

# Pandas 使用教程

---

## 一、实验介绍

---

### 1.1 实验内容

Pandas 是非常著名的开源数据处理库，我们可以通过它完成对数据集进行快速读取、转换、过滤、分析等一系列操作。除此之外，Pandas 拥有强大的缺失数据处理与数据透视功能，可谓是数据预处理中的必备利器。这是 Pandas 使用教程的第 1 章节，将学会安装它，并了解 Pandas 的数据结构。

### 1.2 实验知识点

- Pandas 安装
- Pandas 数据结构

### 1.3 实验环境

- python2.7
- Xfce 终端
- ipython 终端

### 1.4 适合人群

本课程难度为一般，属于初级级别课程，适合具有 Python 基础，并对使用 Pandas 进行数据处理感兴趣的用户。

## 1.5 官方文档

学习本课程之前，你可以先自行下载官方文档（英文）作为辅助学习资料。

```
http://pandas.pydata.org/pandas-docs/stable/pandas.pdf
```

## 二、Pandas 安装

Pandas 目前支持 Python 2.7, 3.4, 3.5, 和 3.6 版本。最简单的安装方式是通过 pip 完成。你可以打开终端，键入以下命令。

```
sudo pip install pandas
```

安装过程大约持续 1 分钟作用，系统会自动下载 numpy 等依赖包。注意，本课程的全部内容基于 Pandas 0.20.3 版本，如果和你当前学习的版本存在不兼容，请通过以下命令安装 0.20.3 版本。

```
sudo pip install -v pandas==0.20.3
```

在正式学习使用 Pandas 进行数据预处理之前，我们先来了解 Pandas 的数据结构。Pandas 大致拥有 3 类数据结构，分别是一维数据 Series、二维数据 DataFrame、以及三维数据 Panel（目前依旧被融入 MultiIndex DataFrame 多维数据）。

下面的内容均在 iPython 交互式终端中演示，你可以通过在线环境左下角的应用程序菜单 > 附件打开。如果你在本地进行练习，推荐使用 Jupyter Notebook 环境。

## 三、一维数据 Series

Series 是 Pandas 中最基本的 1 维数据形式。其可以储存整数、浮点数、字符串等形式的数据。Series 的新建方法如下：

```
s = pandas.Series(data, index=index)
```

其中，data 可以是字典、numpy 里的 ndarray 对象等。index 是数据索引，索引是 pandas 数据结构中的一大特性，它主要的功能是帮助我们更快速地定位数据，这一点后面会谈到。

## 3.1 字典 -> Series

下面，我们将把不同类型的数据转换为 Series。首先是字典类型。

```
import pandas as pd

d = {'a' : 10, 'b' : 20, 'c' : 30}
print pd.Series(d)
```

```
In [3]: print pd.Series(d)
a      10
b      20
c      30
dtype: int64
```



这里，数据值是 10, 20, 30，索引为 a, b, c。我们可以直接通过 index= 参数来设置新的索引。

```
import pandas as pd
d = {'a' : 10, 'b' : 20, 'c' : 30}

s = pd.Series(d, index=['b', 'c', 'd', 'a'])
print s
```

```
In [7]: print s
b      20.0
c      30.0
d       NaN
a      10.0
dtype: float64
```



你会发现，pandas 会自动匹配人为设定的索引值和字典转换过来的索引值。而当索引无对应值时，会显示为 NaN 缺失值。

## 3.2 ndarray -> Series


ndarray 是著名数值计算包 numpy 中的多维数组。我们也可以将 ndarray 直接转换为 Series。

```
import pandas as pd
import numpy as np

data = np.random.randn(5) # 一维随机数
index = ['a', 'b', 'c', 'd', 'e'] # 指定索引

s = pd.Series(data, index)
print s
```

```
In [13]: print s
a      0.576915
b      0.940897
c     -1.539771
d      0.019964
e     -1.110628
dtype: float64
```




上面的两个例子中，我们都指定了 index 的值。而当我们非人为指定索引值时，Pandas 会默认从 0 开始设置索引值。

```
s = pd.Series(data)
print s
```

```
In [16]: s = pd.Series(data)

In [17]: print s
0      0.576915
1      0.940897
2     -1.539771
3      0.019964
4     -1.110628
dtype: float64
```



当我们需要从一维数据 Series 中返回某一个值时，可以直接通过索引完成。

```
import pandas as pd
import numpy as np

data = np.random.randn(5) # 一维随机数
index = ['a', 'b', 'c', 'd', 'e'] # 指定索引

s = pd.Series(data, index)
print s
print s['a']
```

```
In [24]: print s
a      0.817076
b     -0.106519
c     -2.083681
d     -1.050392
e      0.962920
dtype: float64
```

```
In [25]: print s['a']
0.817076480105
```



除此之外，Series 是可以直接进行运算的。例如：

```
import pandas as pd
import numpy as np

data = np.random.randn(5) # 一维随机数
index = ['a', 'b', 'c', 'd', 'e'] # 指定索引

s = pd.Series(data, index)
print s
print 2*s
print s-s
```

```
In [30]: s = pd.Series(data, index)
```

```
In [31]: print s
```

```
a    -1.117403  
b     0.049504  
c     0.285953  
d    -1.613987  
e    -0.938777  
dtype: float64
```

```
In [32]: print 2*s
```

```
a    -2.234806  
b     0.099009  
c     0.571905  
d    -3.227974  
e    -1.877554  
dtype: float64
```

```
In [33]: print s-s
```

```
a     0.0  
b     0.0  
c     0.0  
d     0.0  
e     0.0  
dtype: float64
```



## 四、二维数据 DataFrame

DataFrame 是 Pandas 中最为常见、最重要且使用频率最高的数据结构。你可以想到它箱型为电子表格或 SQL 表具有的结构。DataFrame 可以被看成是以 Series 组成的字典。它和 Series 的区别在于，不但具有行索引，且具有列索引。

DataFrame 可以用于储存多种类型的输入：

- 一维数组、列表、字典或者 Series 字典。
- 二维 numpy.ndarray。
- 结构化的 ndarray。
- 一个 Series。
- 另一个 DataFrame。

### 4.1 Series 字典 -> DataFrame

```
import pandas as pd
```

```
# 带 Series 的字典
```

```
d = {'one' : pd.Series([1., 2., 3.], index=['a', 'b', 'c']), 'two' : pd.  
.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}
```

```
df = pd.DataFrame(d) # 新建 DataFrame
```

```
print df
```

```
In [37]: print df
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0



我们可以看到，这里的行索引为 a, b, c, d，而列索引为 one, two。

## 4.2 ndarrays 或 lists 字典 -> DataFrame

```
import pandas as pd
```

```
# 列表构成的字典
```

```
d = {'one' : [1, 2, 3, 4], 'two' : [4, 3, 2, 1]}
```

```
df1 = pd.DataFrame(d) # 未指定索引
```

```
df2 = pd.DataFrame(d, index=['a', 'b', 'c', 'd']) # 指定索引
```

```
print df1
```

```
print df2
```

```
In [42]: print df
```

	one	two
0	1	4
1	2	3
2	3	2
3	4	1

```
In [43]: print df2
```

	one	two
a	1.0	4.0
b	2.0	3.0
c	3.0	2.0
d	4.0	1.0



注意观察它们之间的不同。

## 4.3 带字典的列表 -> DataFrame

```
import pandas as pd

# 带字典的列表
d = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]

df = pd.DataFrame(d)

print df
```

```
In [51]: print df
```

```
   a    b    c
0  1    2  NaN
1  5   10  20.0
```



## 4.4 DataFrame.from\_ 方法

pandas 的 DataFrame 下面还有 4 个以 from\_ 开头的方法，这也可以用来创建 Dataframe。

```
pd.DataFrame.from
```

```
(m) from_csv(cls, path, header, sep, index_col, pa... DataFrame
(m) from_dict(cls, data, orient, dtype) DataFrame
(m) from_items(cls, items, columns, orient) DataFrame
(m) from_records(cls, data, index, exclude, column...
```

例如：

```
import pandas as pd

d = [('A', [1, 2, 3]), ('B', [4, 5, 6])]
c = ['one', 'two', 'three']

df = pd.DataFrame.from_items(d, orient='index', columns=c)

print df
```



```
In [56]: print df
   one  two  three
A     1    2     3
B     4    5     6
```

## 4.5 列选择，添加，删除

接下来，我们延续上面的 4.4 里面的数据来演示。

在一维数据结构 Series 中，我们用 `df['标签']` 来选择行。而到了二维数据 DataFrame 中，`df['标签']` 表示选择列了。例如：

```
print df['one']
```

```
In [58]: print df
   one  two  three
A     1    2     3
B     4    5     6
```

```
In [59]: print df['one']
A     1
B     4
Name: one, dtype: int64
```

删除列的方法为 `df.pop('列索引名')`，例如：

```
df.pop('one')
print df
```

```
In [69]: print df
   one  two  three
A     1    2     3
B     4    5     6
```

← 原始数据

```
In [70]: df.pop('one')
Out[70]:
A     1
B     4
Name: one, dtype: int64
```

← 执行删除 one 列

```
In [71]: print df
   two  three
A     2     3
B     5     6
```

← 输出剩下数据

添加列的方法未 `df.insert(添加列位置索引序号, '添加列名', 数值)` , 例如 :

```
df.insert(3, 'four', [10, 20])  
print df
```

```
In [76]: print df  
   one  two  three  
A     1    2     3  
B     4    5     6  
  
In [77]: df.insert(3, 'four', [10, 20])  
  
In [78]: print df  
   one  two  three  four  
A     1    2     3    10  
B     4    5     6    20
```

添加列



## 五、三维数据 Panel

Panel 是 Pandas 中使用频率较低的一种数据结构 , 但它是三维数据的重要容器。

### 5.1 面板数据

Panel data 又称面板数据 , 它是计量经济学中派生出来的一个概念。在计量经济学中 , 数据大致可分为三类 : 截面数据 , 时间序列数据 , 以及面板数据。而面板数据即是截面数据与时间序列数据综合起来的一种数据类型。

简单来讲 , 截面数据指在某一时间点收集的不同对象的数据。而时间序列数据是指同一对象在不同时间点所对应的数据集合。

这里引用一个城市和 GDP 关系的示例来解释上面的三个概念 ( 面板数据 (<https://baike.baidu.com/item/%E9%9D%A2%E6%9D%BF%E6%95%B0%E6%8D%AE/6483369>) ) :

截面数据 :

- 例如城市 : 北京、上海、重庆、天津在某一年的 GDP 分别为10、11、9、8 ( 单位亿元 ) 。

时间序列数据:

- 例如：2000、2001、2002、2003、2004 各年的北京市 GDP 分别为8、9、10、11、12（单位亿元）。

面板数据：

- 2000、2001、2002、2003、2004 各年中国所有直辖市的 GDP 分别为（单位亿元）：北京市分别为 8、9、10、11、12；上海市分别为 9、10、11、12、13；天津市分别为 5、6、7、8、9；重庆市分别为 7、8、9、10、11。

## 5.2 Panel 构成

在 Pandas 中，Panel 主要由三个要素构成：

- items: 每个项目（item）对应于内部包含的 DataFrame。
- major\_axis: 每个 DataFrame 的索引（行）。
- minor\_axis: 每个 DataFrame 的索引列。

简而言之，在 Pandas 中，一个 Panel 由多个 DataFrame 组成。下面就生成一个 Panel。

```
import pandas as pd
import numpy as np

wp = pd.Panel(np.random.randn(2, 5, 4), items=['Item1', 'Item2'], major_axis=pd.date_range('1/1/2000', periods=5), minor_axis=['A', 'B', 'C', 'D'])

print wp
```

```
In [4]: print wp
<class 'pandas.core.panel.Panel'>
Dimensions: 2 (items) x 5 (major_axis) x 4 (minor_axis)
Items axis: Item1 to Item2
Major_axis axis: 2000-01-01 00:00:00 to 2000-01-05 00:00:00
Minor_axis axis: A to D
```

我们可以看到，wp 由 2 个项目、5 个主要轴和 4 个次要轴组成。其中，主要轴由 2000-01-01 到 2000-01-05 这 5 天组成的时间序列，次轴从 A 到 D。

你可以输出 Item1 看一看。

```
print wp['Item1']
```

```
In [6]: print wp['Item1']
```

	A	B	C	D
2000-01-01	1.520241	-0.072490	-0.406636	1.166041
2000-01-02	0.936788	0.159986	0.752802	3.117203
2000-01-03	-0.631340	-0.110650	0.779653	2.135608
2000-01-04	0.246908	-0.480319	0.341618	1.090066
2000-01-05	-0.302249	-0.111367	-1.940412	-0.070270



再看一看 Item2。

```
print wp['Item2']
```

```
In [7]: print wp['Item2']
```

	A	B	C	D
2000-01-01	-1.371400	0.528583	0.727346	-1.036593
2000-01-02	-1.384153	-0.043796	-0.151988	1.323957
2000-01-03	1.063050	1.377388	0.037660	1.020970
2000-01-04	1.841794	-0.905643	-0.280490	-0.421995
2000-01-05	-0.737487	-1.039275	1.382705	-0.111830



可以看到，这两个 Dataframe 的行索引及列索引是一致的。由于数据是随机生成的，所以不一致。

## 5.2 Panel 的未来

由于 Panel 在 Pandas 中的使用频率远低于 Series 和 DataFrame，所以 Pandas 决定在未来的版本中将 Panel 移除，转而使用 MultiIndex DataFrame 来表示多维数据结构。

这里，可以用到 Panel.to\_frame() 输出多维数据结构。就拿上面的例子继续：

```
print wp.to_frame()
```

```
In [8]: print wp.to_frame()
```

		Item1	Item2
major	minor		
2000-01-01	A	1.520241	-1.371400
	B	-0.072490	0.528583
	C	-0.406636	0.727346
	D	1.166041	-1.036593
2000-01-02	A	0.936788	-1.384153
	B	0.159986	-0.043796
	C	0.752802	-0.151988
	D	3.117203	1.323957
2000-01-03	A	-0.631340	1.063050
	B	-0.110650	1.377388
	C	0.779653	0.037660
	D	2.135608	1.020970
2000-01-04	A	0.246908	1.841794
	B	-0.480319	-0.905643
	C	0.341618	-0.280490
	D	1.090066	-0.421995
2000-01-05	A	-0.302249	-0.737487
	B	-0.111367	-1.039275
	C	-1.940412	1.382705
	D	-0.070270	-0.111830



## 六、实验总结

这一章节，我们着重介绍了 Pandas 的数据结构，只有熟悉了这三种（尤其是前两种）数据结构之后，才能对后面采用 Pandas 进行数据预处理有更深刻的理解。

## 七、课后作业

你知道 Pandas 的名字是怎么来的吗？

答案：点击 ([http://labfile.oss.aliyuncs.com/courses/906/pandas\\_name.txt](http://labfile.oss.aliyuncs.com/courses/906/pandas_name.txt))

\*本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。

下一节：Pandas 常用的基本方法 (</courses/906/labs/3376/document>)