Import CSV file as a pandas DataFrame

Asked 6 years, 9 months ago Active 27 days ago Viewed 129k times

69

19

What's the Python way to read in a CSV file into a pandas DataFrame (which I can then use for statistical operations, can have differently-typed columns, etc.)?

My CSV file `"value.txt"` has the following content:

```
Date,"price","factor_1","factor_2"
2012-06-11,1600.20,1.255,1.548
2012-06-12,1610.02,1.258,1.554
2012-06-13,1618.07,1.249,1.552
2012-06-14,1624.40,1.253,1.556
2012-06-15,1626.15,1.258,1.552
2012-06-16,1626.15,1.263,1.558
2012-06-17,1626.15,1.264,1.572
```

In R we would read this file in using:

```
price <- read.csv("value.txt")
```

and that would return an R data.frame:

```
> price <- read.csv("value.txt")
> price
     Date   price factor_1 factor_2
1  2012-06-11 1600.20    1.255    1.548
2  2012-06-12 1610.02    1.258    1.554
3  2012-06-13 1618.07    1.249    1.552
4  2012-06-14 1624.40    1.253    1.556
5  2012-06-15 1626.15    1.258    1.552
6  2012-06-16 1626.15    1.263    1.558
7  2012-06-17 1626.15    1.264    1.572
```

Is there a Pythonic way to get the same functionality?

python pandas csv dataframe

edited Jun 30 at 19:48 — smci — 17.3k 6 83 113

asked Jan 16 '13 at 18:50 — mazlor — 620 3 11 19

will you commend on what's special with is dataframe and what statistical operation can you do with it? – LWZ Jan 16 '13 at 19:53

here to 'sell' the benefits package. The wider question *"What are the benefits of using a pandas dataframe over plain Python array/list-of-list?"* has many benefits, way too many to list here... – smci Jun 30 at 19:49

8 Answers

125

[pandas](#) to the rescue:

```
import pandas as pd
print pd.read_csv('value.txt')
```

```
         Date    price  factor_1  factor_2
0  2012-06-11  1600.20     1.255     1.548
1  2012-06-12  1610.02     1.258     1.554
2  2012-06-13  1618.07     1.249     1.552
3  2012-06-14  1624.40     1.253     1.556
4  2012-06-15  1626.15     1.258     1.552
5  2012-06-16  1626.15     1.263     1.558
6  2012-06-17  1626.15     1.264     1.572
```

This returns pandas [DataFrame](#) that is similar to `R's` .

edited Jan 16 '13 at 19:01

answered Jan 16 '13 at 18:56

root
44.7k 16 85 107

6

Here's an alternative to pandas library using Python's built-in [csv module](#).

```
import csv
from pprint import pprint
with open('foo.csv', 'rb') as f:
    reader = csv.reader(f)
    headers = reader.next()
    column = {h:[] for h in headers}
    for row in reader:
        for h, v in zip(headers, row):
            column[h].append(v)
    pprint(column)    # Pretty printer
```

will print

```
{'Date': ['2012-06-11',
          '2012-06-12',
          '2012-06-13',
```

```
'1610.02',
'1618.07',
'1624.40',
'1626.15',
'1626.15',
'1626.15']}
```

edited Jan 16 '13 at 19:34

answered Jan 16 '13 at 19:20

sidi
1,698 11 20

4

To read a CSV file as a pandas DataFrame, you'll need to use `pd.read_csv` .

But this isn't where the story ends; data exists in many different formats and is stored in different ways so you will often need to pass additional parameters to `read_csv` to ensure your data is read in properly.

Here's a table listing common scenarios encountered with CSV files along with the appropriate argument you will need to use. You will usually need all or some combination of the arguments below to read in *your* data.

Scenario Example	Argument
Read CSV with different separator[1] `read_csv(..., sep=';')`	sep/delimiter
Read CSV with tab/whitespace separator `read_csv(..., delim_whitespace=True)`	delim_whitespace
Fix UnicodeDecodeError while reading[2] `read_csv(..., encoding='latin-1')`	encoding
Read CSV without headers[3] `read_csv(..., header=False, names=['x', 'y', 'z'])`	header and names
Specify which column to set as the index[4] `read_csv(..., index_col=[0])`	index_col
Read subset of columns `read_csv(..., usecols=['x', 'y'])`	usecols
Numeric data is in European format (eg., 1.234,56) `read_csv(..., thousands='.', decimal=',')`	thousands and decimal

Footnotes

1. By default, `read_csv` uses a C parser engine for performance. The C parser can only handle single character separators. If your CSV has a multi-character separator, you will need to

your data is likely to fit into one of these.

3. `header=False` specifies that the first row in the CSV is a data row rather than a header row, and the `names=[...]` allows you to specify a list of column names to assign to the DataFrame when it is created.
4. "Unnamed: 0" occurs when a DataFrame with an un-named index is saved to CSV and then re-read after. Instead of having to fix the issue while reading, you can also fix the issue when writing by using

```
df.to_csv(..., index=False)
```

There are other arguments I've not mentioned here, but these are the ones you'll encounter most frequently.

edited Jun 30 at 20:39

answered May 21 at 5:33

cs95
171k 34 253 315

1 Can you please repaste/reformat the table-as-image as Unicode text using e.g. ozh.github.io/ascii-tables ? Otherwise it's hard to read, and won't be indexed for content search, or findable with plain old Ctrl-F text search. – smci Jun 30 at 19:54

@smci I agree what you said about indexing (plus the fact that images cannot be edited by other people) but it does not look nearly as good and requires scrolling which makes it hard to follow. – cs95 Jun 30 at 20:17

anyway you need to rewrite it to be user-friendly to a new user; instead of incomprehensible stuff like *'Avoid "Unnamed: 0"'* it should say something plain English like *' `index_col` : tell pandas which column(s) to use as index for your dataframe'* – smci Jun 30 at 20:19

@smci Looks like ASCII tables don't play nicely with superscripts -- it messes up the table formatting. – cs95 Jun 30 at 20:25

cs95: I recommended Unicode, not ASCII. Unicode works fine. So just change the default in the pulldown on ozh.github.io/ascii-tables – smci Jun 30 at 20:26

2

```
import pandas as pd
df = pd.read_csv('/PathToFile.txt', sep = ',')
```

This will import your .txt or .csv file into a DataFrame.

answered Sep 7 at 16:09

Rishabh
21 1

Replace the file target location, with where your data set is found, refer this url https://medium.com/@kanchanardj/jargon-in-python-used-in-data-science-to-laymans-language-part-one-12ddfd31592f

answered Sep 20 at 6:51

Dulangi_Kanchana
125 6

0

You can use the csv module found in the python standard library to manipulate CSV files.

example:

```
import csv
with open('some.csv', 'rb') as f:
    reader = csv.reader(f)
    for row in reader:
        print row
```

answered Jan 16 '13 at 19:03

KurzedMetal
9,986 4 30 56

-0. Coming from R, mazlor wouldn't be looking for the `csv` module as it is too low level. `pandas` provides the requested level of abstraction. – Steven Rumbalski Jan 16 '13 at 19:10

...in addition it does read the data into a useful Python object such as a numpy array... – Paul Hiemstra Jan 16 '13 at 19:16

0

```
%cd C:\Users\asus\Desktop\python
import pandas as pd
df = pd.read_csv('value.txt')
df.head()
    Date    price   factor_1    factor_2
0   2012-06-11  1600.20 1.255   1.548
1   2012-06-12  1610.02 1.258   1.554
2   2012-06-13  1618.07 1.249   1.552
3   2012-06-14  1624.40 1.253   1.556
4   2012-06-15  1626.15 1.258   1.552
```

answered Sep 5 at 13:59

Kamal
1 3

```
with open( value.txt ,  'r' ) as f:
    csv_reader = reader(f)
    num = '  '
    for row in csv_reader:
        print num, '\t'.join(row)
        if num == '  ':
            num=0
        num=num+1
```

Not as compact, but it does the job:

```
   Date price   factor_1    factor_2
1 2012-06-11    1600.20 1.255   1.548
2 2012-06-12    1610.02 1.258   1.554
3 2012-06-13    1618.07 1.249   1.552
4 2012-06-14    1624.40 1.253   1.556
5 2012-06-15    1626.15 1.258   1.552
6 2012-06-16    1626.15 1.263   1.558
7 2012-06-17    1626.15 1.264   1.572
```

answered Jan 16 '13 at 19:12

Lee-Man
368 1 8

1 This does not answer the OP's question as it does not read the csv data into a Python object. – Paul Hiemstra Jan 16 '13 at 19:15

maybe replace the num with enumerate in the for loop? – LWZ Jan 16 '13 at 19:35

@PaulHiemstra, OP did not mention "object", but ask for ease. Still, I suspect the "pandas" approach better fits what was being asked for. – Lee-Man Jan 16 '13 at 21:32