



# Python for Data Analysis – Pandas (Continue)

Data Analysis with Pandas and Python. In this tutorial, we will begin discussing IO, or input/output, with Pandas, and begin with a realistic use-case.

Let's say we're interested in maybe purchasing or selling a home in Austin, Texas. The zipcode there is 77006. We could go to the local housing listings and see what the current prices are, but this doesn't really give us any real historical information, so let's just try to get some data on this. Let's query for "home value index 77006."

**Starting with this code, loading in a CSV to a dataframe can be as simple as:**

**Create a .py file and run it:**

```
import pandas as pd

df = pd.read_csv('ZILL-Z77006_3B.csv')

print(df.head())
```

**Output:**

	Date	Value
0	2015-06-30	502300
1	2015-05-31	501500
2	2015-04-30	500100
3	2015-03-31	495800
4	2015-02-28	492700

**What if the file doesn't have headers? No problem**

```
import pandas as pd

df = pd.read_csv('newcsv2.csv', names = ['Date','House_Price'], index_col=0)

print(df.head())
```

**Output:**

	House_Price
Date	
2015-06-30	502300
2015-05-31	501500
2015-04-30	500100
2015-03-31	495800
2015-02-28	492700

Finally, what if we want to actually rename just one of the columns? Earlier, you were shown how to name all columns, but maybe you just want to change one without having to type all the others out. Easy enough:

```
import pandas as pd

df = pd.read_csv('newcsv2.csv', names = ['Date','House_Price'])

print(df.head())

df.rename(columns={'House_Price':'Prices'}, inplace=True)

print(df.head())
```

Output:

	Date	House_Price
0	2015-06-30	502300
1	2015-05-31	501500
2	2015-04-30	500100
3	2015-03-31	495800
4	2015-02-28	492700

  

	Date	Prices
0	2015-06-30	502300
1	2015-05-31	501500
2	2015-04-30	500100
3	2015-03-31	495800
4	2015-02-28	492700

So here, we first imported the headless file, giving the column names of Date and House\_Price. Then, we decided, nope we want to call House\_Price just Price instead. So, we used df.rename, specifying that we wanted to rename columns, then, in dictionary form, the Key is the original name and the value is the new name. We finally use inplace=True so the original object is modified.

One interesting thing is the use of Pandas for conversion. So, maybe you are inputting data from a CSV, but you'd really like to display that data to HTML on your website. Since HTML is one of the datatypes, we can just export to HTML, like so:

```
import pandas as pd
```

```
df = pd.read_csv('newcsv2.csv', names = ['Date','House_Price'])
```

```
df.to_html('example.html')
```

Now we have an HTML file. Open it up, and boom you have a table in HTML.

House_Prices	
Date	
2015-06-30	502300
2015-05-31	501500
2015-04-30	500100
2015-03-31	495800
2015-02-28	492700
2015-01-31	493000
2014-12-31	494200
2014-11-30	490900

2014-10-31	486000
2014-09-30	479800
2014-08-31	473900
2014-07-31	467100
2014-06-30	461400
2014-05-31	455400
2014-04-30	450500
2014-03-31	450300

Note, this table is automatically assigned the class of "dataframe." This means you can have custom CSS to handle for dataframe-specific tables!

I particularly like to use Pandas when I have an SQL dump of data. I tend to pour the database data right into a Pandas dataframe, perform the operations that I want to perform, then I display the data in a graph maybe, or I otherwise serve the data in some way.