HOW TO CREATE A PANDAS DATAFRAME FROM A DICTIONARY

Here is yet another example of how useful and powerful Pandas is. Pandas can create dataframes from many kinds of data structures—without you having to write lots of lengthy code. One of those data structures is a dictionary.

In this tutorial, we show you two approaches to doing that.

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

A word on Pandas versions

Before you start, upgrade Python to at least 3.7. With Python 3.4, the highest version of Pandas available is 0.22, which does not support specifying column names when creating a dictionary in all cases.

If you are running **virtualenv**, create a new Python environment and install Pandas like this:

```
virtualenv py37  --python=python3.7
pip install pandas
```

You can check the Pandas version with:

```
import pandas as pd
pd.__version__
```

Create dataframe with Pandas DataFrame constructor

Here we construct a Pandas dataframe from a dictionary. We use the Pandas constructor, since it can handle different types of data structures.

The dictionary below has two keys, scene and facade. Each value has an array of four elements, so it naturally fits into what you can think of as a table with 2 columns and 4 rows.

Pandas is designed to work with row and column data. Each row has a row index. By default, it is the numbers 0, 1, 2, 3, … But it also lets you use names.

So, let's use the same in the array idx.

```
import pandas as pd

dict =  {'scene': ,

'facade': }

idx =

dp = pd.DataFrame(dict,index=idx)
```

Here is the resulting dataframe:

	facade	scene
hamlet	fair	foul
lear	beaten	murder
falstaff	fat	drunken
puck	elf	intrigue

Create dataframe with Pandas from_dict() Method

Pandas also has a **Pandas.DataFrame.from_dict()** method. If that sounds repetitious, since the regular constructor works with dictionaries, you can see from the example below that the **from_dict()** method supports parameters unique to dictionaries.

In the code, the keys of the dictionary are columns. The row indexes are numbers. That is default orientation, which is **orient='columns'** meaning take the dictionary keys as columns and put the values in rows.

```
pd.DataFrame.from_dict(dict)
```

	facade	scene
0	fair	foul
1	beaten	murder
2	fat	drunken
3	elf	intrigue

Now we flip that on its side. We will make the rows the dictionary keys.

```
pd.DataFrame.from_dict(dict,orient='index')
```

Notice that the columns have no names, only numbers. That's not very useful, so below we use the **columns** parameter, which was introduced in Pandas 0.23.

	0	1	2	3
scene	foul	murder	drunken	intrigue
facade	fair	beaten	fat	elf

It's as simple as putting the column names in an array and passing it as the columns parameter. One wonders why the earlier versions of Pandas did not have that.

```
pd.DataFrame.from_dict(dict,orient='index',columns=idx)

hamlet    lear falstaff      puck

scene    foul  murder  drunken  intrigue

facade   fair  beaten      fat       elf
```

Related reading

- BMC Machine Learning & Big Data Blog
- Pandas: How To Read CSV & JSON Files
- Apache Spark Guide, a series of tutorials
- Snowflake Guide
- Creating Redshift User Defined Function (UDF) in Python