data36

MENU

Pandas Tutorial 1: Pandas Basics (Reading Data Files, DataFrames, Data Selection)

Written by Tomi Mester on July 10, 2018

Pandas is one of the most popular Python libraries for Data Science and Analytics. I like to say it's the "SQL of Python." Why? Because pandas helps you to manage two-dimensional data tables in Python. Of course, it has many more features. In this pandas tutorial series, I'll show you the most important (that is, the most often used) things that you have to know as an Analyst or a Data Scientist. This is the first episode and we will start from the basics!

Note 1: this is a hands-on tutorial, so I recommend doing the coding part with me!

Before we start

If you haven't done so yet, I recommend going through these articles first:

- How to install Python, R, SQL and bash to practice data science
- Python for Data Science – Basics #1 – Variables and basic operations
- Python Import Statement and the Most Important Built-in Modules

We use cookies to ensure that we give you the best experience on our website.

Ok Read more

To follow this pandas tutorial

1. You will need a fully functioning data server with Python3, numpy and pandas on it.
   Note 1 : Again, with this tutorial you can set up your data server and Python3. And with this article you can set up numpy and pandas, too.
   Note 2: or take this step-by-step data server set up video course.

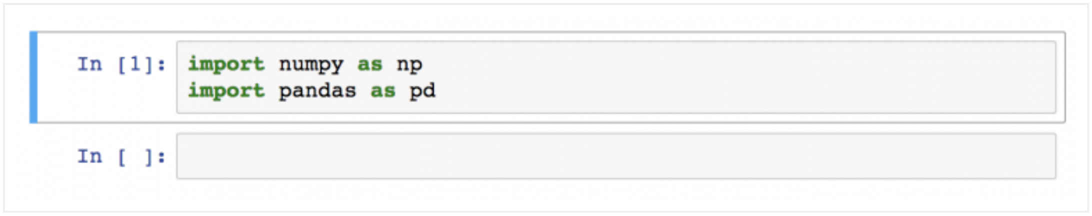
2. Next step: log in to your server and fire up Jupyter. Then open a new Jupyter Notebook in your favorite browser. (If you don't know how to do that, I really do recommend going through the articles I linked in the "*Before we start*" section.)
   Note: I'll also rename my Jupyter Notebook to "pandas_tutorial_1".

```
tomis-mbp:~ tomimester$
```

Firing up Jupyter Notebook

3. Import numpy and pandas to your Jupyter Notebook by running these two lines in a cell:

```
import numpy as np
import pandas as pd
```



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

In [ ]:
```

> *Note: It's conventional to refer to 'pandas' as 'pd'. When you add the* `as pd` *at the end of your import statement, your Jupyter Notebook understands that from this point on every time you type* `pd`*, you are actually referring to the* `pandas` *library.*

Okay, now we have everything! Let's start with this pandas tutorial!
The first question is:

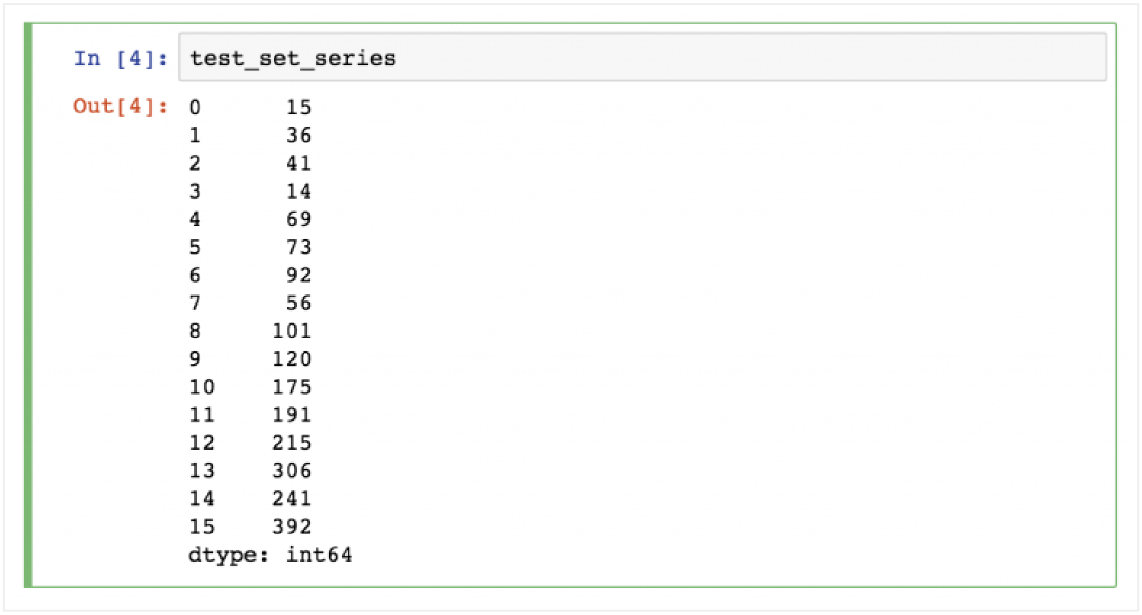
How to open data files in pandas

You might have your data in .csv files or SQL tables. Maybe Excel files. Or .tsv files. Or something else. But the goal is the same in all cases. If you want to analyze that data using pandas, the first step will be to read it into a data structure that's compatible with pandas.

Pandas data structures

There are two types of data structures in pandas: **Series** and **DataFrames**.

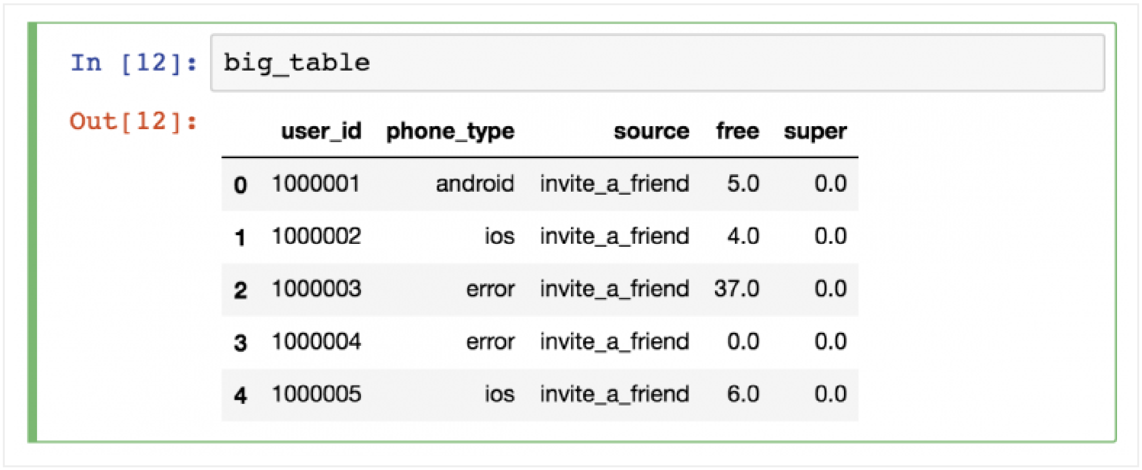
Series: a pandas Series is a one dimensional data structure (*"a one dimensional ndarray"*) that can store values — and for every value it holds a unique index, too.



```
In [4]: test_set_series
Out[4]: 0       15
        1       36
        2       41
        3       14
        4       69
        5       73
        6       92
        7       56
        8      101
        9      120
        10     175
        11     191
        12     215
        13     306
        14     241
        15     392
        dtype: int64
```

Pandas Series example

DataFrame: a pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes.



```
In [12]: big_table
Out[12]:
```

	user_id	phone_type	source	free	super
0	1000001	android	invite_a_friend	5.0	0.0
1	1000002	ios	invite_a_friend	4.0	0.0
2	1000003	error	invite_a_friend	37.0	0.0
3	1000004	error	invite_a_friend	0.0	0.0
4	1000005	ios	invite_a_friend	6.0	0.0

Pandas DataFrame example

In this pandas tutorial, I'll focus mostly on DataFrames. The reason is simple: most of the analytical methods I will talk about will make more sense in a 2D datatable than in a 1D array.

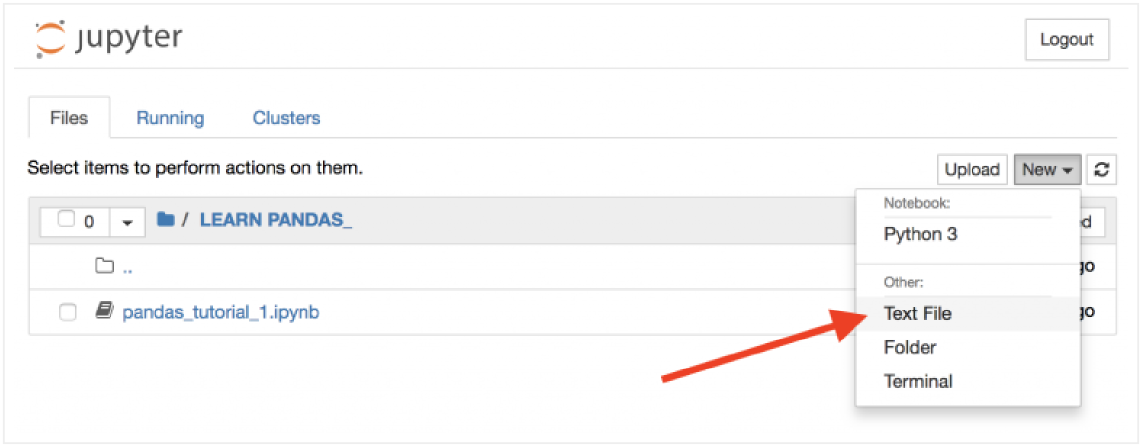
Loading a .csv file into a pandas DataFrame

Okay, time to put things into practice! Let's load a .csv data file into pandas! There is a function for it, called `read_csv()`.

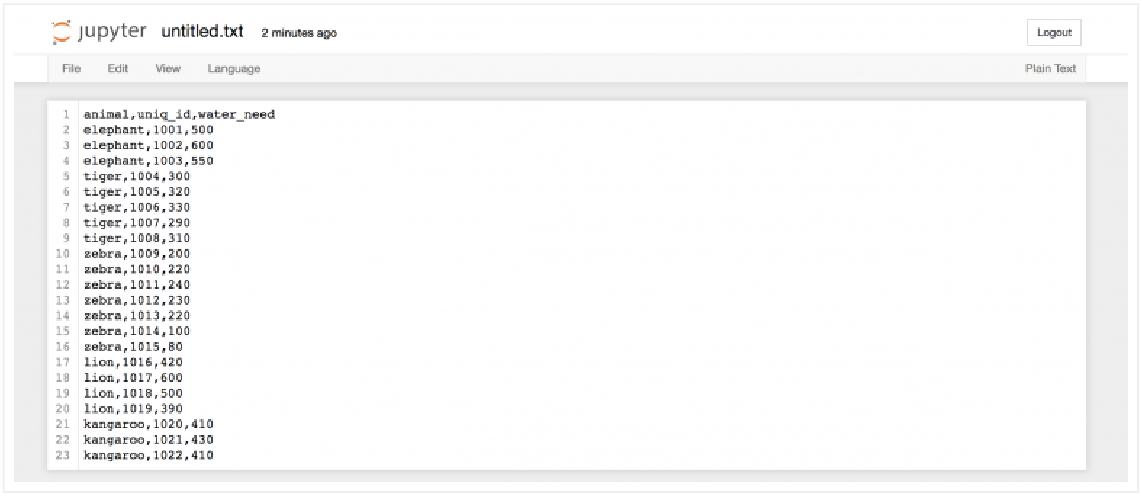
Start with a simple demo data set, called zoo! This time – for the sake of practicing – you will create a .csv file for yourself! Here's the raw data:

```
animal,uniq_id,water_need
elephant,1001,500
elephant,1002,600
elephant,1003,550
tiger,1004,300
tiger,1005,320
tiger,1006,330
tiger,1007,290
tiger,1008,310
zebra,1009,200
zebra,1010,220
zebra,1011,240
zebra,1012,230
zebra,1013,220
zebra,1014,100
zebra,1015,80
lion,1016,420
lion,1017,600
lion,1018,500
lion,1019,390
kangaroo,1020,410
kangaroo,1021,430
kangaroo,1022,410
```

Go back to your Jupyter Home tab and create a new text file…

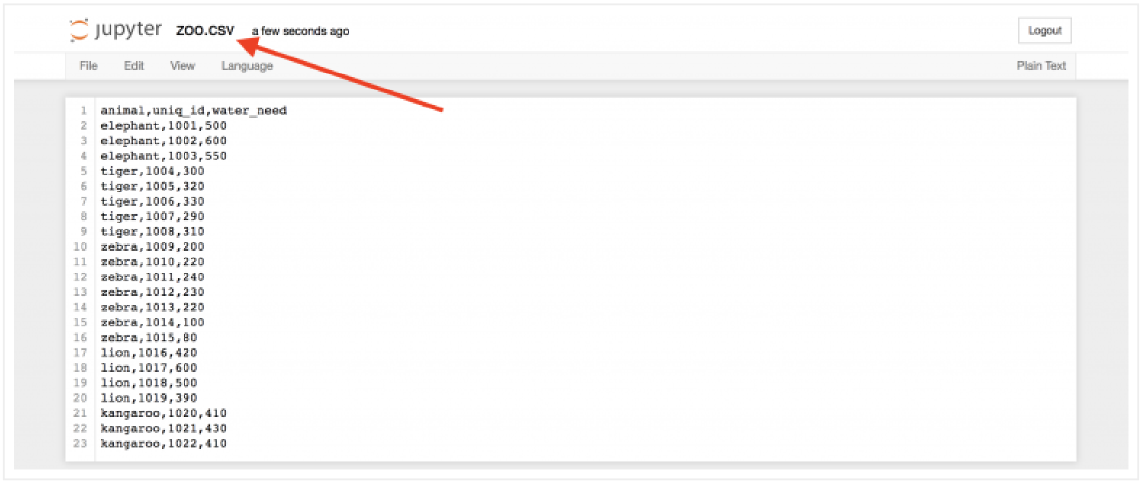


…then copy-paste the above zoo data into this text file…



```
animal,uniq_id,water_need
elephant,1001,500
elephant,1002,600
elephant,1003,550
tiger,1004,300
tiger,1005,320
tiger,1006,330
tiger,1007,290
tiger,1008,310
zebra,1009,200
zebra,1010,220
zebra,1011,240
zebra,1012,230
zebra,1013,220
zebra,1014,100
zebra,1015,80
lion,1016,420
lion,1017,600
lion,1018,500
lion,1019,390
kangaroo,1020,410
kangaroo,1021,430
kangaroo,1022,410
```

… and then rename this text file to zoo.csv!

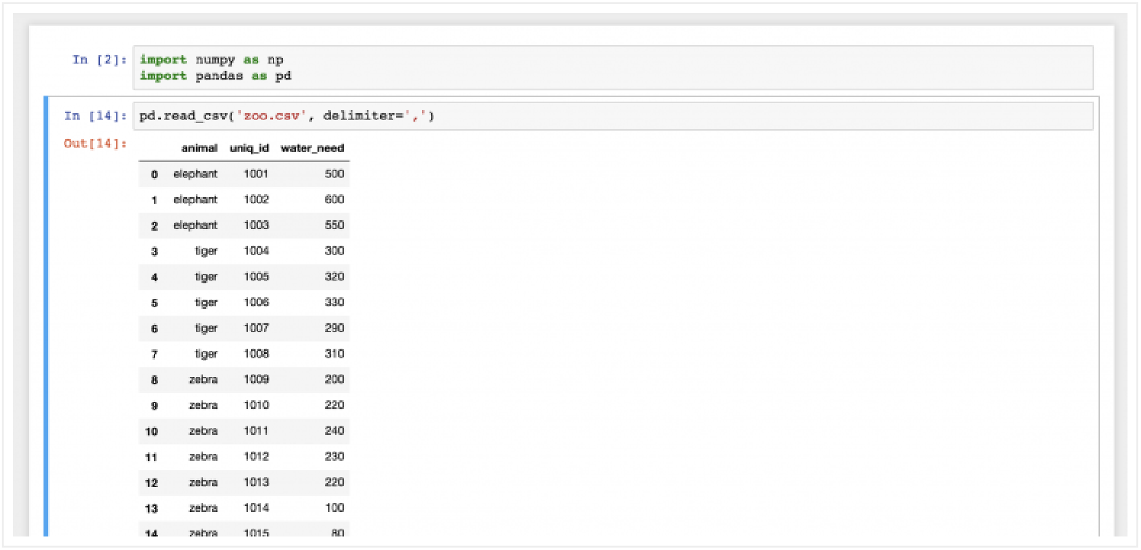


```
animal,uniq_id,water_need
elephant,1001,500
elephant,1002,600
elephant,1003,550
tiger,1004,300
tiger,1005,320
tiger,1006,330
tiger,1007,290
tiger,1008,310
zebra,1009,200
zebra,1010,220
zebra,1011,240
zebra,1012,230
zebra,1013,220
zebra,1014,100
zebra,1015,80
lion,1016,420
lion,1017,600
lion,1018,500
lion,1019,390
kangaroo,1020,410
kangaroo,1021,430
kangaroo,1022,410
```

Okay, this is our .csv file.
Now, go back to your Jupyter Notebook (that I named '*pandas_tutorial_1*') and open this freshly created .csv file in it!

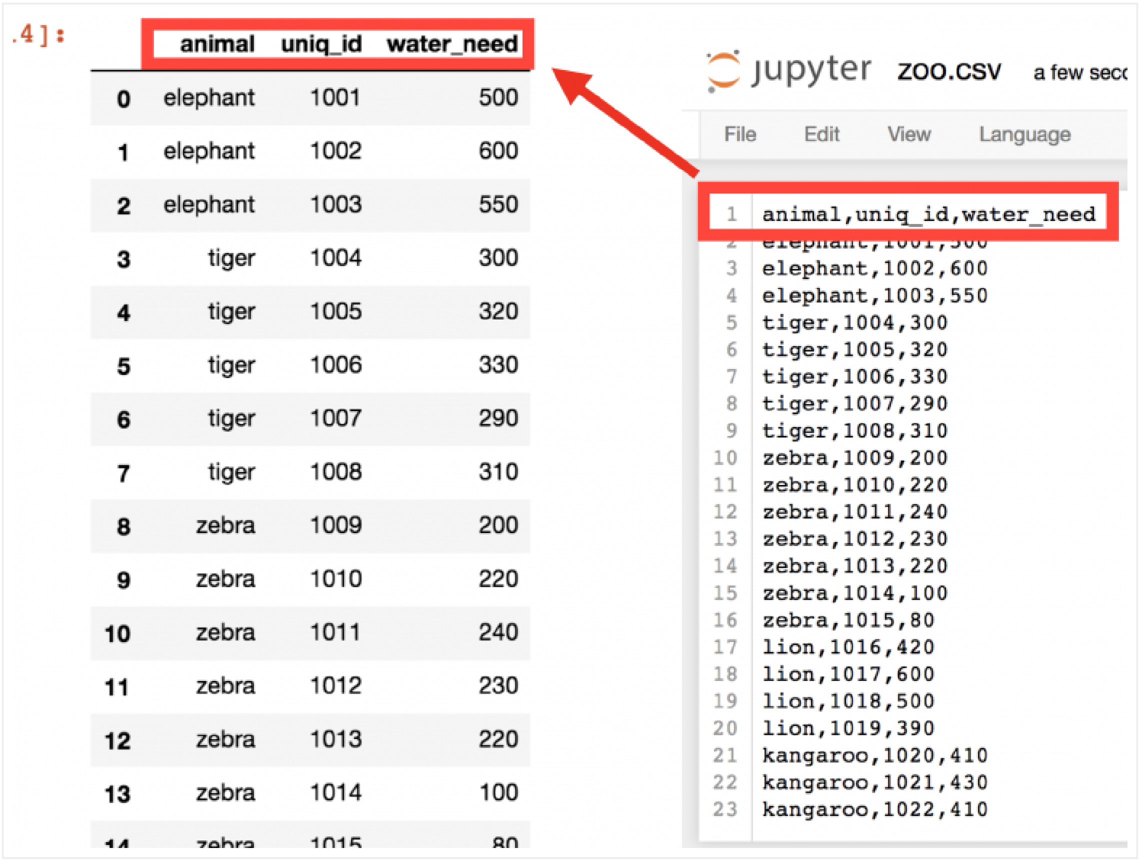
Again, the function that you have to use is: `read_csv()`
Type this to a new cell:

```
pd.read_csv('zoo.csv', delimiter = ',')
```



```
In [2]: import numpy as np
        import pandas as pd

In [14]: pd.read_csv('zoo.csv', delimiter=',')

Out[14]:
       animal  uniq_id  water_need
0    elephant     1001         500
1    elephant     1002         600
2    elephant     1003         550
3       tiger     1004         300
4       tiger     1005         320
5       tiger     1006         330
6       tiger     1007         290
7       tiger     1008         310
8       zebra     1009         200
9       zebra     1010         220
10      zebra     1011         240
11      zebra     1012         230
12      zebra     1013         220
13      zebra     1014         100
14      zebra     1015          80
```

And there you go! This is the zoo.csv data file, brought to pandas. This nice 2D table? Well, this is a *pandas dataframe*. The numbers on the left are the indexes. And the column names on the top are picked up from the first row of our zoo.csv file.



```
.4]:
       animal  uniq_id  water_need
0    elephant     1001         500
1    elephant     1002         600
2    elephant     1003         550
3       tiger     1004         300
4       tiger     1005         320
5       tiger     1006         330
6       tiger     1007         290
7       tiger     1008         310
8       zebra     1009         200
9       zebra     1010         220
10      zebra     1011         240
11      zebra     1012         230
12      zebra     1013         220
13      zebra     1014         100
14      zebra     1015          80
```

```
animal,uniq_id,water_need
elephant,1001,500
elephant,1002,600
elephant,1003,550
tiger,1004,300
tiger,1005,320
tiger,1006,330
tiger,1007,290
tiger,1008,310
zebra,1009,200
zebra,1010,220
zebra,1011,240
zebra,1012,230
zebra,1013,220
zebra,1014,100
zebra,1015,80
lion,1016,420
lion,1017,600
lion,1018,500
lion,1019,390
kangaroo,1020,410
kangaroo,1021,430
kangaroo,1022,410
```

To be honest, though, you will probably never create a .csv data file for yourself, like we just did… you will use pre-existing data files. So you have to learn how to download .csv files to your server!

If you are here from the Junior Data Scientist's First Month video course then you have already dealt with downloading your .txt or .csv data files to your data server, so you must be pretty proficient in it… But if you are not here from the course (or if you want to learn another way to download a .csv file to your server and to get another exciting dataset), follow these steps:

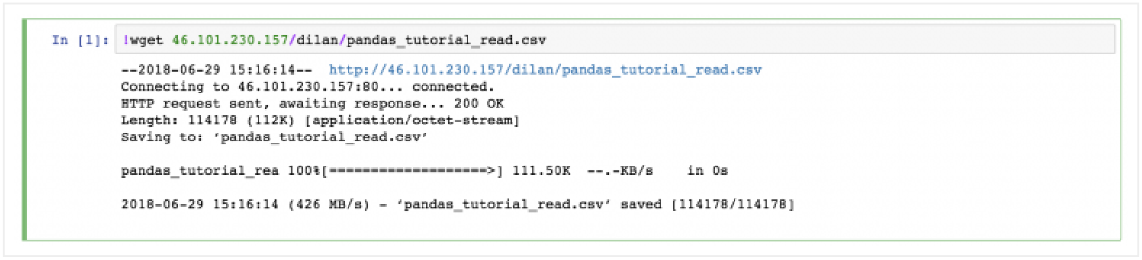
I've uploaded a small sample dataset here: DATASET

(Link: 46.101.230.157/dilan/pandas_tutorial_read.csv)

If you click the link, the data file will be downloaded to your computer. But you don't want to download this data file to your computer, right? You want to download it to your server and then load it to your Jupyter Notebook. It only takes two steps.

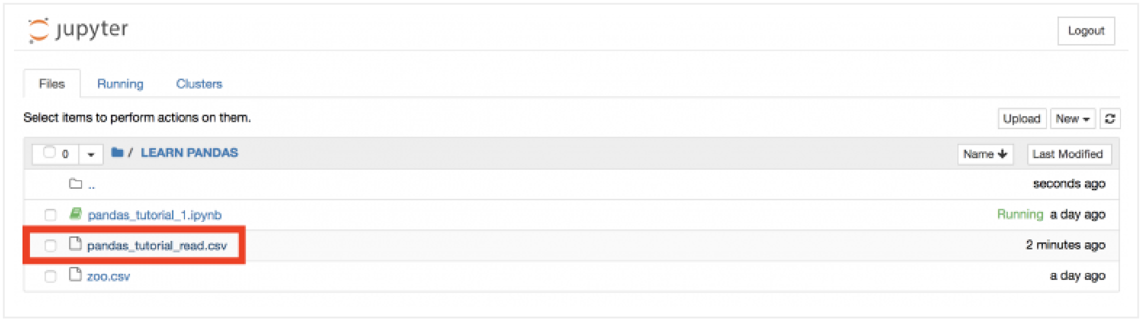
STEP 1) Go back to your Jupyter Notebook and type this command:

```
!wget 46.101.230.157/dilan/pandas_tutorial_read.csv
```



```
In [1]: !wget 46.101.230.157/dilan/pandas_tutorial_read.csv

        --2018-06-29 15:16:14--  http://46.101.230.157/dilan/pandas_tutorial_read.csv
        Connecting to 46.101.230.157:80... connected.
        HTTP request sent, awaiting response... 200 OK
        Length: 114178 (112K) [application/octet-stream]
        Saving to: 'pandas_tutorial_read.csv'

        pandas_tutorial_rea 100%[===================>] 111.50K  --.-KB/s    in 0s

        2018-06-29 15:16:14 (426 MB/s) - 'pandas_tutorial_read.csv' saved [114178/114178]
```

This downloaded the pandas_tutorial_read.csv file to your server. Just check it out:



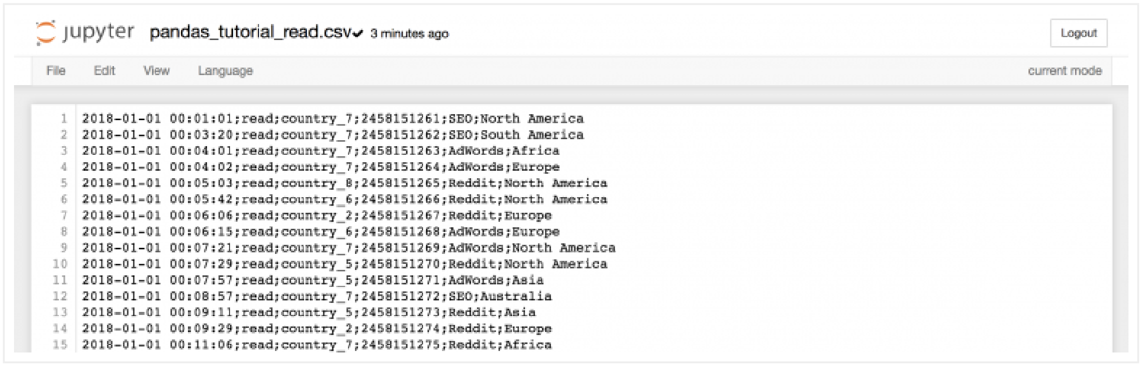
Select items to perform actions on them.

0 / LEARN PANDAS

..		seconds ago
pandas_tutorial_1.ipynb	Running	a day ago
pandas_tutorial_read.csv		2 minutes ago
zoo.csv		a day ago

See? It's there.

If you click it…



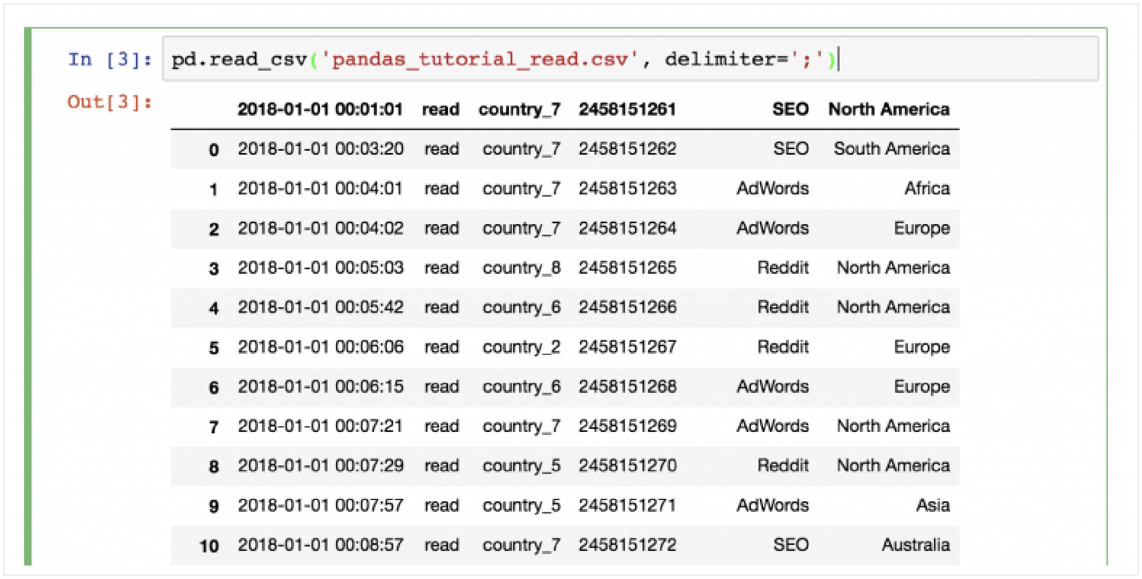
```
2018-01-01 00:01:01;read;country_7;2458151261;SEO;North America
2018-01-01 00:03:20;read;country_7;2458151262;SEO;South America
2018-01-01 00:04:01;read;country_7;2458151263;AdWords;Africa
2018-01-01 00:04:02;read;country_7;2458151264;AdWords;Europe
2018-01-01 00:05:03;read;country_8;2458151265;Reddit;North America
2018-01-01 00:05:42;read;country_6;2458151266;Reddit;North America
2018-01-01 00:06:06;read;country_2;2458151267;Reddit;Europe
2018-01-01 00:06:15;read;country_6;2458151268;AdWords;Europe
2018-01-01 00:07:21;read;country_7;2458151269;AdWords;North America
2018-01-01 00:07:29;read;country_5;2458151270;Reddit;North America
2018-01-01 00:07:57;read;country_5;2458151271;AdWords;Asia
2018-01-01 00:08:57;read;country_7;2458151272;SEO;Australia
2018-01-01 00:09:11;read;country_5;2458151273;Reddit;Asia
2018-01-01 00:09:29;read;country_2;2458151274;Reddit;Europe
2018-01-01 00:11:06;read;country_7;2458151275;Reddit;Africa
```

…you can even check out the data in it.

STEP 2) Now, go back again to your Jupyter Notebook and use the same `read_csv` function that we have used before (but don't forget to change the file name and the delimiter value):

```
pd.read_csv('pandas_tutorial_read.csv', delimiter=';')
```

The data is loaded into pandas!

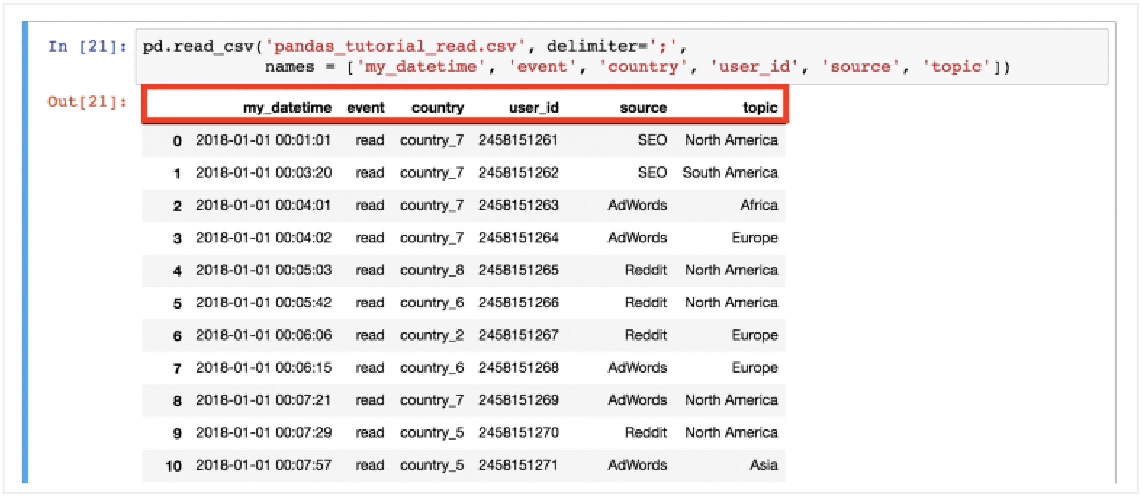


```
In [3]: pd.read_csv('pandas_tutorial_read.csv', delimiter=';')
Out[3]:
```

	2018-01-01 00:01:01	read	country_7	2458151261	SEO	North America
0	2018-01-01 00:03:20	read	country_7	2458151262	SEO	South America
1	2018-01-01 00:04:01	read	country_7	2458151263	AdWords	Africa
2	2018-01-01 00:04:02	read	country_7	2458151264	AdWords	Europe
3	2018-01-01 00:05:03	read	country_8	2458151265	Reddit	North America
4	2018-01-01 00:05:42	read	country_6	2458151266	Reddit	North America
5	2018-01-01 00:06:06	read	country_2	2458151267	Reddit	Europe
6	2018-01-01 00:06:15	read	country_6	2458151268	AdWords	Europe
7	2018-01-01 00:07:21	read	country_7	2458151269	AdWords	North America
8	2018-01-01 00:07:29	read	country_5	2458151270	Reddit	North America
9	2018-01-01 00:07:57	read	country_5	2458151271	AdWords	Asia
10	2018-01-01 00:08:57	read	country_7	2458151272	SEO	Australia

Does something feel off? Yes, this time we didn't have a header in our csv file, so we have to set it up manually! Add the names parameter to your function!

```
pd.read_csv('pandas_tutorial_read.csv', delimiter=';', names =
['my_datetime', 'event', 'country', 'user_id', 'source', 'topic'])
```

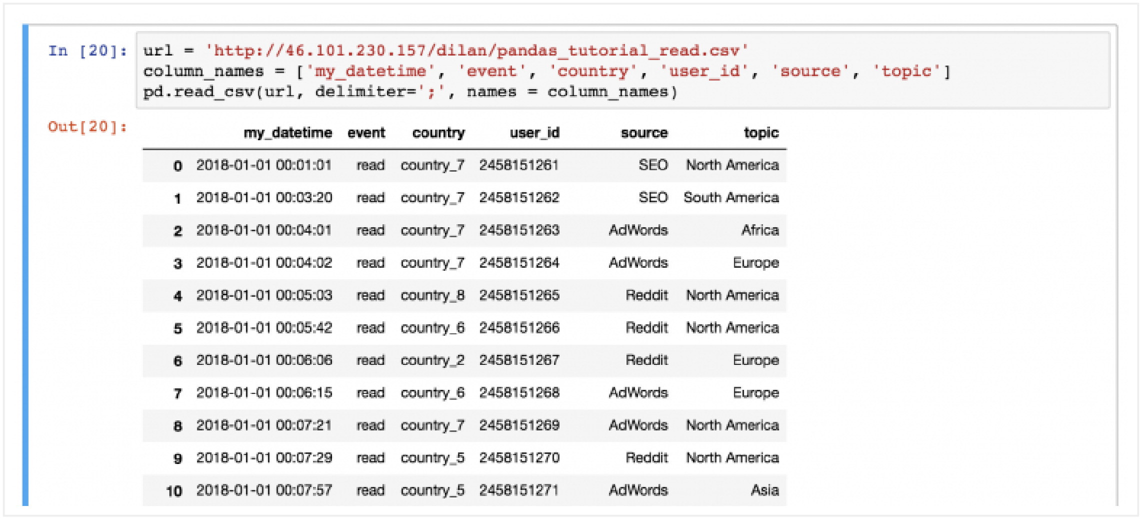


```
In [21]: pd.read_csv('pandas_tutorial_read.csv', delimiter=';',
                   names = ['my_datetime', 'event', 'country', 'user_id', 'source', 'topic'])
Out[21]:
```

	my_datetime	event	country	user_id	source	topic
0	2018-01-01 00:01:01	read	country_7	2458151261	SEO	North America
1	2018-01-01 00:03:20	read	country_7	2458151262	SEO	South America
2	2018-01-01 00:04:01	read	country_7	2458151263	AdWords	Africa
3	2018-01-01 00:04:02	read	country_7	2458151264	AdWords	Europe
4	2018-01-01 00:05:03	read	country_8	2458151265	Reddit	North America
5	2018-01-01 00:05:42	read	country_6	2458151266	Reddit	North America
6	2018-01-01 00:06:06	read	country_2	2458151267	Reddit	Europe
7	2018-01-01 00:06:15	read	country_6	2458151268	AdWords	Europe
8	2018-01-01 00:07:21	read	country_7	2458151269	AdWords	North America
9	2018-01-01 00:07:29	read	country_5	2458151270	Reddit	North America
10	2018-01-01 00:07:57	read	country_5	2458151271	AdWords	Asia

Better!
And with that, we finally loaded our .csv data into a **pandas dataframe!**

Note 1: Just so you know, there is an alternative method. (I don't prefer it though.) You can load the .csv data using the URL directly. In this case the data won't be downloaded to your data server.



```
In [20]: url = 'http://46.101.230.157/dilan/pandas_tutorial_read.csv'
         column_names = ['my_datetime', 'event', 'country', 'user_id', 'source', 'topic']
         pd.read_csv(url, delimiter=';', names = column_names)
Out[20]:
```

	my_datetime	event	country	user_id	source	topic
0	2018-01-01 00:01:01	read	country_7	2458151261	SEO	North America
1	2018-01-01 00:03:20	read	country_7	2458151262	SEO	South America
2	2018-01-01 00:04:01	read	country_7	2458151263	AdWords	Africa
3	2018-01-01 00:04:02	read	country_7	2458151264	AdWords	Europe
4	2018-01-01 00:05:03	read	country_8	2458151265	Reddit	North America
5	2018-01-01 00:05:42	read	country_6	2458151266	Reddit	North America
6	2018-01-01 00:06:06	read	country_2	2458151267	Reddit	Europe
7	2018-01-01 00:06:15	read	country_6	2458151268	AdWords	Europe
8	2018-01-01 00:07:21	read	country_7	2458151269	AdWords	North America
9	2018-01-01 00:07:29	read	country_5	2458151270	Reddit	North America
10	2018-01-01 00:07:57	read	country_5	2458151271	AdWords	Asia

read the .csv directly from the server (using its URL)

Note 2: If you are wondering what's in this data set – this is the data log of a travel blog. This is a log of one day only (if you are a JDS course participant, you will get much more of this data set on the last week of the course ;-)). I guess the names of the columns are fairly self-explanatory.

Selecting data from a dataframe in pandas

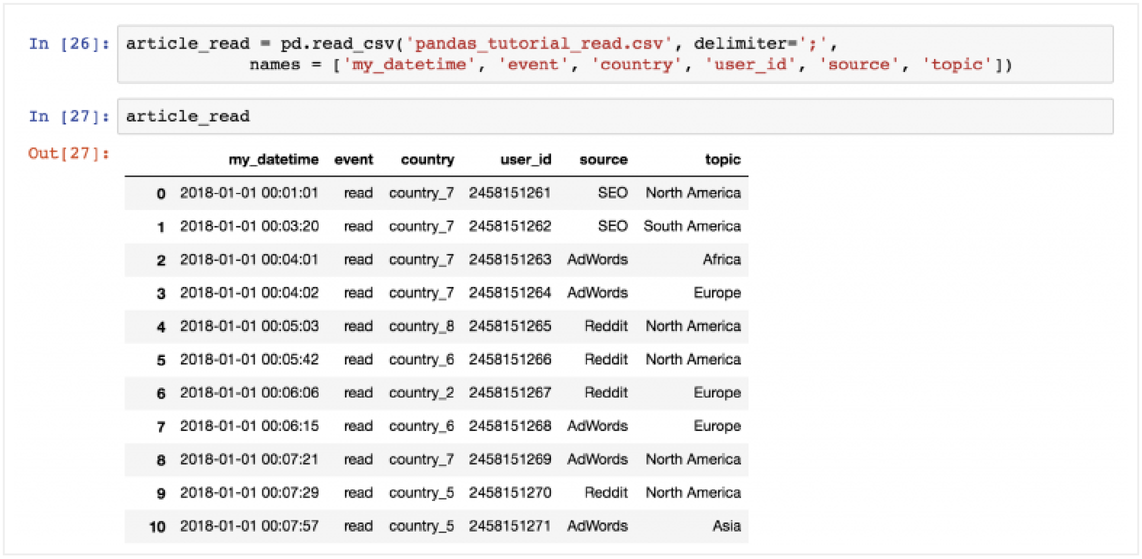
This is the first episode of this pandas tutorial series, so let's start with a few very basic data selection methods – and in the next episodes we will go deeper!

1) Print the whole dataframe

The most basic method is to print your whole data frame to your screen. Of course, you don't have to run the `pd.read_csv()` function again and again and again. Just store its output the first time you run it!

```
article_read = pd.read_csv('pandas_tutorial_read.csv', delimiter=';', names = ['my_datetime', 'event', 'country', 'user_id', 'source', 'topic'])
```

After that, you can call this `article_read` value anytime to print your DataFrame!

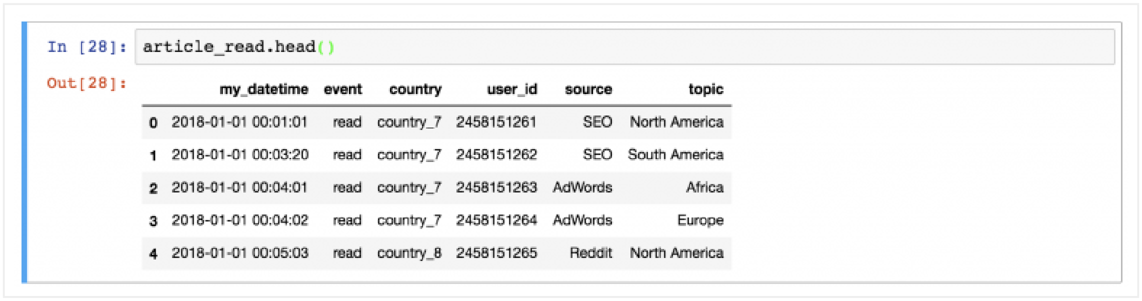


```
In [26]: article_read = pd.read_csv('pandas_tutorial_read.csv', delimiter=';',
                        names = ['my_datetime', 'event', 'country', 'user_id', 'source', 'topic'])

In [27]: article_read

Out[27]:
            my_datetime  event    country     user_id   source          topic
 0  2018-01-01 00:01:01   read  country_7  2458151261      SEO  North America
 1  2018-01-01 00:03:20   read  country_7  2458151262      SEO  South America
 2  2018-01-01 00:04:01   read  country_7  2458151263  AdWords         Africa
 3  2018-01-01 00:04:02   read  country_7  2458151264  AdWords         Europe
 4  2018-01-01 00:05:03   read  country_8  2458151265   Reddit  North America
 5  2018-01-01 00:05:42   read  country_6  2458151266   Reddit  North America
 6  2018-01-01 00:06:06   read  country_2  2458151267   Reddit         Europe
 7  2018-01-01 00:06:15   read  country_6  2458151268  AdWords         Europe
 8  2018-01-01 00:07:21   read  country_7  2458151269  AdWords  North America
 9  2018-01-01 00:07:29   read  country_5  2458151270   Reddit  North America
10  2018-01-01 00:07:57   read  country_5  2458151271  AdWords           Asia
```

2) Print a sample of your dataframe

Sometimes, it's handy not to print the whole dataframe and flood your screen with data. When a few lines is enough, you can print only the first 5 lines – by typing:

```
article_read.head()
```

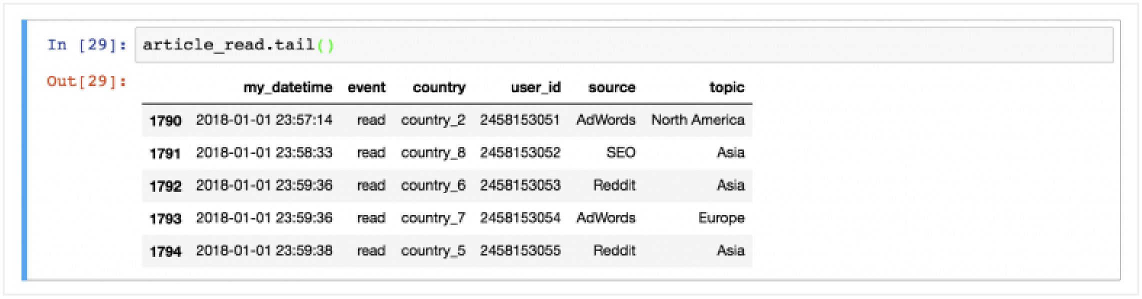


```
In [28]: article_read.head()
Out[28]:
```

	my_datetime	event	country	user_id	source	topic
0	2018-01-01 00:01:01	read	country_7	2458151261	SEO	North America
1	2018-01-01 00:03:20	read	country_7	2458151262	SEO	South America
2	2018-01-01 00:04:01	read	country_7	2458151263	AdWords	Africa
3	2018-01-01 00:04:02	read	country_7	2458151264	AdWords	Europe
4	2018-01-01 00:05:03	read	country_8	2458151265	Reddit	North America

Or the last few lines by typing:

```
article_read.tail()
```

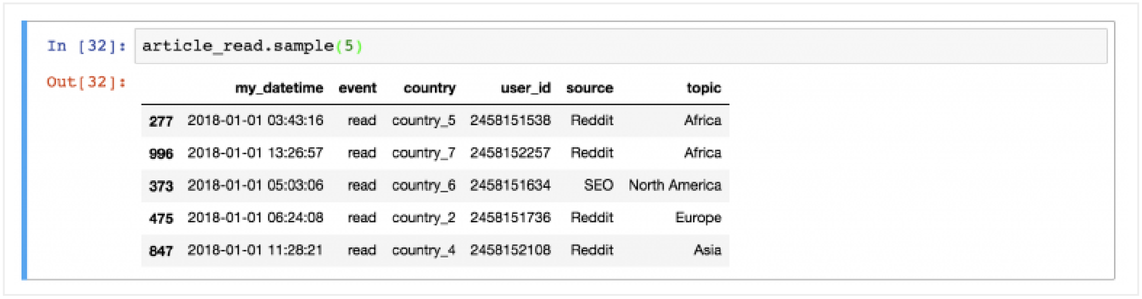


```
In [29]: article_read.tail()
Out[29]:
```

	my_datetime	event	country	user_id	source	topic
1790	2018-01-01 23:57:14	read	country_2	2458153051	AdWords	North America
1791	2018-01-01 23:58:33	read	country_8	2458153052	SEO	Asia
1792	2018-01-01 23:59:36	read	country_6	2458153053	Reddit	Asia
1793	2018-01-01 23:59:36	read	country_7	2458153054	AdWords	Europe
1794	2018-01-01 23:59:38	read	country_5	2458153055	Reddit	Asia

Or a few random lines by typing:

```
article_read.sample(5)
```



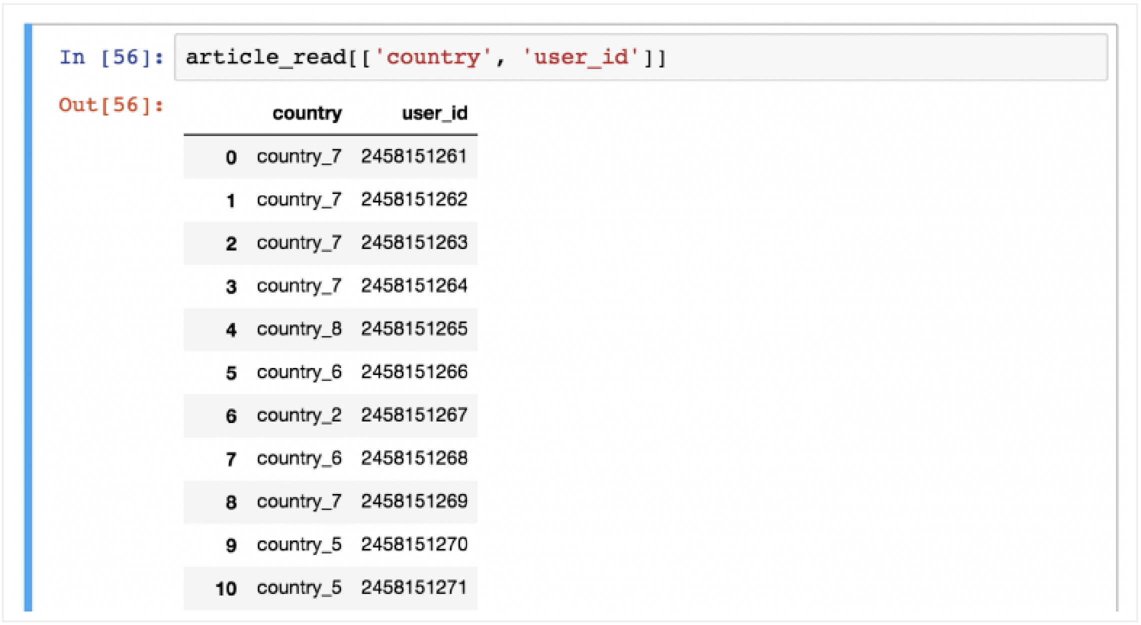
```
In [32]: article_read.sample(5)
Out[32]:
```

	my_datetime	event	country	user_id	source	topic
277	2018-01-01 03:43:16	read	country_5	2458151538	Reddit	Africa
996	2018-01-01 13:26:57	read	country_7	2458152257	Reddit	Africa
373	2018-01-01 05:03:06	read	country_6	2458151634	SEO	North America
475	2018-01-01 06:24:08	read	country_2	2458151736	Reddit	Europe
847	2018-01-01 11:28:21	read	country_4	2458152108	Reddit	Asia

3) Select specific columns of your dataframe

This one is a bit tricky! Let's say you want to print the 'country' and the 'user_id' columns only.
You should use this syntax:

```
article_read[['country', 'user_id']]
```



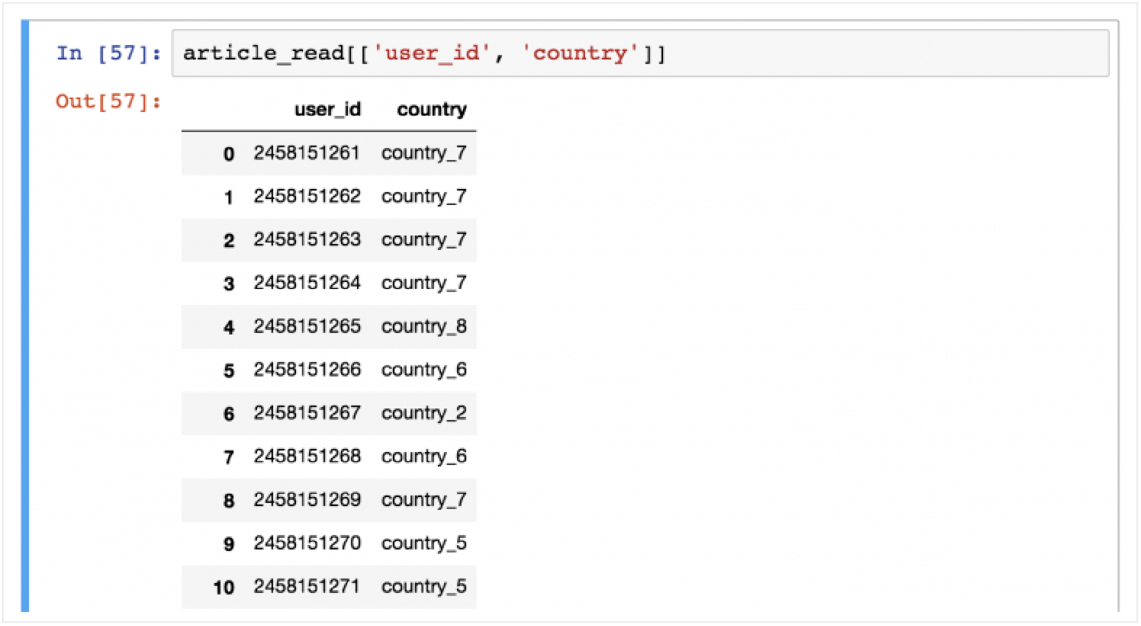
```
In [56]: article_read[['country', 'user_id']]
Out[56]:
```

	country	user_id
0	country_7	2458151261
1	country_7	2458151262
2	country_7	2458151263
3	country_7	2458151264
4	country_8	2458151265
5	country_6	2458151266
6	country_2	2458151267
7	country_6	2458151268
8	country_7	2458151269
9	country_5	2458151270
10	country_5	2458151271

Any guesses why we have to use double bracket frames? It seems a bit over-complicated, I admit, but maybe this will help you remember: the outer bracket frames tell pandas that you want to select columns, and the inner brackets are for the list (*remember? Python lists go between bracket frames*) of the column names.

By the way, if you change the order of the column names, the order of the returned columns will change, too:

```
article_read[['user_id', 'country']]
```



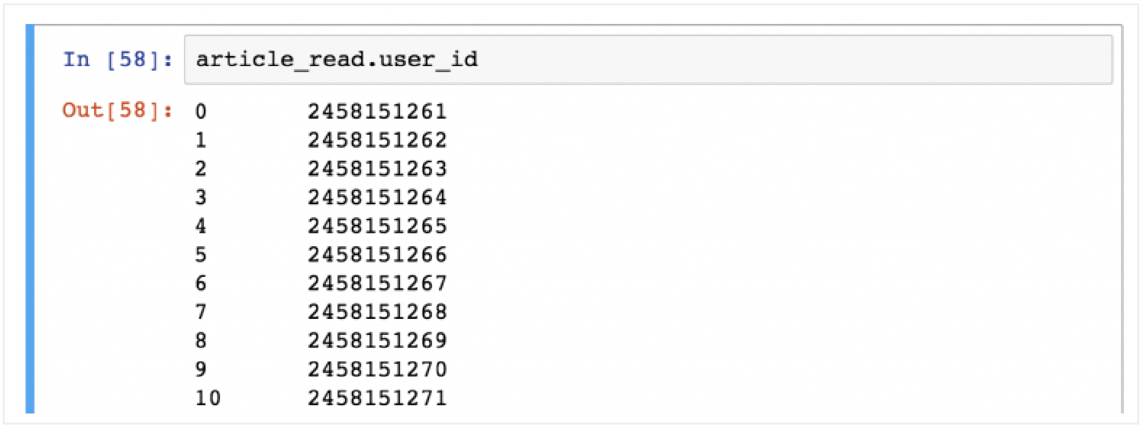
```
In [57]: article_read[['user_id', 'country']]
Out[57]:
```

	user_id	country
0	2458151261	country_7
1	2458151262	country_7
2	2458151263	country_7
3	2458151264	country_7
4	2458151265	country_8
5	2458151266	country_6
6	2458151267	country_2
7	2458151268	country_6
8	2458151269	country_7
9	2458151270	country_5
10	2458151271	country_5

This is the DataFrame of your selected columns.

Note: Sometimes (especially in predictive analytics projects), you want to get Series objects instead of DataFrames. You can get a Series using any of these two syntaxes (and selecting only one column):

`article_read.user_id`
`article_read['user_id']`



```
In [58]: article_read.user_id

Out[58]: 0        2458151261
         1        2458151262
         2        2458151263
         3        2458151264
         4        2458151265
         5        2458151266
         6        2458151267
         7        2458151268
         8        2458151269
         9        2458151270
         10       2458151271
```

output is a Series object and not a DataFrame object

4) Filter for specific values in your dataframe

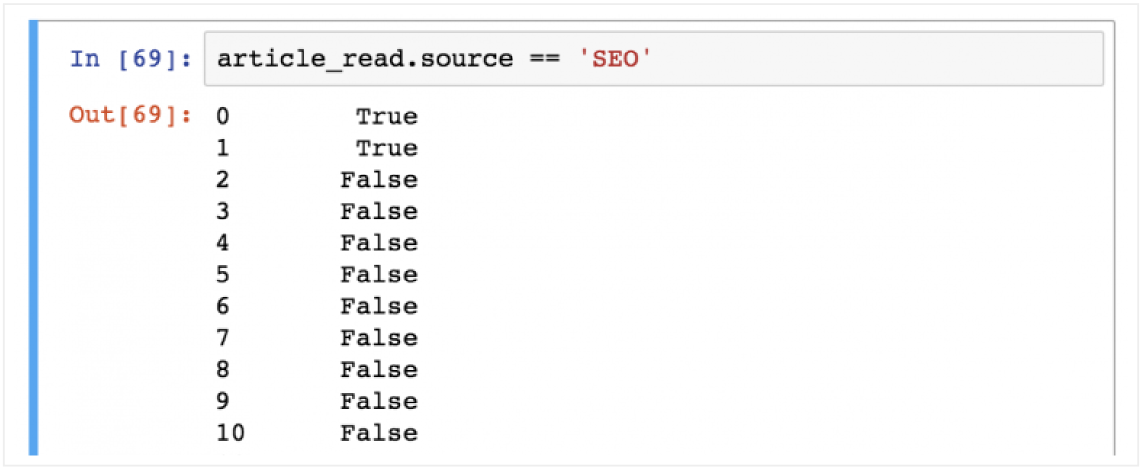
If the previous one was a *bit* tricky, this one will be *really* tricky!

Let's say, you want to see a list of only the users who came from the 'SEO' source. In this case you have to filter for the 'SEO' value in the 'source' column:

`article_read[article_read.source == 'SEO']`

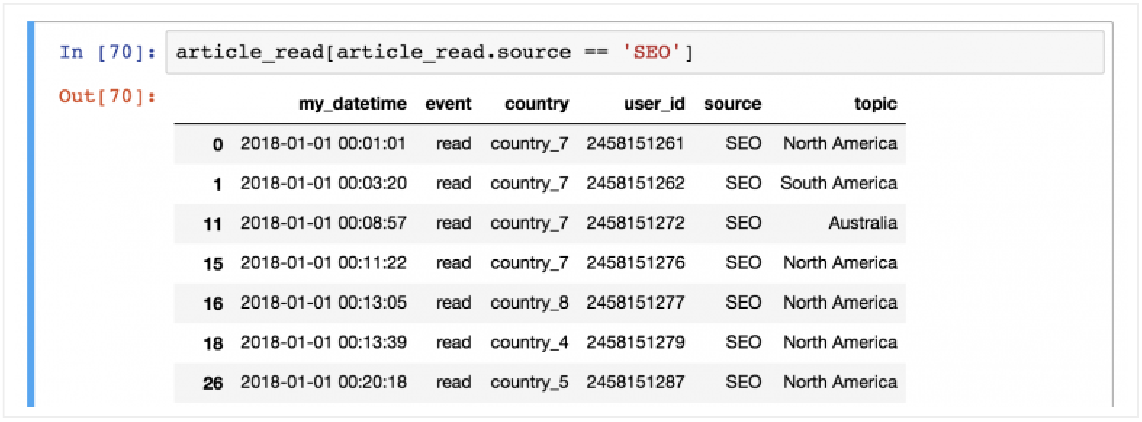
It's worth it to understand how pandas thinks about data filtering:

STEP 1) First, between the bracket frames it evaluates every line: is the `article_read.source` column's value `'SEO'` or not? The results are boolean values (`True` or `False`).



```
In [69]: article_read.source == 'SEO'

Out[69]: 0          True
         1          True
         2         False
         3         False
         4         False
         5         False
         6         False
         7         False
         8         False
         9         False
         10        False
```

STEP 2) Then from the `article_read` table, it prints every row where this value is `True` and doesn't print any row where it's `False`.



```
In [70]: article_read[article_read.source == 'SEO']

Out[70]:
```

	my_datetime	event	country	user_id	source	topic
0	2018-01-01 00:01:01	read	country_7	2458151261	SEO	North America
1	2018-01-01 00:03:20	read	country_7	2458151262	SEO	South America
11	2018-01-01 00:08:57	read	country_7	2458151272	SEO	Australia
15	2018-01-01 00:11:22	read	country_7	2458151276	SEO	North America
16	2018-01-01 00:13:05	read	country_8	2458151277	SEO	North America
18	2018-01-01 00:13:39	read	country_4	2458151279	SEO	North America
26	2018-01-01 00:20:18	read	country_5	2458151287	SEO	North America

Does it look over-complicated? Maybe. But this is the way it is, so let's just learn it because you will use this a lot! 😉

Functions can be used after each other

It's very important to understand that pandas's logic is very linear (compared to SQL, for instance). So if you apply a function, you can always apply another one on it. In this case, the input of the latter function will always be the output of the previous function.

E.g. combine these two selection methods:

```
article_read.head()[['country', 'user_id']]
```

This line first selects the first 5 rows of our data set. And then it takes only the 'country' and the 'user_id' columns.

Could you get the same result with a different chain of functions? Of course you can:

```
article_read[['country', 'user_id']].head()
```

In this version, you select the columns first, then take the first five rows. The result is the same – the order of the functions (and the execution) is different.

One more thing. What happens if you replace the 'article_read' value with the original `read_csv()` function:

```
pd.read_csv('pandas_tutorial_read.csv', delimiter=';', names =
['my_datetime', 'event', 'country', 'user_id', 'source', 'topic'])
[['country', 'user_id']].head()
```

This will work, too – only it's ugly (and inefficient). **But it's really important that you understand that working with pandas is nothing but applying the right functions and methods, one by one.**

Test yourself!

As always, here's a short assignment to test yourself! Solve it, so the content of this article can sink in better!

Select the user_id, the country and the topic columns for the users who are from country_2! Print the first five rows only!

Okay, go ahead and solve it!

.

.

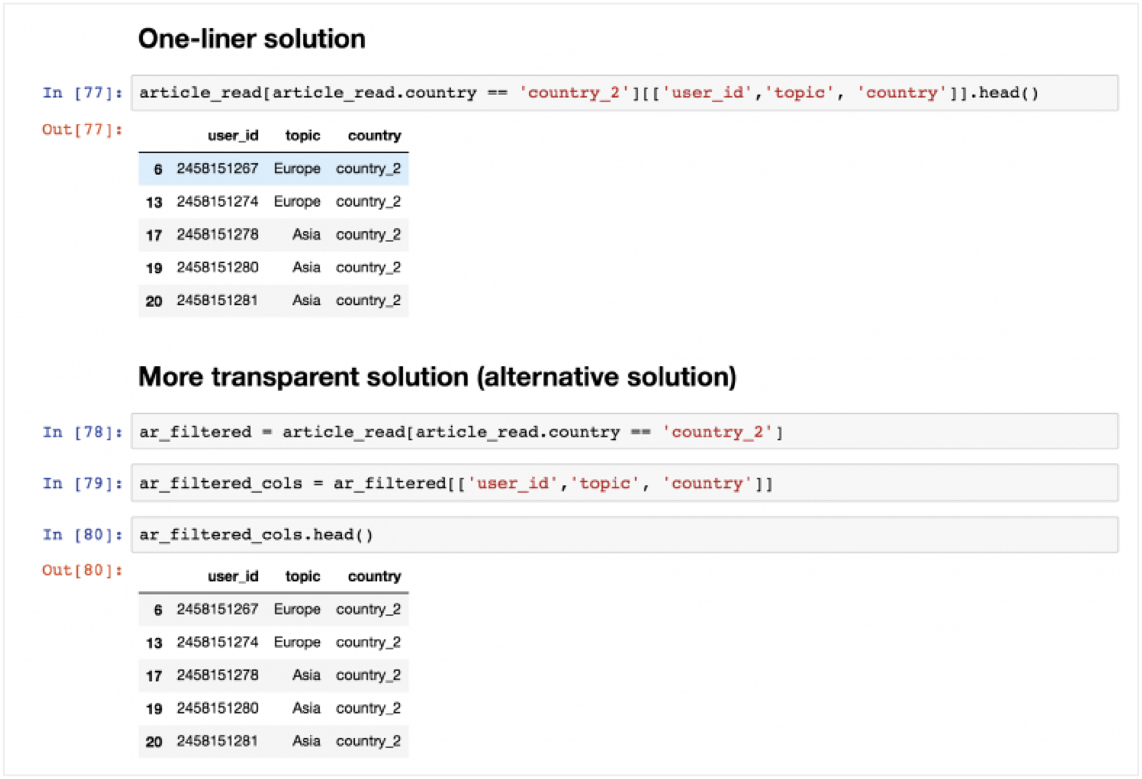
.

And here's my solution!
It can be a one-liner:

```
article_read[article_read.country == 'country_2']
[['user_id','topic', 'country']].head()
```

Or, to be more transparent, you can break this into more lines:

```
ar_filtered = article_read[article_read.country == 'country_2']
ar_filtered_cols = ar_filtered[['user_id','topic', 'country']]
ar_filtered_cols.head()
```



One-liner solution

```
In [77]: article_read[article_read.country == 'country_2'][['user_id','topic', 'country']].head()
Out[77]:
         user_id   topic    country
  6   2458151267  Europe  country_2
 13   2458151274  Europe  country_2
 17   2458151278    Asia  country_2
 19   2458151280    Asia  country_2
 20   2458151281    Asia  country_2
```

More transparent solution (alternative solution)

```
In [78]: ar_filtered = article_read[article_read.country == 'country_2']

In [79]: ar_filtered_cols = ar_filtered[['user_id','topic', 'country']]

In [80]: ar_filtered_cols.head()
Out[80]:
         user_id   topic    country
  6   2458151267  Europe  country_2
 13   2458151274  Europe  country_2
 17   2458151278    Asia  country_2
 19   2458151280    Asia  country_2
 20   2458151281    Asia  country_2
```

Either way, the logic is the same. First you take your original dataframe (`article_read`), then you filter for the rows where the country value is country_2 (`[article_read.country == 'country_2']`), then you take the three columns that were required (`[['user_id','topic', 'country']]`) and eventually you take the first five rows only (`.head()`).

Conclusion

You are done with the first episode of my pandas tutorial series! Great job! In the next article, you can learn more about the different aggregation methods (e.g. sum, mean, max, min) and about grouping (so basically about segmentation). Stay with me: Pandas Tutorial, Episode 2!

- If you want to learn more about how to become a data scientist, take my 50-minute video course: How to Become a Data Scientist. (It's free!)

- Also check out my 6-week online course: The Junior Data Scientist's First Month video course.
- Or simply subscribe to my Newsletter.

Cheers,

Tomi Mester

Help your friends see this article:

f 120 in 0 0

July 10, 2018 In Coding In Data Science and Analytics
#analytics #data #data analysis #data coding #data science #learn data science #learn python #learn to code #pandas #python #python3 #statistics #tomi mester

← PREVIOUS POST NEXT POST →

4 Comments

ygautomo
FEBRUARY 14, 2019

Hi Tomi Mester,

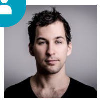
Nice tutorial about Pandas.

For last command "article_read[article_read.country == 'country_2'][['user_id','topic', 'country']].head()"

I guess it would be better if rewrite as
article_read[[‘user_id’,’topic’, ‘country’]][article_read.country == ‘country_2’].head()
as sames as sql notation SELECT [field] FROM [table] WHERE [filter].
Just curious, because my background as SQL developer and i’m little bit confusing about pandas syntax.

Thanks.

REPLY



Tomi Mester
MARCH 6, 2019

hey,

well, the logic in pandas is more linear than in SQL (it’s more “inside-out” in SQL).
Actually, both version will work fine.

Your version might be a little bit more efficient (in terms of calculation time) though — because of the particular order you use (column selection first – then filtering for the rows), but we are talking about milliseconds here. 🙂

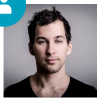
Cheers,
Tomi

REPLY

Andres Cardenas
MARCH 1, 2019

Excellent tutorial. I am just starting with pandas.
Thanks

REPLY



Tomi Mester

MARCH 6, 2019

Great stuff! Way to go! 😉

REPLY

Leave a Reply

COMMENT

NAME *

EMAIL *

WEBSITE

Post Comment