**[[译]从列表或字典创建Pandas的DataFrame对象](https://www.cnblogs.com/everfight/p/create_dataframe_from_different_type.html)**

**原文来源：**[**http://pbpython.com/pandas-list-dict.html**](http://pbpython.com/pandas-list-dict.html)

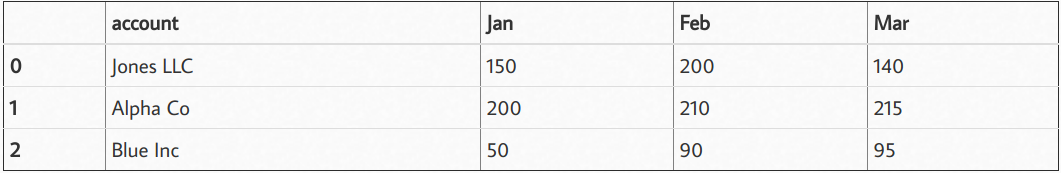


**介绍**

每当我使用pandas进行分析时，我的第一个目标是使用众多可用选项中的一个将数据导入Pandas的DataFrame 。  
对于绝大多数情况下，我使用的 read\_excel ， read\_csv 或 read\_sql 。

但是，有些情况下我只需要几行数据或包含这些数据里的一些计算。  
在这些情况下，了解如何从标准python列表或字典创建DataFrames会很有帮助。  
基本过程并不困难，但因为有几种不同的选择，所以有助于理解每种方法的工作原理。  
我永远记不住我是否应该使用 from\_dict ， from\_records ， from\_items 或默认的 DataFrame 构造函数。  
通常情况下，通过一些反复试验和错误，我能搞定它。但由于它仍然让我感到困惑，我想我会通过以下几个例子来澄清这些不同的方法。  
在本文的最后，我简要介绍了在生成Excel报表时如何使用它。

**从Python的数据结构中生成DataFrame**

您可以使用多种方法来获取标准python数据结构并创建Pandas的DataFrame。  
出于这些示例的目的，我将为3个虚构公司创建一个包含3个月销售信息的DataFrame。  


**字典**

在展示下面的示例之前，我假设已执行以下导入：

import pandas as pd

from collections import OrderedDict

from datetime import date

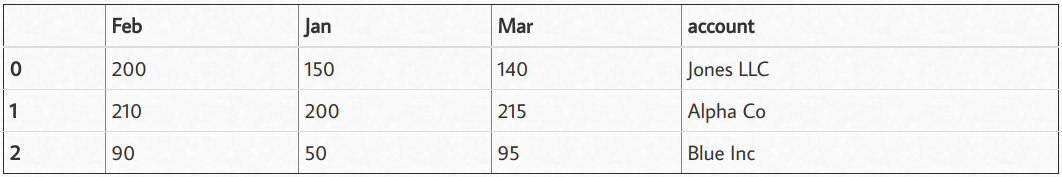
从python创建DataFrame的“默认”方式是使用字典列表。在这种情况下，每个字典键用于列标题。将自动创建默认索引：

sales = [{'account': 'Jones LLC', 'Jan': 150, 'Feb': 200, 'Mar': 140},

{'account': 'Alpha Co', 'Jan': 200, 'Feb': 210, 'Mar': 215},

{'account': 'Blue Inc', 'Jan': 50, 'Feb': 90, 'Mar': 95 }]

df = pd.DataFrame(sales)

  
如您所见，这种方法非常“面向行”。如果您想以“面向列”的方式创建DataFrame，您可以使用 from\_dict

sales = {'account': ['Jones LLC', 'Alpha Co', 'Blue Inc'],

'Jan': [150, 200, 50],sheng cheng

'Feb': [200, 210, 90],

'Mar': [140, 215, 95]}

df = pd.DataFrame.from\_dict(sales)

使用此方法，您可以获得与上面相同的结果。需要考虑的关键点是哪种方法更容易理解您独特的使用场景。  
有时，以面向行的方式获取数据更容易，而其他时候以列为导向的则更容易。  
了解这些选项将有助于使您的代码更简单，更易于理解，以满足您的特定需求。

大多数人会注意到列的顺序看起来不对。这个问题出现的原因是标准的python字典不保留其键的顺序。  
如果要控制列顺序，则有两种方式。

第一种，您可以手动重新排序列：

df = df[['account', 'Jan', 'Feb', 'Mar']]

或者你可以使用python中的OrderedDict 创建你的有序字典 。

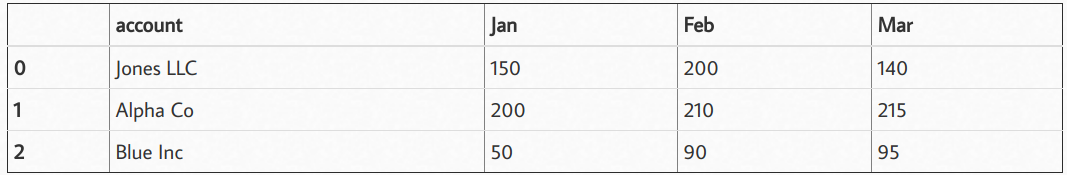
sales = OrderedDict([ ('account', ['Jones LLC', 'Alpha Co', 'Blue Inc']),

('Jan', [150, 200, 50]),

('Feb', [200, 210, 90]),

('Mar', [140, 215, 95]) ] )

df = pd.DataFrame.from\_dict(sales)

这两种方法都会按照您可能期望的顺序为您提供结果。  
  
由于我在下面概述的原因，我倾向于专门重新排序我的列，尽管使用OrderedDict一直是一个很好理解的选项。

**列表**

从python创建DataFrame的另一个选择是将数据包含在列表结构中。  
第一种方法是使用pandas进行面向行的方法 from\_records 。此方法类似于字典方法，但您需要显式调出列标签。

sales = [('Jones LLC', 150, 200, 50),

('Alpha Co', 200, 210, 90),

('Blue Inc', 140, 215, 95)]

labels = ['account', 'Jan', 'Feb', 'Mar']

df = pd.DataFrame.from\_records(sales, columns=labels)

第二种方法是 from\_items 面向列的，实际上看起来类似于 OrderedDict 上面的例子。

sales = [('account', ['Jones LLC', 'Alpha Co', 'Blue Inc']),

('Jan', [150, 200, 50]),

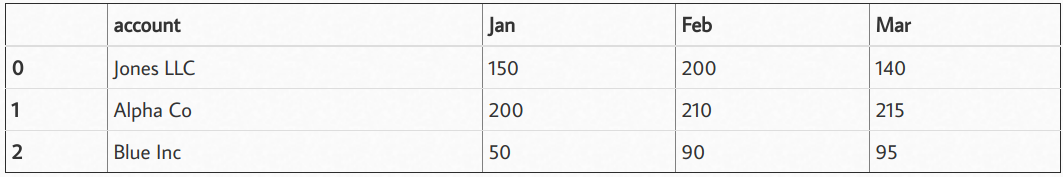
('Feb', [200, 210, 90]),

('Mar', [140, 215, 95]),

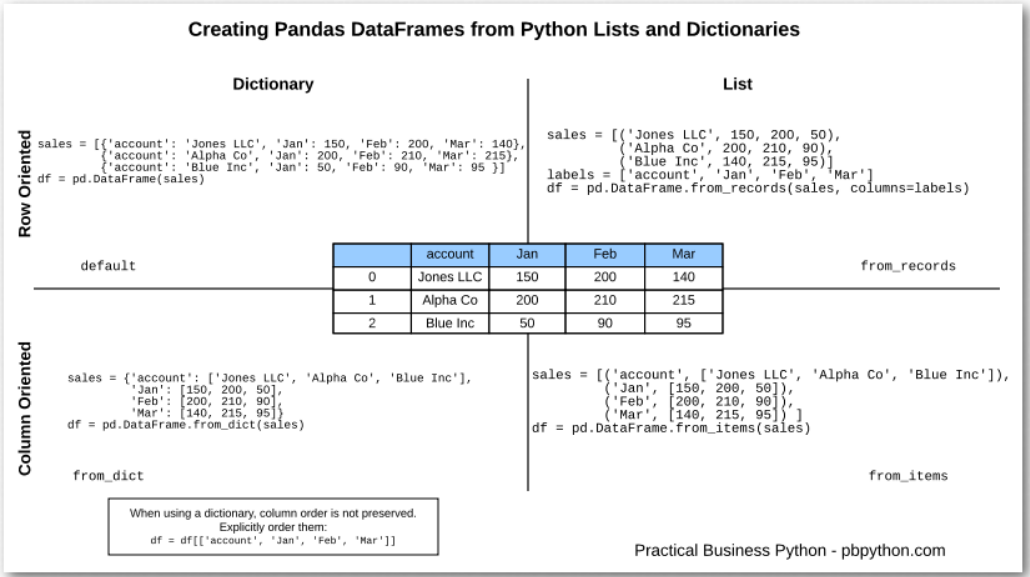
]

df = pd.DataFrame.from\_items(sales)

这两个示例都将生成以下DataFrame：



**各种选项的直观总结**

为了保持各种选项在我的脑海中清晰，我将这个简单的图形放在一起，以显示字典与列表选项以及行与列导向的方法。  
这是一个2X2的网格，所以我希望所有来询问的人都留下深刻的印象！  


为简单起见，我没有展示 OrderedDict 方法，因为这种 from\_items 方法可能更像是一个现实世界的解决方案。  
如果这有点难以阅读，您也可以获得PDF版本。

**简单的例子**

对于一个简单的概念，这似乎有很多解释。  
但是，我经常使用这些方法来构建小型DataFrame，并将其与更复杂的分析结合起来。

举一个例子，假设我们要保存我们的DataFrame并包含一个页脚，以便我们知道它何时被创建以及它是由谁创建的。  
如果我们填充DataFrame并将其写入Excel比我们尝试将单个单元格写入Excel更容易。

拿我们现有的DataFrame：

sales = [('account', ['Jones LLC', 'Alpha Co', 'Blue Inc']),

('Jan', [150, 200, 50]),

('Feb', [200, 210, 90]),

('Mar', [140, 215, 95]),

]

df = pd.DataFrame.from\_items(sales)

现在构建一个页脚（以列为导向）：

from datetime import date

create\_date = "{:%m-%d-%Y}".format(date.today())

created\_by = "CM"

footer = [('Created by', [created\_by]), ('Created on', [create\_date]), ('Version', [1.1])]

df\_footer = pd.DataFrame.from\_items(footer)

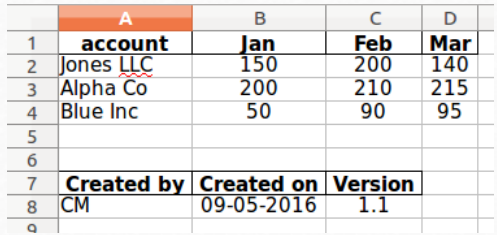
  
合并进入一个Excel中的一个sheet：

writer = pd.ExcelWriter('simple-report.xlsx', engine='xlsxwriter')

df.to\_excel(writer, index=False)

df\_footer.to\_excel(writer, startrow=6, index=False)

writer.save()

  
这里的秘诀是使用 startrow 在销售数据框架下面写入页脚DataFrame。还有一个相应的startcol，所以你可以控制成为你想要的列布局。  
这使得基本 to\_excel 功能具有很大的灵活性。

**总结**

大多数Pandas用户很快就熟悉了电子表格，CSV和SQL数据的摄取。  
但是，有时您会在基本列表或字典中包含数据并希望填充DataFrame。  
Pandas提供了几种选择，但可能并不总是立即明确何时使用哪种选择。

没有一种方法是“最好的”，它实际上取决于您的需求。  
我倾向于喜欢基于列表的方法，因为我通常关心排序，列表确保我保留顺序。  
最重要的是要知道这些选项是可用的，这样您就可以聪明地使用最简单的选项来满足您的特定情况。

从表面上看，这些代码样例看似简单，但我发现使用这些方法生成快速的信息片非常常见，他们可以增加或澄清更复杂的分析。  
DataFrame中数据的好处在于它很容易转换为其他格式，如Excel，CSV， HTML，LaTeX等。  
这种灵活性对于临时报告生成非常方便。