How do I get the row count of a Pandas DataFrame?

Asked 10 years ago Modified 1 month ago Viewed 3.9m times

1776

How do I get the number of rows of a pandas dataframe `df`?

python pandas dataframe

Share Improve this question Follow

edited Mar 28, 2022 at 11:49

Mateen Ulhaq
23.6k ● 16 ● 95 ● 132

asked Apr 11, 2013 at 8:14

yemu
25.5k ● 10 ● 32 ● 29

21 ok I found out, i should have called method not check property, so it should be df.count() no df.count – yemu Apr 11, 2013 at 8:15

98 ^ Dangerous! Beware that `df.count()` will only return the count of non-NA/NaN rows for each column. You should use `df.shape[0]` instead, which will always correctly tell you the number of rows. – smci Apr 18, 2014 at 12:04

6 Note that df.count will not return an int when the dataframe is empty (e.g., pd.DataFrame(columns=["Blue","Red").count is not 0) – Marcelo Bielsa Sep 1, 2015 at 3:32

1 could use df.info() so you get row count (# entries), number of non-null entries in each column, dtypes and memory usage. Good complete picture of the df. If you're looking for a number you can use programatically then df.shape[0]. – MikeB2019x May 4, 2022 at 20:06

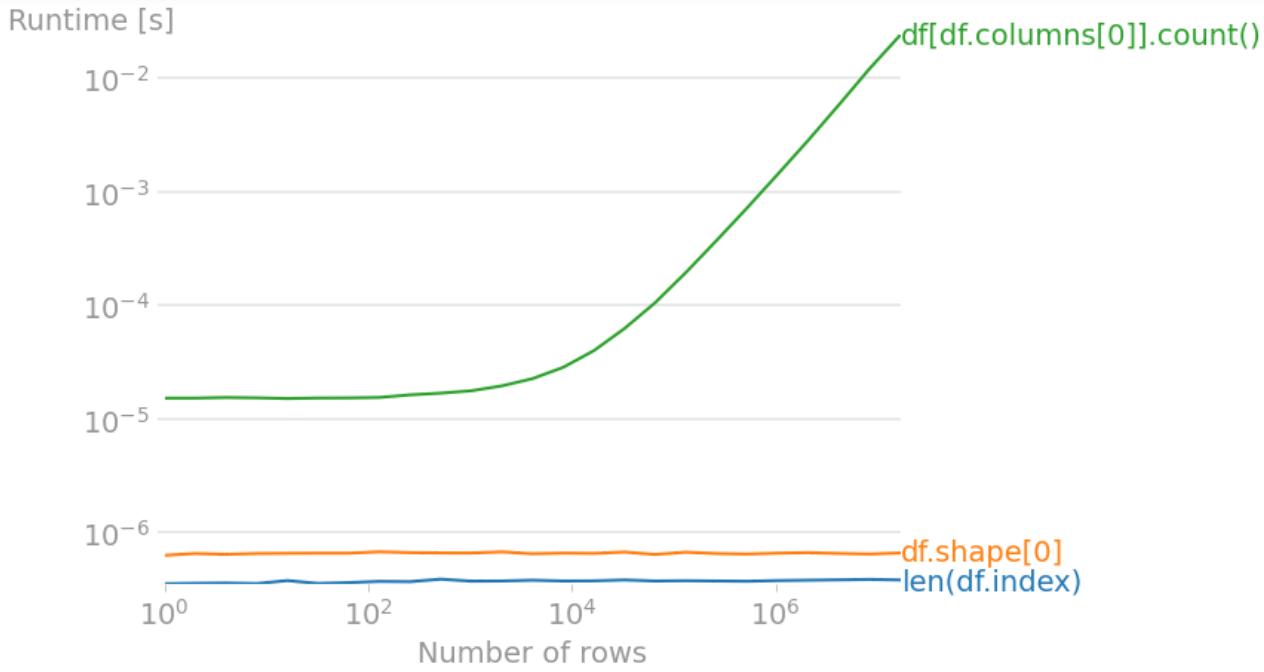
18 Answers

Sorted by: Highest score (default)

2640

For a dataframe `df`, one can use any of the following:

- `len(df.index)`
- `df.shape[0]`
- `df[df.columns[0]].count()` (== number of non-NaN values in first column)



Code to reproduce the plot:

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

perfplot.save(
    "out.png",
    setup=lambda n: pd.DataFrame(np.arange(n * 3).reshape(n, 3)),
    n_range=[2**k for k in range(25)],
    kernels=[
        lambda df: len(df.index),
        lambda df: df.shape[0],
        lambda df: df[df.columns[0]].count(),
    ],
    labels=["len(df.index)", "df.shape[0]", "df[df.columns[0]].count()"],
    xlabel="Number of rows",
)
```

Share Improve this answer Follow

edited Oct 6, 2021 at 23:48
Mateen Ulhaq
23.6k ● 16 ● 95 ● 132

answered Apr 11, 2013 at 8:24
root
75.4k ● 25 ● 108 ● 119

28 There's one good reason why to use `shape` in interactive work, instead of len(df): Trying out different filtering, I often need to know how many items remain. With shape I can see that just by adding .shape after my filtering. With len() the editing of the command-line becomes much more cumbersome, going back and forth. – K.-Michael Aye Feb 25, 2014 at 4:51

12 Won't work for OP, but if you just need to know whether the dataframe is empty, `df.empty` is the best option. – itschoonhoven Mar 16, 2016 at 21:26

3 times slower, taking 1.17 microseconds. did I miss something? @root – T.G. May 22, 2017 at 18:34

13 (3,3) matrix is bad example as it does not show the order of the shape tuple – xaedes Aug 15, 2017 at 16:42

8 How is `df.shape[0]` faster than `len(df)` or `len(df.columns)`? Since **1 ns** (nanosecond) = **1000 µs** (microsecond), therefore 1.17µs = 1170ns, which means it's roughly 3 times slower than 381ns – itsjef Mar 24, 2018 at 3:19

471

Suppose `df` is your dataframe then:

```
count_row = df.shape[0]  # Gives number of rows
count_col = df.shape[1]  # Gives number of columns
```

Or, more succinctly,

```
r, c = df.shape
```

Share Improve this answer Follow

edited Feb 8, 2021 at 15:14

Peter Mortensen
31k ● 21 ● 105 ● 126

answered Feb 20, 2016 at 13:30

Nasir Shah
4,953 ● 1 ● 10 ● 9

19 If the data set is large, len (df.index) is significantly faster than df.shape[0] if you need only row count. I tested it. – Sumit Pokhrel Jan 2, 2020 at 14:47

1 Why i do not have shape method on my DataFrame? – Ardalan Shahgholi Oct 6, 2020 at 20:00

1 @ArdalanShahgholi it's probably because what was returned is a series, which is always 1 dimensional. Therefore, only `len(df.index)` will work – Connor Aug 1, 2021 at 23:54

@Connor I need to have Number of rows and number of Columns from my DF. In my DF also i have a select it means i have a table and now the question is why i do not have SHAPE function on my DF? – Ardalan Shahgholi Aug 17, 2021 at 18:41

Great question, make it a separate question on SO, share what you've tried and what you see as a result (give a full working set of code that's simple for others to replicate) and then share the link to that question here. I'll see if I can help – Connor Aug 19, 2021 at 20:06

263

Use `len(df)` :-).

`__len__()` is documented with "Returns length of index".

Timing info, set up the same way as in root's answer:

```
In [8]: timeit len(df)
1000000 loops, best of 3: 573 ns per loop
```

Due to one additional function call, it is of course correct to say that it is a bit slower than calling `len(df.index)` directly. But this should not matter in most cases. I find `len(df)` to be quite readable.

Share Improve this answer Follow edited Jul 21, 2021 at 10:17 answered Aug 19, 2013 at 15:02

Dr. Jan-Philip Gehrcke
32.2k ● 14 ● 81 ● 130

155

How do I get the row count of a Pandas DataFrame?

This table summarises the different situations in which you'd want to count something in a DataFrame (or Series, for completeness), along with the recommended method(s).

Operation	Series `s`	DataFrame `df`
Row count	`len(s)`, `s.size`, `len(s.index)`	`len(df)`, `df.shape[0]`, `len(df.index)`
Column count	N/A	`df.shape[1]`, `len(df.columns)`
Non-null row count (ignore NaNs)	`s.count()`	`df.count()` [1]
Row count per group (via `groupby`)	`s.groupby(...).size()`	`df.groupby(...).size()` [2]
Non-null row count per group	`s.groupby(...).count()`	`df.groupby(...).count()` [3]

Footnotes

1. `DataFrame.count` returns counts for each column as a `Series` since the non-null count varies by column.
2. `DataFrameGroupBy.size` returns a `Series`, since all columns in the same group share the same row-count.
3. `DataFrameGroupBy.count` returns a `DataFrame`, since the non-null count could differ across columns in the same group. To get the group-wise non-null count for a specific column, use `df.groupby(...)['x'].count()` where "x" is the column to count.

Minimal Code Examples

Below, I show examples of each of the methods described in the table above. First, the setup -

```
df = pd.DataFrame({
    'A': list('aabbc'), 'B': ['x', 'x', np.nan, 'x', np.nan]})
s = df['B'].copy()

df

   A    B
0  a    x
1  a    x
2  b  NaN
3  b    x
4  c  NaN

s

0      x
1      x
2    NaN
3      x
4    NaN
Name: B, dtype: object
```

Row Count of a DataFrame: `len(df)`, `df.shape[0]`, or `len(df.index)`

```
len(df)
# 5

df.shape[0]
# 5

len(df.index)
# 5
```

It seems silly to compare the performance of constant time operations, especially when the difference is on the level of "seriously, don't worry about it". But this seems to be a trend with other answers, so I'm doing the same for completeness.

Of the three methods above, `len(df.index)` (as mentioned in other answers) is the fastest.

> **Note**
>
> - All the methods above are constant time operations as they are simple attribute lookups.
> - `df.shape` (similar to `ndarray.shape`) is an attribute that returns a tuple of `(# Rows,`

Column Count of a DataFrame: `df.shape[1]`, `len(df.columns)`

```
df.shape[1]
# 2

len(df.columns)
# 2
```

Analogous to `len(df.index)`, `len(df.columns)` is the faster of the two methods (but takes more characters to type).

Row Count of a Series: `len(s)`, `s.size`, `len(s.index)`

```
len(s)
# 5

s.size
# 5

len(s.index)
# 5
```

`s.size` and `len(s.index)` are about the same in terms of speed. But I recommend `len(df)`.

> **Note** `size` is an attribute, and it returns the number of elements (=count of rows for any Series). DataFrames also define a size attribute which returns the same result as `df.shape[0] * df.shape[1]`.

Non-Null Row Count: `DataFrame.count` and `Series.count`

The methods described here only count non-null values (meaning NaNs are ignored).

Calling `DataFrame.count` will return non-NaN counts for *each* column:

```
df.count()

A    5
B    3
dtype: int64
```

For Series, use `Series.count` to similar effect:

```
s.count()
# 3
```

For `DataFrames`, use `DataFrameGroupBy.size` to count the number of rows per group.

```
df.groupby('A').size()

A
a    2
b    2
c    1
dtype: int64
```

Similarly, for `Series`, you'll use `SeriesGroupBy.size`.

```
s.groupby(df.A).size()

A
a    2
b    2
c    1
Name: B, dtype: int64
```

In both cases, a `Series` is returned. This makes sense for `DataFrames` as well since all groups share the same row-count.

Group-wise Non-Null Row Count: `GroupBy.count`

Similar to above, but use `GroupBy.count`, not `GroupBy.size`. Note that `size` always returns a `Series`, while `count` returns a `Series` if called on a specific column, or else a `DataFrame`.

The following methods return the same thing:

```
df.groupby('A')['B'].size()
df.groupby('A').size()

A
a    2
b    2
c    1
Name: B, dtype: int64
```

Meanwhile, for `count`, we have

```
df.groupby('A').count()

   B
A
a  2
b  1
c  0
```

```
df.groupby('A')['B'].count()

A
a    2
b    1
c    0
Name: B, dtype: int64
```

Called on a specific column.

Share Improve this answer Follow edited Jan 31, 2022 at 2:32 answered Mar 30, 2019 at 19:55

cs95
369k ● 94 ● 683 ● 735

`s.shape[0]` work for row count in a series. – rubengavidia0x Jan 26, 2022 at 21:23

Hi, could you take a look at this question stackoverflow.com/questions/70954791/... – Aaditya Ura Feb 2, 2022 at 11:34

TL;DR use `len(df)`

76

`len()` returns the number of items(the length) of a list object(also works for dictionary, string, tuple or range objects). So, for getting row counts of a DataFrame, simply use `len(df)`. For more about *len* function, see the official page.

Alternatively, you can access all rows and all columns with `df.index`, and `df.columns`,respectively. Since you can use the `len(anyList)` for getting the element numbers, using the `len(df.index)` will give the number of rows, and `len(df.columns)` will give the number of columns.

Or, you can use `df.shape` which returns the number of rows and columns together (as a tuple) where you can access each item with its index. If you want to access the number of rows, only use `df.shape[0]`. For the number of columns, only use: `df.shape[1]`.

Share Improve this answer Follow edited Jun 19, 2022 at 10:07 answered Jun 25, 2016 at 5:23

Memin
3,688 ● 29 ● 28

2 @BrendanMetcalfe, I dont know what might me wrong with your dataframe without seeing the its data. You can check the small script end the end to see, indeed `len` works well for getting row counts. Here is the script onecompiler.com/python/3xc9nuvrx – Memin Sep 22, 2021 at 19:19

I can't wrap my head around, why `df.shape` isn't faster than `len` as it just have to get the `shape` attribute and not call the function ... – CuteBoiCan Nov 1, 2022 at 8:40

24

Apart from the previous answers, you can use `df.axes` to get the tuple with row and column indexes and then use the `len()` function:

```
total_rows = len(df.axes[0])
total_cols = len(df.axes[1])
```

Share Improve this answer Follow

edited Feb 8, 2021 at 15:13

Peter Mortensen
31k ● 21 ● 105 ● 126

answered Aug 19, 2015 at 19:07

Nik
421 ● 5 ● 10

4 This returns index objects, which may or may not be copies of the original, which is wasteful if you are just discarding them after checking the length. Unless you intend to do anything else with the index, **DO NOT USE**. – cs95 Mar 30, 2019 at 20:13

14

...building on Jan-Philip Gehrcke's answer.

The reason why `len(df)` or `len(df.index)` is faster than `df.shape[0]`:

Look at the code. df.shape is a `@property` that runs a DataFrame method calling `len` twice.

```
df.shape??
Type:        property
String form: <property object at 0x1127b33c0>
Source:
# df.shape.fget
@property
def shape(self):
    """
    Return a tuple representing the dimensionality of the DataFrame.
    """
    return len(self.index), len(self.columns)
```

And beneath the hood of len(df)

```
df.__len__??
Signature: df.__len__()
Source:
    def __len__(self):
        """Returns length of info axis, but here we use the index """
        return len(self.index)
File:      ~/miniconda2/lib/python2.7/site-packages/pandas/core/frame.py
Type:      instancemethod
```

`len(df.index)` will be slightly faster than `len(df)` since it has one less function call, but this is


Peter Mortensen
31k ● 21 ● 105 ● 126


debo
309 ● 2 ● 8

1 The syntax highlighting does not seem quite right. Can you fix it? E.g., is this a mixture of output, code, and annotation (not a rhetorical question)? – Peter Mortensen Feb 8, 2021 at 15:22

@PeterMortensen This output is from ipython/jupyter. Executing a function name with two question marks and without the parenthesis will show the function definition. ie for function `len()` you would execute `len??` – debo Apr 8, 2021 at 4:04

10

I come to Pandas from an R background, and I see that Pandas is more complicated when it comes to selecting rows or columns.

I had to wrestle with it for a while, and then I found some ways to deal with:

Getting the number of columns:

```
len(df.columns)
## Here:
# df is your data.frame
# df.columns returns a string. It contains column's titles of the df.
# Then, "len()" gets the length of it.
```

Getting the number of rows:

```
len(df.index) # It's similar.
```

Share Improve this answer Follow

edited Feb 8, 2021 at 15:19

Peter Mortensen
31k ● 21 ● 105 ● 126

answered Sep 29, 2016 at 7:41

Catbuilts
4,455 ● 1 ● 34 ● 28

After using *Pandas* for a while, I think we should go with `df.shape`. It returns the number of rows and columns respectively. – Catbuilts Oct 29, 2018 at 10:16

7

In case you want to get the row count in the middle of a chained operation, you can use:

```
df.pipe(len)
```

Example:

```
row_count = (
```

```
        .pipe(len)
)
```

This can be useful if you don't want to put a long statement inside a `len()` function.

You could use `__len__()` instead but `__len__()` looks a bit weird.

Share Improve this answer Follow

edited Mar 26, 2020 at 7:19

Chris Tang
567 ● 7 ● 18

answered Feb 22, 2018 at 2:58

Allen Qin
19.3k ● 7 ● 50 ● 67

1 It seems pointless to want to "pipe" this operation because there's nothing else you can pipe this into (it returns an integer). I would much rather `count = len(df.reset_index())` than `count = df.reset_index().pipe(len)`. The former is just an attribute lookup without the function call. – cs95 Mar 30, 2019 at 20:15

7

You can do this also:

Let's say `df` is your dataframe. Then `df.shape` gives you the shape of your dataframe i.e `(row,col)`

Thus, assign the below command to get the required

```
row = df.shape[0], col = df.shape[1]
```

Share Improve this answer Follow

edited Feb 8, 2021 at 15:34

Peter Mortensen
31k ● 21 ● 105 ● 126

answered May 12, 2020 at 7:14

Saurav
143 ● 2 ● 4

Or you can directly use `row, col = df.shape` instead if you need to get both at the same them (it's shorter and you do not have to care about indexes). – Nerxis May 17, 2021 at 8:46

4

Either of this can do it (`df` is the name of the DataFrame):

Method 1: Using the `len` function:

`len(df)` will give the number of rows in a DataFrame named `df`.

Method 2: using `count` function:

`df[col].count()` will count the number of rows in a given column `col`.

Share Improve this answer Follow

edited Feb 8, 2021 at 15:31

Peter Mortensen
31k ● 21 ● 105 ● 126

answered Apr 24, 2020 at 16:30

Kiprono Elijah Koech
198 ● 1 ● 5

4 This is a fine answer, but there are already sufficient answers to this question, so this doesn't really add anything. – John Apr 24, 2020 at 18:07

3

For dataframe df, a printed comma formatted row count used while exploring data:

```
def nrow(df):
    print("{:,}".format(df.shape[0]))
```

Example:

```
nrow(my_df)
12,456,789
```

Share Improve this answer Follow

answered Sep 21, 2017 at 1:59

Vlad
2,974 ● 4 ● 24 ● 50

3

When using `len(df)` or `len(df.index)` you might encounter this error:

```
----> 4 df['id'] = np.arange(len(df.index)
TypeError: 'int' object is not callable
```

Solution:

```
lengh = df.shape[0]
```

Share Improve this answer Follow

edited Oct 19, 2022 at 3:03

Peter Mortensen
31k ● 21 ● 105 ● 126

answered Jun 13, 2022 at 12:53

Lorenzo Bassetti
705 ● 8 ● 15

2

For a dataframe `df`:

When you're still writing your code:

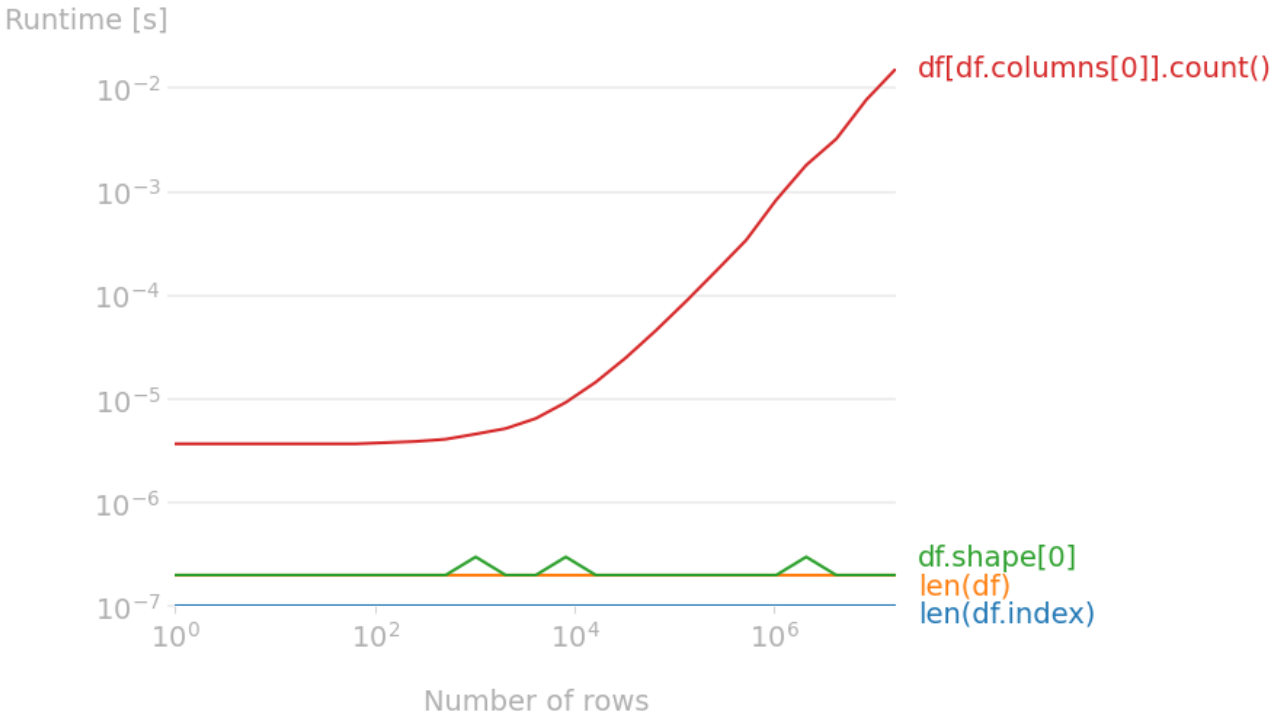
1. `len(df)`

Fastest once your code is done:

- `len(df.index)`

At normal data sizes each option will finish in under a second. So the "fastest" option is actually whichever one lets you work the fastest, which can be `len(df)` or `df.shape[0]` if you already have a subsetted `df` and want to just add `.shape[0]` briefly in an interactive session.

In final optimized code, the fastest runtime is `len(df.index)`.



`df[df.columns[0]].count()` was omitted in the above discussion because no commenter has identified a case where it is useful. It is exponentially slow, and long to type. It provides the number of non-NaN values in the first column.

Code to reproduce the plot:

`pip install pandas perfplot`

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

perfplot.save(
    "out.png",
    setup=lambda n: pd.DataFrame(np.arange(n * 3).reshape(n, 3)),
    n_range=[2**k for k in range(25)],
    kernels=[
        lambda df: len(df.index),
```

```
    labels=["len(df.index)", "df.shape[0]", "df[df.columns[0]].count()"],
    xlabel="Number of rows",
)
```

Share Improve this answer Follow

answered Feb 22 at 18:23

Jimmy Carter
154 ● 7

I've tried twice to improve the accepted answer and been rejected both times. The accepted answer is unclear and pointlessly verbose, not telling people the fastest right of the bat. It also doesn't mention `len(df)` nor any purpose for `df[df.columns[0]].count()`. – Jimmy Carter Feb 22 at 18:25

0

An alternative method to finding out the amount of rows in a dataframe which I think is the most readable variant is `pandas.Index.size`.

Do note that, as I commented on the accepted answer,

> Suspected `pandas.Index.size` would actually be faster than `len(df.index)` but `timeit` on my computer tells me otherwise (~150 ns slower per loop).

Share Improve this answer Follow

edited Feb 8, 2021 at 15:29

Peter Mortensen
31k ● 21 ● 105 ● 126

answered Feb 24, 2020 at 15:14

gosuto
5,294 ● 4 ● 36 ● 57

0

I'm not sure if this would work (data *could* be omitted), but this may work:

```
*dataframe name*.tails(1)
```

and then using this, you could find the number of rows by running the code snippet and looking at the row number that was given to you.

Share Improve this answer Follow

edited Feb 8, 2021 at 15:31

Peter Mortensen
31k ● 21 ● 105 ● 126

answered Apr 5, 2020 at 19:49

Abhiraam Eranti
350 ● 4 ● 19

0

len(df.index) would work the fastest of all the ways listed

Share Improve this answer Follow

answered Aug 17, 2022 at 13:13

Zaid Parkar

1 Why would that be? And do you have some performance measurements (incl. conditions, like hardware platform, all with versions)? – Peter Mortensen Oct 19, 2022 at 1:38

-1

Think, the dataset is "data" and name your dataset as " data_fr " and number of rows in the data_fr is "nu_rows"

```
#import the data frame. Extention could be different as csv,xlsx or etc.
data_fr = pd.read_csv('data.csv')

#print the number of rows
nu_rows = data_fr.shape[0]
print(nu_rows)
```

Share Improve this answer Follow edited Feb 16, 2021 at 20:16

answered Jan 2, 2021 at 23:04

SamithaP

92 ● 1 ● 1 ● 13

What is the conclusion of that first sentence? – Peter Mortensen Oct 19, 2022 at 1:41

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.