008-LoadingData

Friday, March 6, 2020 5:13 PM

008-LoadingData

%pwd	#present working directory
%cd <path>	#change directory to filename path

Loading Data in Python

1. Manually loading a file.
2. Using `np.loadtxt`
3. Using `np.genfromtxt`
4. Using `pd.read_csv`
5. Using `pickle`

In [1]:

```
import numpy as np
import pickle
import pandas as pd
filename = "load.csv"
```

Please try and never manually load a file in. For your own sanity.

In [2]:

```
cols = None
data = []
with open(filename) as f:
    for line in f.readlines():
        vals = line.replace("\n", "").split(",")
        if cols is None:
            cols = vals
        else:
            data.append([float(x) for x in vals])
d0 = pd.DataFrame(data, columns=cols)
print(d0.dtypes)
d0.head()
```

```
A    float64
B    float64
C    float64
D    float64
E    float64
dtype: object
```

Out[2]:

	A	B	C	D	E
0	1.276	21.400	63.957	216.204	528.0
1	1.002	21.950	61.697	204.484	514.0
2	1.114	22.454	63.522	205.608	514.0
3	1.133	22.494	61.590	206.565	501.0
4	0.845	21.654	63.729	201.289	532.0

np.loadtxt

Good for simple data arrays with minimal formatting. Ie like data saved out using `np.savetxt` .

In [3]:

```
d1 = np.loadtxt(filename, skiprows=1, delimiter=",")
print(d1.dtype)
print(d1[:5, :])
```

```
float64
[[  1.276  21.4    63.957 216.204 528.   ]
 [  1.002  21.95   61.697 204.484 514.   ]
 [  1.114  22.454  63.522 205.608 514.   ]
 [  1.133  22.494  61.59  206.565 501.   ]
 [  0.845  21.654  63.729 201.289 532.   ]]
```

np.genfromtxt $\longrightarrow$ skip

A more flexible version of loadtxt with far better parsing. Supports different types, named columns and more.

```
In [4]:
```

```
d2 = np.genfromtxt(filename, delimiter=",", names=True, dtype=None)
print(d2.dtype)
print(d2[:5])
```

```
[('A', '<f8'), ('B', '<f8'), ('C', '<f8'), ('D', '<f8'), ('E', '<i
8')]
[(1.276, 21.4  , 63.957, 216.204, 528)
 (1.002, 21.95 , 61.697, 204.484, 514)
 (1.114, 22.454, 63.522, 205.608, 514)
 (1.133, 22.494, 61.59 , 206.565, 501)
 (0.845, 21.654, 63.729, 201.289, 532)]
```

pandas.read_csv

By far the best and most flexible CSV/txt file reader. Highly, highly recommended. If you don't believe me, just look at the obscene number of arguments you can parse to read_csv in the documentation (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html).

In [5]:

```
d3 = pd.read_csv(filename)
print(d3.dtypes)
d3.head()
```

```
A    float64
B    float64
C    float64
D    float64
E      int64
dtype: object
```

Out[5]:

	A	B	C	D	E
0	1.276	21.400	63.957	216.204	528
1	1.002	21.950	61.697	204.484	514
2	1.114	22.454	63.522	205.608	514
3	1.133	22.494	61.590	206.565	501
4	0.845	21.654	63.729	201.289	532

pickle

For when your data or object is not a nice 2D array and harder to save as something human readable. Note that if you just have a 3D, 4D... ND array of all the same type, you can also use `np.save` which will save an arbitrary numpy array in binary format. Super quick to save, super quick to load in, and small file size.

Pickle is for everything that is more complicated. You can save dictionaries, arrays, even objects.

In [6]:

```
with open("load_pickle.pickle", "rb") as f: #rb=read binary
    d4 = pickle.load(f)
print(d4.dtypes)
d4.head()
```

```
A    float64
B    float64
C    float64
D    float64
E      int32
dtype: object
```

Out[6]:

	A	B	C	D	E
0	1.276405	21.400157	63.957476	216.204466	528
1	1.002272	21.950088	61.697286	204.483906	514
2	1.114404	22.454274	63.522075	205.608375	514
3	1.133367	22.494079	61.589683	206.565339	501
4	0.844701	21.653619	63.728872	201.289175	532