How to read a text file into a list or an array with Python

Asked 9 years, 2 months ago Modified 5 months ago Viewed 1.1m times

210

50

I am trying to read the lines of a text file into a list or array in python. I just need to be able to individually access any item in the list or array after it is created.

The text file is formatted as follows:

```
0,0,200,0,53,1,0,255,...,0.
```

Where the `...` is above, there actual text file has hundreds or thousands more items.

I'm using the following code to try to read the file into a list:

```
text_file = open("filename.dat", "r")
lines = text_file.readlines()
print lines
print len(lines)
text_file.close()
```

The output I get is:

```
['0,0,200,0,53,1,0,255,...,0.']
1
```

Apparently it is reading the entire file into a list of just one item, rather than a list of individual items. What am I doing wrong?

python arrays list text

Share Improve this question Follow

edited Jun 21, 2018 at 19:19

codeforester

33.4k ● 13 ● 91 ● 121

asked Feb 3, 2013 at 19:19

user2037744

2,109 ● 2 ● 12 ● 3

1 Just as a note. It looks like this question should be rephrased as how to read a csv file into a list in Python. But I defer to the OP's original intentions over 4 years ago which I don't know. – demongolem Jun 29, 2017 at 13:32

Related, likely duplicate of: stackoverflow.com/questions/7844118/..., stackoverflow.com/questions/24662571/python-import-csv-to-list – AMC Feb 15, 2020 at 1:12

1 Does this answer your question? How to convert comma-delimited string to list in Python? – AMC Feb 15, 2020 at 1:12

1 In fact, looking at the top answer, this is a duplicate of stackoverflow.com/questions/3277503/.... – AMC Feb 15, 2020 at 1:37

Highest score (default)

You will have to split your string into a list of values using `split()`

175

So,

```
lines = text_file.read().split(',')
```

EDIT: I didn't realise there would be so much traction to this. Here's a more idiomatic approach.

```
import csv
with open('filename.csv', 'r') as fd:
    reader = csv.reader(fd)
    for row in reader:
        # do something
```

Share Improve this answer Follow

edited Sep 6, 2020 at 8:35

answered Feb 3, 2013 at 19:29

Achrome
7,493 ● 14 ● 33 ● 45

3 I think that this answer could be bettered... If you consider a multiline `.csv` file (as mentioned by the OP), e.g., a file containing the alphabetic characters 3 by row (`a,b,c` , `d,e,f` , etc) and apply the procedure described above what you get is a list like this: `['a', 'b', 'c\nd', 'e', ... ]` (note the item `'c\nd'`). I'd like to add that, the above problem notwistanding, this procedure collapses data from individual rows in a single mega-list, usually not what I want when processing a record-oriented data file. – gboffi Jan 24, 2017 at 18:52

split is going to leave the newlines. Don't do this, use `csv` module or some other existing parser – Jean-François Fabre ♦ May 24, 2020 at 17:09

You can also use numpy loadtxt like

61

```
from numpy import loadtxt
lines = loadtxt("filename.dat", comments="#", delimiter=",", unpack=False)
```

Share Improve this answer Follow

answered Feb 4, 2013 at 10:46

Thiru
2,993 ● 7 ● 34 ● 49

2 I need this too. I noticed on a Raspberry Pi that numpy works really slow. For this application I reverted to open a file and read it line by line. – August Sep 14, 2013 at 15:51

4 This is useful for specifying format too, via `dtype : data-type` parameter. docs.scipy.org/doc/numpy/reference/generated/numpy.loadtxt.html Pandas read_csv is very easy to use. But I did not see a way to specify format for it. It was reading floats from my file, whereas I needed string. Thanks @Thiru for showing loadtxt. – Ozgur Ozturk Feb 6, 2017 at 16:25

1 if txt files contains strings, then dtype should be specified, so it should be like lines = loadtxt("filename.dat", dtype=str, comments="#", delimiter=",", unpack=False) – Alex M981 Sep 20, 2018 at 10:29

25

So you want to create a list of lists... We need to start with an empty list

```
list_of_lists = []
```

next, we read the file content, line by line

```
with open('data') as f:
    for line in f:
        inner_list = [elt.strip() for elt in line.split(',')]
        # in alternative, if you need to use the file content as numbers
        # inner_list = [int(elt.strip()) for elt in line.split(',')]
        list_of_lists.append(inner_list)
```

A common use case is that of columnar data, but our units of storage are the rows of the file, that we have read one by one, so you may want to *transpose* your list of lists. This can be done with the following idiom

```
by_cols = zip(*list_of_lists)
```

Another common use is to give a name to each column

```
col_names = ('apples sold', 'pears sold', 'apples revenue', 'pears revenue')
by_names = {}
for i, col_name in enumerate(col_names):
    by_names[col_name] = by_cols[i]
```

so that you can operate on homogeneous data items

```
 mean_apple_prices = [money/fruits for money, fruits in
                      zip(by_names['apples revenue'], by_names['apples_sold'])]
```

Most of what I've written can be speeded up using the `csv` module, from the standard library. Another third party module is `pandas`, that lets you automate most aspects of a typical data analysis (but has a number of dependencies).

Update While in Python 2 `zip(*list_of_lists)` returns a different (transposed) list of lists, in Python 3 the situation has changed and `zip(*list_of_lists)` returns *a zip object* that is not subscriptable.

If you *need* indexed access you can use

```
by_cols = list(zip(*list_of_lists))
```

that gives you a list of lists in both versions of Python.

On the other hand, if you *don't need* indexed access and what you want is just to build a dictionary indexed by column names, a zip object is just fine...

```
columns = zip(*((x.strip() for x in line.split(',')) for line in file)))
d = {}
for name, column in zip(names, columns): d[name] = column
```

Share Improve this answer Follow

edited Aug 15, 2017 at 12:50

answered Nov 20, 2014 at 17:47

gboffi
19.3k ● 5 ● 52 ● 79

The OP said they wanted a list of data from a CSV, not a "list of lists". Just use the `csv` module... – Blairg23 Mar 14, 2018 at 5:29

8

This question is asking how to read the comma-separated value contents from a file into an iterable list:

`0,0,200,0,53,1,0,255,...,0.`

The easiest way to do this is with the `csv` module as follows:

```
import csv
with open('filename.dat', newline='') as csvfile:
    spamreader = csv.reader(csvfile, delimiter=',')
```

Now, you can easily iterate over `spamreader` like this:

```
for row in spamreader:
    print(', '.join(row))
```

See documentation for more examples.

Share Improve this answer Follow

answered Mar 14, 2018 at 5:36

Blairg23
9,950 ● 5 ● 65 ● 67

0

Im a bit late but you can also read the text file into a dataframe and then convert corresponding column to a list.

```
lista=pd.read_csv('path_to_textfile.txt', sep=",", header=None)[0].tolist()
```

example.

```
lista=pd.read_csv('data/holdout.txt',sep=',',header=None)[0].tolist()
```

Note: the column name of the corresponding dataframe will be in the form of integers and i choose 0 because i was extracting only the first column

Shreyas H.V
19 • 3

0

Better this way,

```
def txt_to_lst(file_path):

   try:
       stopword=open(file_path,"r")
       lines = stopword.read().split('\n')
       print(lines)

   except Exception as e:
       print(e)
```

Share Improve this answer Follow

answered Oct 21, 2021 at 7:39

Pranab Sarkar
21 • 4

1 Your answer could be improved with additional supporting information. Please edit to add further details, such as citations or documentation, so that others can confirm that your answer is correct. You can find more information on how to write good answers in the help center. – Community Bot Oct 21, 2021 at 7:53

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.