[Pandas 读写json](https://geek-docs.com/pandas/pandas-read-write/pandas-reading-and-writing-json.html)

**Pandas 读写json**，Json是最常用的标准数据格式之一，特别是web数据的传输，通常在使用这些数据之前，需要对数据格式进行处理。本章会介绍常用的几个处理json的API函数。

* read\_json：从json文件中读取数据
* to\_json：将数据写入到json文件中
* json\_normalize：对json数据进行规范化处理

阅读本章前，可以先了解Pandas相关的基础知识：[Pandas基础教程](http://geek-docs.com/pandas/pandas-tutorials/pandas-tutorial.html)， [Pandas数据处理](http://geek-docs.com/pandas/pandas-data-process/pandas-data-processing.html)

**JSONViewer**

**JSONViewer** 是用于检测Json格式是否正确的一个在线应用工具，其网址是http://jsonviewer.stack.hu/， 输入或复制JSON数据到这个web应用中，就可以检测其格式是否合法，还可以以树状结构显示数据，方便大家理解数据的结构，如下所示：

{

"firstName": "John",

"lastName": "Smith",

"gender": "man",

"age": 32,

"address": {

"streetAddress": "21 2nd Street",

"city": "New York",

"state": "NY",

"postalCode": "10021"

},

"phoneNumbers": [

{ "type": "home", "number": "212 555-1234" },

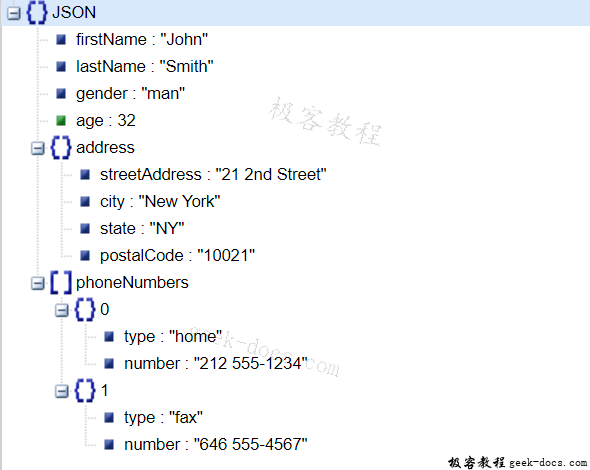
{ "type": "fax", "number": "646 555-4567" }

]

}

JSON

Copy

树状结构效果如下图所示：  


**写入json数据**

如果有一个DataFrame,需要将其转换成JSON文件，首先定义一个[DataFrame](http://coolcou.com/pandas/pandas-tutorials/pandas-create-dataframe.html)对象，然后调用它的to\_json()函数，传入你要创建的json文件名作为参数。如下所示：

import pandas as pd

import numpy as np

frame = pd.DataFrame(np.arange(16).reshape(4, 4),

index=['white', 'black', 'red', 'blue'],

columns=['up', 'down', 'right', 'left'])

print(frame)

frame.to\_json('frame.json')

Python

Copy

输出结果如下:

up down right left

white 0 1 2 3

black 4 5 6 7

red 8 9 10 11

blue 12 13 14 15

Shell

Copy

frame.json文件的内容如下:

{"up":{"white":0,"black":4,"red":8,"blue":12},"down":{"white":1,"black":5,"red":9,"blue":13},"right":{"white":2,"black":6,"red":10,"blue":14},"left":{"white":3,"black":7,"red":11,"blue":15}}

**读取json数据**

读取json数据也很简单，使用to\_json()函数，传入文件名作为参数即可。如下所示:

import pandas as pd

frame = pd.read\_json('frame.json')

print(frame)

输出结果如下:

down left right up

black 5 7 6 4

blue 13 15 14 12

red 9 11 10 8

white 1 3 2 0

上述例子中因为frame.json文件是由DataFrame对象转换而来，JSON数据为列表形式。然而，json文件中的数据通常不是列表形式,因此，需要将字典结构的文件转成列表形式，这个过程就称为**规范化**。

**规范化**

Pandas 库中的 json\_normalize()函数能够将字典或列表转换成表格，使用前，可以通过如下方式导入这个函数：

from pandas.io.json import json\_normalize

创建json文件，将其保存到工作目录下，文件名为books.json，文件内容如下:

[{"writer":"mark Ross",

"nationality":"USA",

"books":[

{"title":"XML Cookbook", "price":23.56},

{"title":"Python Fundamentals","price":50.70},

{"title":"The Numpy library","price":12.30}

]

},

{"writer":"Barbara Bracket",

"nationality":"UK",

"books": [

{"title":"Java Enterprise","price":28.60},

{"title":"HTML5","price":31.35},

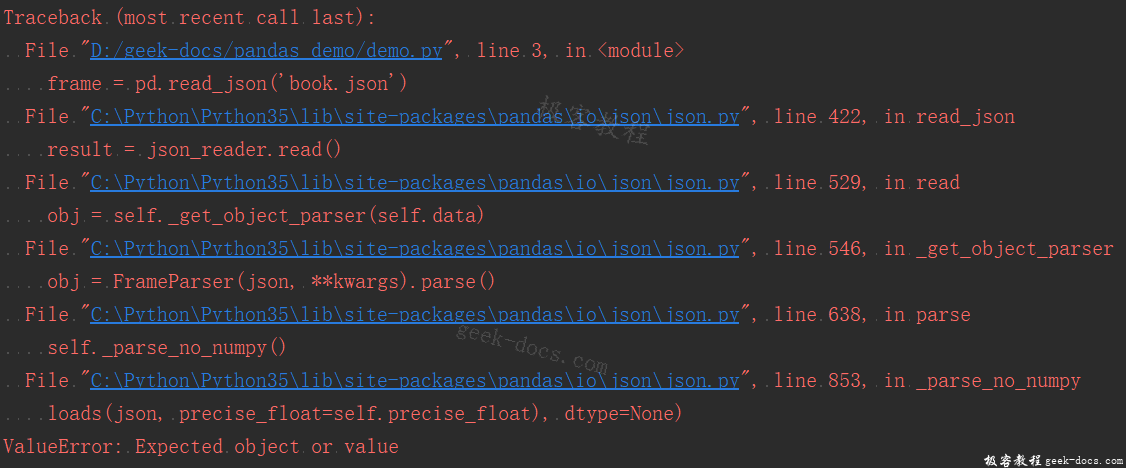
{"title":"Python for Dummies", "price":28.00}

]

}]

JSON

Copy

如上所示，文件结构不再是列表形式，而是一种更为复杂的形式。因此无法再使用read\_json()函数来处理，执行报错如下所示:  


我们仍然可以从这个数据结构中获取到列表数据，首先，加载JSON文件的内容，并将其转换为一个字符串。

import json

import pandas as pd

file = open("books.json", "r")

text = file.read()

text = json.loads(text)

然后调用 json\_normalize()函数进行规范化处理，如下所示，你可能想得到一个包含所有图书信息的表格，只要把键books作为第二个参数即可。

import json

import pandas as pd

from pandas.io.json import json\_normalize

file = open("books.json", "r")

text = file.read()

text = json.loads(text)

print(json\_normalize(text, 'books'))

输出结果如下:

price title

0 23.56 XML Cookbook

1 50.70 Python Fundamentals

2 12.30 The Numpy library

3 28.60 Java Enterprise

4 31.35 HTML5

5 28.00 Python for Dummies

json\_normalize()函数会读取所有books作为键的元素的值，元素的所有属性将会转换为嵌套的列名称，而属性值将会转换为DataFrame元素，该函数使用一串递增的数字作为索引。

你得到的DataFrame对像只包含了一部分内部信息，增加books位于同一级的其他键的值可能会由用处，把存储键名的列表作为第三个参数传入即可。如下所示:

import json

import pandas as pd

from pandas.io.json import json\_normalize

file = open("books.json", "r")

text = file.read()

text = json.loads(text)

print(json\_normalize(text, 'books',['writer', 'nationality']))

输出结果如下:

price title writer nationality

0 23.56 XML Cookbook mark Ross USA

1 50.70 Python Fundamentals mark Ross USA

2 12.30 The Numpy library mark Ross USA

3 28.60 Java Enterprise Barbara Bracket UK

4 31.35 HTML5 Barbara Bracket UK

5 28.00 Python for Dummies Barbara Bracket UK