:00:00             45.0  ...             0.615556
2019-05-07 05:00:00              NaN  ...                  NaN
2019-05-07 06:00:00              NaN  ...                  NaN

[5 rows x 5 columns]
```

The calculation is again element-wise, so the `/` is applied *for the values in each row*.

Also other mathematical operators (`+`, `-`, `*`, `/`,...) or logical operators (`<`, `>`, `==`,...) work element-wise. The latter was already used in the subset data tutorial to filter rows of a table using a conditional expression.

If you need more advanced logic, you can use arbitrary Python code via `apply()`.

I want to rename the data columns to the corresponding station identifiers used by OpenAQ.

```
In [8]: air_quality_renamed = air_quality.rename(
   ...:     columns={
   ...:         "station_antwerp": "BETR801",
   ...:         "station_paris": "FR04014",
   ...:         "station_london": "London Westminster",
   ...:     }
   ...: )
   ...:
```

```
In [9]: air_quality_renamed.head()
Out[9]:
                     BETR801  FR04014  ...  london_mg_per_cubic  ratio_paris_antwerp
datetime                               ...
```

```
2019-05-07 04:00:00     45.0     27.7  ...               35.758            0.615556
2019-05-07 05:00:00      NaN     50.4  ...               30.112                 NaN
2019-05-07 06:00:00      NaN     61.9  ...                  NaN                 NaN

[5 rows x 5 columns]
```

The `rename()` function can be used for both row labels and column labels. Provide a dictionary with the keys the current names and the values the new names to update the corresponding names.

The mapping should not be restricted to fixed names only, but can be a mapping function as well. For example, converting the column names to lowercase letters can be done using a function as well:

```
In [10]: air_quality_renamed = air_quality_renamed.rename(columns=str.lower)

In [11]: air_quality_renamed.head()
Out[11]:
                     betr801  fr04014  ...  london_mg_per_cubic  ratio_paris_antwerp
datetime                               ...
2019-05-07 02:00:00      NaN      NaN  ...               43.286                  NaN
2019-05-07 03:00:00     50.5     25.0  ...               35.758             0.495050
2019-05-07 04:00:00     45.0     27.7  ...               35.758             0.615556
2019-05-07 05:00:00      NaN     50.4  ...               30.112                  NaN
2019-05-07 06:00:00      NaN     61.9  ...                  NaN                  NaN

[5 rows x 5 columns]
```

To user guide Details about column or row label renaming is provided in the user guide section on renaming labels.

REMEMBER

- Create a new column by assigning the output to the DataFrame with a new column name in between the `[]`.
- Operations are element-wise, no need to loop over rows.
- Use `rename` with a dictionary or function to rename row labels or column names.

To user guide The user guide contains a separate section on column addition and deletion.

Previous
How do I create plots in pandas?

Next
How to calculate summary statistics