USING STRINGIO TO READ DELIMITED TEXT FILES INTO NUMPY

In this tutorial, we'll show you how to read delimited text data into a NumPy array using the StringIO package.

(This tutorial is part of our Pandas Guide. Use the right-hand menu to navigate.)

Data we used

We will read this crime data:

```
,crime$cluster,Murder,Assault,UrbanPop,Rape
Alabama,4,13.2,236,58,21.2
Alaska,4,10,263,48,44.5
Arizona,4,8.1,294,80,31
Arkansas,3,8.8,190,50,19.5
California,4,9,276,91,40.6
Colorado,3,7.9,204,78,38.7
Connecticut,2,3.3,110,77,11.1
Delaware,4,5.9,238,72,15.8
Florida,4,15.4,335,80,31.9
```

Parameters

In the code below, we download the data using **urllib**. Then we use **np.genfromtxt** to import it to the NumPy array. Note the following parameters:

delimiter=","	The delimiter between columns.
skip_header=1	We skip the header since that has column headers and not data.
dtype=dtypes	This parameter means use the tuples (name, dtype) to convert the data using the name as the assigned numpy dtype (data type). If we don't want to assign names we would use (dtype1, dtype2, …). Note that we use the type **float**. Since NumPy is built using the C language, you can use any of the many **ctypes**, like 32 bit integers etc. We use **S12** for str as **str** converts this data to " ". You could also use unicode U12. We also could have written **np.string_** and **np.unicode_** but that does not give any length, so it means a null terminated byte, which is not a string. So, it would return a blank space. We could have used **object** as well. Note that NumPy uses these names: · dtype=) · The < sign refers to the byte order which can be **little-endian** or **big-endian**.
usecols=(1,5)	We did not use this parameter. If we had used it, it would have skipped the first column.

The code explained

Here is the code:

```
import urllib
import numpy as np
from io import StringIO
url =
"https://raw.githubusercontent.com/werowe/MLexamples/master/crime_data.csv"
file = urllib.request.urlopen(url)
data = ""
for d in file:
data = data + d.decode('utf-8')
dtypes=
arr=np.genfromtxt(StringIO(data), delimiter=",", skip_header=1,
dtype=dtypes)
```

Results in:

```
array(
```

Results in:

```
array(,
,
```

Having assigned names to columns we can refer to their name instead of index:

```
arr
```

```
array()
```

Missing values

We can tell NumPy to plug in a value for a missing value, like -1, using **missing_values**. The default behavior for floats is **np.nan**. For int it is -1.

```
Alaska,4,10,263,48,44.5
Arizona,4, ,1,294,80,31
```

That concludes this tutorial.

Related reading

- BMC Machine Learning & Big Data Blog
- Python Development Tools: Your Python Starter Kit
- Data Visualization Guide, a series of tutorials
- Snowflake Guide