PANDAS: HOW TO READ CSV & JSON FILES

Here we show how to load CSV files and JSON files into a Pandas dataframe using Pandas. (Brand new to Pandas? Get the basics in our Pandas introduction.)

This illustrates, yet again, why Pandas is so powerful. It does all the heavy lifting of downloading a file from the internet, opening it, looping through it, parsing it, and converting it to a dataframe. And it does it in a single line of code.

The Jupyter notebook for this code is here. You will need to **pip install pandas** if you don't already have that.

How to read a CSV file with Python Pandas

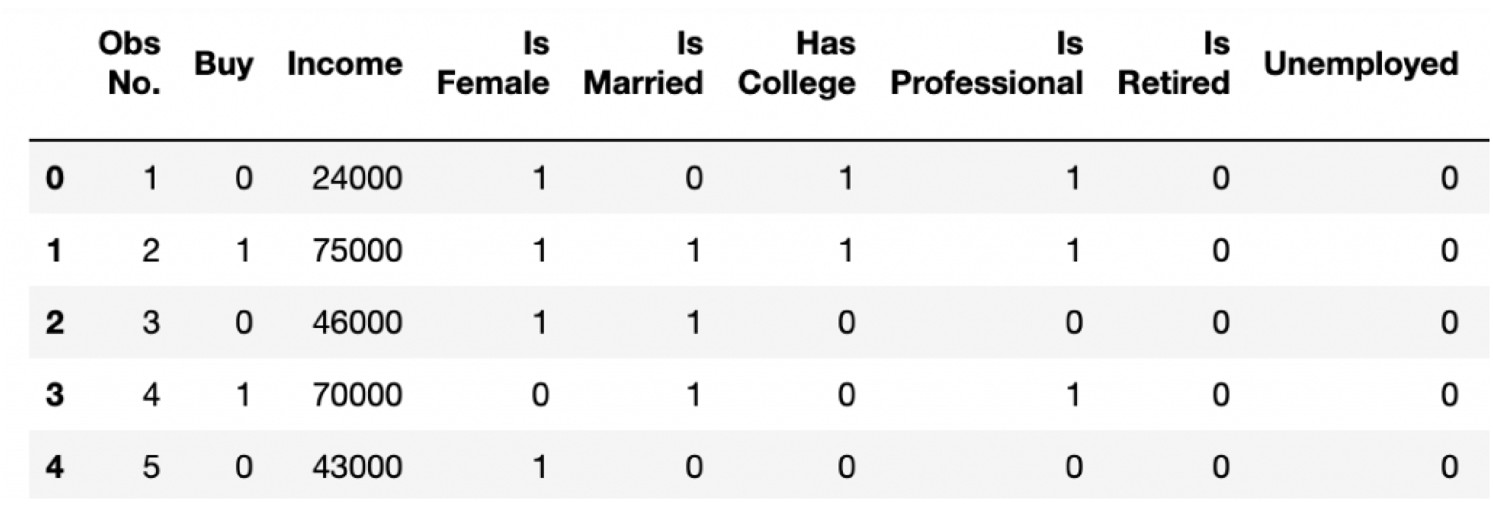
Pandas can open a URL directly. That means you don't need to download a file to read it. Below we read a .csv file:

```
import pandas as pd

url =
'https://raw.githubusercontent.com/werowe/logisticRegressionBestModel/master/
KidCreative.csv'

df = pd.read_csv(url, delimiter=',')
```

Then look at the top of it:

```
df.head()
```

The results look like this. As you can see, it parsed the file by the delimiter and added the column names from the first row in the .csv file.



	Obs No.	Buy	Income	Is Female	Is Married	Has College	Is Professional	Is Retired	Unemployed
0	1	0	24000	1	0	1	1	0	0
1	2	1	75000	1	1	1	1	0	0
2	3	0	46000	1	1	0	0	0	0
3	4	1	70000	0	1	0	1	0	0
4	5	0	43000	1	0	0	0	0	0

How to read a JSON file with Pandas

JSON is slightly more complicated, as the JSON is deeply nested. Pandas does not automatically unwind that for you.

Here we follow the same procedure as above, except we use **pd.read_json()** instead of **pd.read_csv()**.

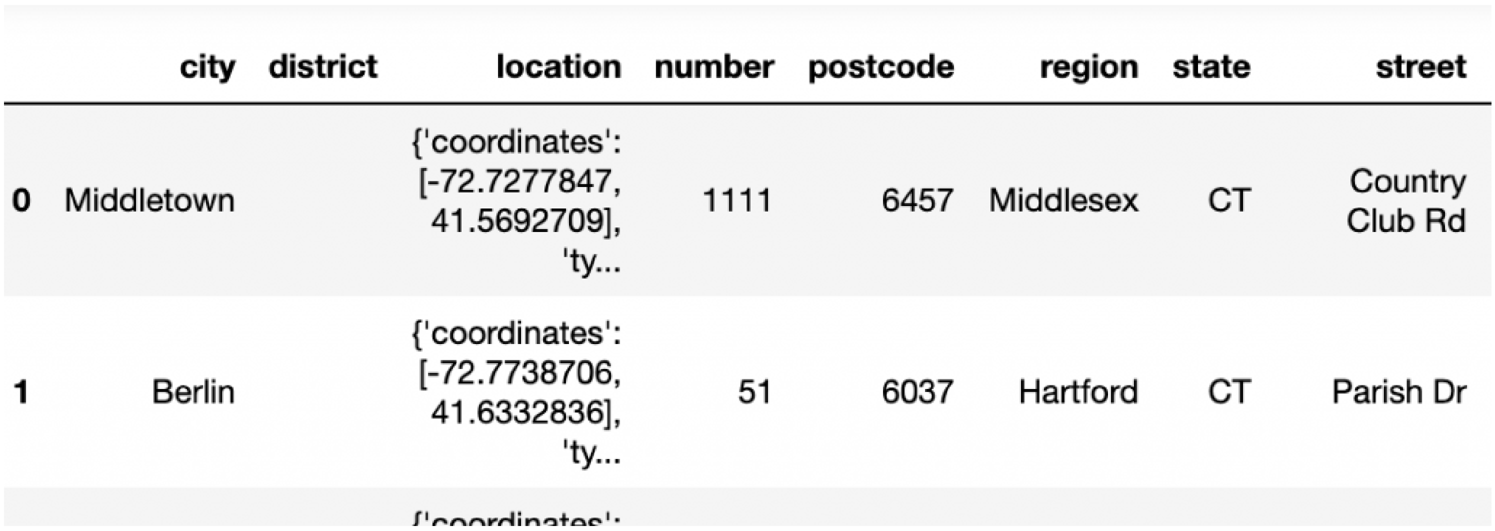
Notice that in this example we put the parameter **lines=True** because the file is in JSONP format. That means it's not a valid JSON file. Rather it is a file with multiple JSON records, one right after the other.

```
import pandas as pd

url =
'https://raw.githubusercontent.com/werowe/logisticRegressionBestModel/master/
ct1.json'

dfct=pd.read_json(url,lines=True)
```

Now look at the dataframe:

```
dfct.head()
```

Results in:



	city	district	location	number	postcode	region	state	street
0	Middletown		{'coordinates': [-72.7277847, 41.5692709], 'ty...	1111	6457	Middlesex	CT	Country Club Rd
1	Berlin		{'coordinates': [-72.7738706, 41.6332836], 'ty...	51	6037	Hartford	CT	Parish Dr
			{'coordinates':					

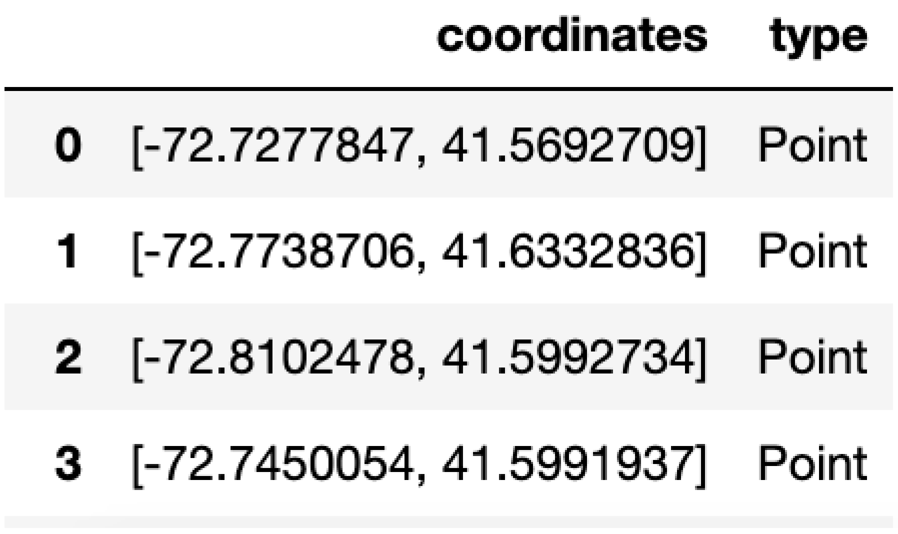
Notice that Pandas did not unwind the **location** JSON object. The input JSON looks like this:

```
{

            "state": "CT",
            "postcode": "06037",
            "street": "Parish Dr",
            "district": "",
            "unit": "",
            "location": {
                        "type": "Point",
                        "coordinates":
            },
            "region": "Hartford",
            "number": "51",
            "city": "Berlin"
}
```

So, we need an additional step. We turn the elements in location into list and then construct a DataFrame from that

```
pd.DataFrame(list(dfct))
```

Results in a new dataframe with **coordinate** and **type:**



	coordinates	type
0	[-72.7277847, 41.5692709]	Point
1	[-72.7738706, 41.6332836]	Point
2	[-72.8102478, 41.5992734]	Point
3	[-72.7450054, 41.5991937]	Point

Related reading

- BMC Machine Learning & Big Data Blog
- Python Development Tools: Your Python Starter Kit
- Creating Redshift User Defined Function (UDF) in Python
- Apache Spark Guide, a series of tutorials
- Snowflake Guide