Python is open source. It’s great, but has the inherent problem of open source: many packages do (or try to do) the same thing. If you’re new to Python, it’s hard to know the best package for a specific task. You need someone who has experience to tell you. And I tell you today: there’s one package you absolutely need to learn for data science, and it’s called **pandas**.



And what’s really interesting with pandas is that many other packages are hidden in it. Pandas is a core package with additional features from a variety of other packages. And that’s great because you can work only using pandas.

pandas is like Excel in Python: it uses tables (namely **DataFrame**) and operates transformations on the data. But it can do a lot more.

*If you’re already familiar with Python, you can go straight to the 3rd paragraph*

Let’s start:

Don’t ask me why “pd” and not “p” or any other, it’s just like that. Deal with it :)

**The most elementary functions of pandas**

**Reading data**

sep means separator. If you’re working with French data, csv separator in Excel is “;” so you need to explicit it. Encoding is set to “latin-1” to read French characters. nrows=1000 means reading the first 1000 rows. skiprows=[2,5] means you will remove the 2nd and 5th row when reading the file

The most usual functions: read\_csv, read\_excel  
Some other great functions: read\_clipboard (which I use way too often, copying data from Excel or from the web), read\_sql

**Writing data**

index=None will simply write the data as it is. If you don’t write index=None, you’ll get an additional first column of 1,2,3, … until the last row.

I usually don’t go for the other functions, like .to\_excel, .to\_json, .to\_pickle since .to\_csv does very well the job. And because csv is the most common way to save tables. There’s also the .to\_clipboard if you’re like me an Excel maniac who wants to paste your results from Python to Excel.

**Checking the data**

Gives (#rows, #columns)

Computes basic statistics

**Seeing the data**

Print the first 3 rows of the data. Similarly to .head(), .tail() will look at the last rows of the data.

Print the 8th row

Print the value of the 8th row on “column\_1”

Subset from row 4 to 6 (excluded)

**The basic functions of pandas**

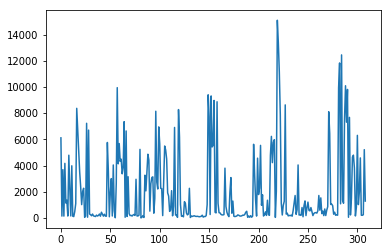
**Logical operations**

Subset the data thanks to logical operations. To use & (AND), ~ (NOT) and | (OR), you have to add “(“ and “)” before and after the logical operation.

Instead of writing multiple ORs for the same column, use the .isin() function

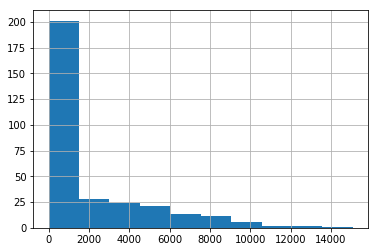
**Basic plotting**

This feature is made possible thanks to the matplotlib package. As we said in the intro, it’s usable directly in pandas.



Example of .plot() output

Plots the distribution (histogram)



Example of .hist() output

If you’re working with Jupyter, don’t forget to write this line (only once in the notebook), before plotting

**Updating the data**

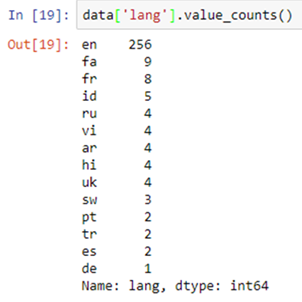
Replace the value in the 8th row at the ‘column\_1’ by ‘english’

Change values of multiple rows in one line

Alright, now you can do things that were easily accessible in Excel. Let’s dig in some amazing things that are not doable in Excel.

**Medium level functions**

**Counting occurrences**



Example of .value\_counts() output

**Operations on full rows, columns, or all data**

The len() function is applied to each element of the ‘column\_1’

The .map() operation applies a function to each element of a column.

A great pandas feature is the [chaining method](https://tomaugspurger.github.io/method-chaining). It helps you do multiple operations (.map() and .plot() here) in one line, for more simplicity and efficiency

.apply() applies a function to columns. Use .apply(, axis=1) to do it on the rows.

.applymap() applies a function to all cells in the table (DataFrame).

**tqdm, the one and only**

When working with large datasets, pandas can take some time running .map(), .apply(), .applymap() operations. tqdm is a very useful package that helps predict when theses operations will finish executing (yes I lied, I said we would use only pandas).

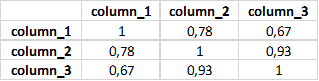
setup of tqdm with pandas

Replace .map() by .progress\_map(), same for .apply() and .applymap()

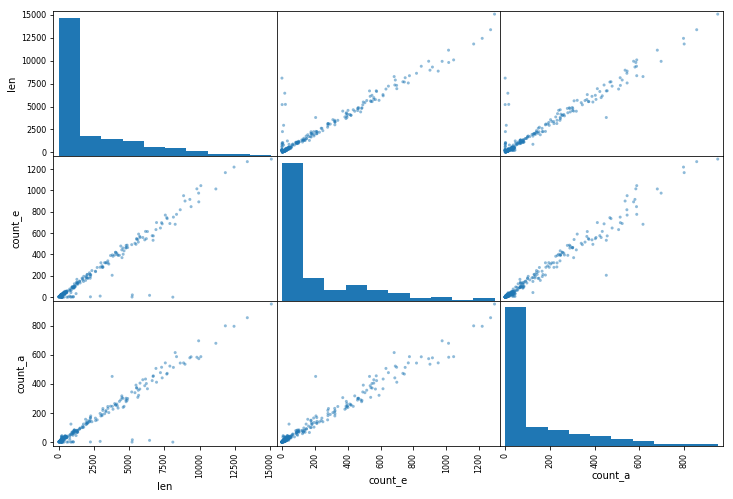
https://cdn-images-1.medium.com/max/800/1*uerveZ-vqCl5sTyaeRLwSw.gif

This is the progress bar you get in Jupyter with tqdm and pandas

**Correlation and scatter matrices**



.corr() will give you the correlation matrix



Example of scatter matrix. It plots all combinations of two columns in the same chart.

**Advanced operations in pandas**

**The SQL join**

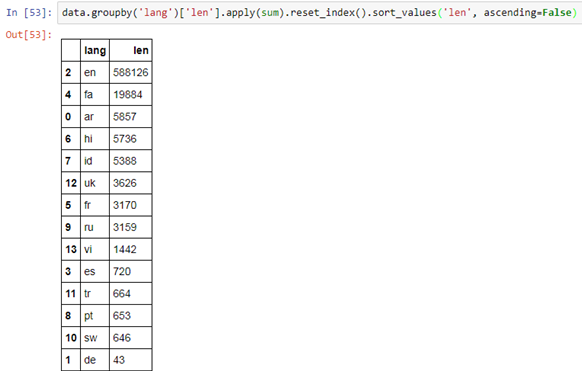
Joining in pandas is overly simple.

Joining on 3 columns takes just one line

**Grouping**

Not quite simple at the beginning, you need to master the syntax first and you’ll see yourself using this feature all the time.

Group by a column, the select another column on which to operate a function. The .reset\_index() reshapes your data as a DataFrame (table)



As explained previously, chain your functions in one line for optimal code

**Iterating over rows**

The .iterrows() loops through 2 variables together: the index of the row and the row (**i** and **row** in the code above).

**Overall pandas is one of the reason why Python is such a great language**

There are many other interesting pandas features I could have shown, but it’s already enough to understand why a data scientist cannot do without pandas.

To sum up, pandas is

* simple to use, hiding all the complex and abstract computations behind
* (generally) intuitive
* fast, if not the fastest data analysis package (it highly optimized in C)

It is THE tool that helps a data scientist to quickly read and understand data and be more efficient at his role.

I hope you found this article useful, and if you did, consider giving at least 50 claps :)